

Article

Emulation-Based Analysis of Multiple Cell Upsets in LEON3 SDRAM: A Workload-Dependent Vulnerability Study

Afef Kchaou ^{1,2,*} , Sehmi Saad ³  and Hatem Garrab ^{4,5} ¹ IUT Bordeaux, GEII Department, 15 Street of Naudet, 33170 Gradignan, France² Campus El Manar, Sciences Faculty of Tunis, Tunis El Manar University, 20 Street of Tolède, Tunis 2092, Tunisia³ RF2S Spectrum Solutions, 18 Street of the Faïencerie, 33300 Bordeaux, France; sehmi.saad@gmail.com⁴ Electronics and Micro-Electronic Laboratory (LEμE), Bd de L'environnement, Monastir 5000, Tunisia; garhat@yahoo.fr⁵ Higher Institute of Applied Sciences and Technology of Sousse, University of Sousse, Street Taher Ben Achour, Sousse 4003, Tunisia

* Correspondence: kchaouafef@gmail.com; Tel.: +33-652893317

Abstract

The reliability of embedded processors in safety- and mission-critical domains is increasingly threatened by radiation-induced soft errors, particularly multiple-cell upsets (MCUs) that simultaneously corrupt adjacent cells in external SDRAM. While prior studies on the LEON3 processor have largely focused on single-event upsets (SEUs) in internal SRAM structures, they overlook MCU effects in off-chip SDRAM, a critical gap that limits fault coverage and compromises system-level reliability assessment in modern high-density embedded systems. This paper presents an SDRAM-based fault injection framework using FPGA emulation to evaluate the impact of MCUs on the LEON3 soft-core processor, with faults directly injected into the external memory subsystem where data corruptions can rapidly propagate into system-level failures. The methodology injects spatially correlated two-bit MCUs directly into SDRAM during realistic workload execution. Three architecturally diverse benchmarks were analyzed, each representing a distinct computational workload: a numerical (matrix multiplication), signal-processing (FFT), and a cryptographic (AES-128 encryption) application, chosen to capture arithmetic-intensive, iterative, and control-intensive execution profiles, respectively. The results reveal a distinct workload-dependent vulnerability profile. Matrix multiplication exhibited >99.99% fault activation, with outcomes overwhelmingly dominated by data store errors. FFT showed >97% activation in steady-state execution, following an initial phase sensitive to alignment and data access exceptions. AES displayed 88.12% non-propagating faults, primarily due to injections in inactive memory regions, but remained exposed to critical memory access violations and control-flow exceptions that enable fault-based cryptanalysis. These findings demonstrate that SEU-only models severely underestimate real-world MCU risks and underscore the necessity of selective, workload-aware fault-tolerance strategies: lightweight ECC for cryptographic data structures, alignment monitoring for signal processing, and algorithm-based fault tolerance (ABFT) for numerical kernels. This work provides actionable insights for hardening LEON3-based systems against emerging multi-bit threats in radiation-rich and adversarial environments.

Keywords: LEON3; FPGA-based fault emulation; multiple-cell upsets (MCUs); SDRAM reliability; fault tolerance; vulnerability

Academic Editors: Ruyi Huang,
Xi Zhang and Jianguo Miao

Received: 21 October 2025

Revised: 14 November 2025

Accepted: 21 November 2025

Published: 23 November 2025

Citation: Kchaou, A.; Saad, S.; Garrab, H. Emulation-Based Analysis of Multiple Cell Upsets in LEON3 SDRAM: A Workload-Dependent Vulnerability Study. *Electronics* **2025**, *14*, 4582. <https://doi.org/10.3390/electronics14234582>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Fault emulation has emerged as a cornerstone methodology for evaluating the resilience of embedded processors to radiation-induced soft errors, offering a practical balance between the realism of hardware experiments and the scalability of simulation. Within this domain, FPGA-based emulation has become particularly powerful, enabling near-real-time injection of faults into synthesized designs while executing realistic workloads. Recent advances include Xilinx's TMR SEM IP, which supports runtime fault injection and correction via dynamic partial reconfiguration [1–5], and ICAP/JTAG-based platforms that enable bit-accurate injection of multi-cell upsets (MCUs) without disrupting system execution [4–7]. These capabilities make FPGA emulation the method of choice for studying workload-dependent fault propagation in soft-core processors like LEON3.

Despite significant progress, a critical research gap persists in the characterization of MCUs affecting external SDRAM in FPGA-based embedded systems. Most prior LEON3 studies have concentrated on single event upsets (SEUs) and single event transients (SETs) within on-chip SRAM structures such as caches, register files, and tightly coupled memories [8–22]. However, external SDRAM, which typically stores program data, heap, and stack segments, has received little attention, even though its dense cell geometry and charge-sharing effects make it highly susceptible to spatially correlated multi-bit errors [23]. The lack of MCU-oriented investigations in SDRAM limits fault coverage and hinders the development of accurate reliability models for modern embedded processors.

The consequences of this gap are twofold. First, system-level reliability may be overestimated, as internal hardening mechanisms (e.g., TMR, ECC) do not protect off-chip memory [24]. Second, in cryptographic workloads, undetected SDRAM faults can enable fault-based attacks (e.g., differential fault analysis) that bypass traditional side-channel protections, yet SEU-centric models fail to capture these multi-bit threats.

To address this limitation, this work proposes a hardware-based MCU fault injection framework that directly targets the SDRAM subsystem of the LEON3 soft-core processor implemented on a Xilinx Virtex-5 FPGA. Unlike traditional SEU emulation, which perturbs individual bits in isolation, the proposed approach injects spatially correlated two-bit upsets across adjacent SDRAM cells, closely reflecting multi-bit failure patterns observed in radiation tests [25]. This design enhances fault realism and diversity, enabling the analysis of correlated error propagation paths and their impact on system behavior during runtime execution.

The study analyzes three architecturally distinct benchmarks to capture the workload-dependent nature of MCU effects:

- Matrix multiplication (arithmetic- and memory-intensive),
- FFT (iterative, alignment-sensitive signal processing), and
- AES-128 (control-intensive, security-critical cryptographic kernel).

Together, these benchmarks cover numerical, signal-processing, and cryptographic domains, offering a comprehensive evaluation of MCU-induced fault behaviors under distinct operational profiles.

Experimental results reveal clear workload-dependent fault characteristics. Matrix multiplication exhibited nearly 100% data store errors with negligible masking, indicating direct vulnerability to memory corruption. FFT showed address alignment and data access exceptions during early injections, transitioning to store-dominated failures in later campaigns. AES demonstrated a high proportion of non-propagating faults (~88%), though a small fraction of unmasked faults triggered instruction and data access exceptions with potential security implications. These findings demonstrate that MCU-induced errors propagate differently across workload classes and emphasize the importance of selective, application-aware fault-tolerance strategies in embedded processor design.

The remainder of this paper is organized as follows. Section 2 reviews related work on fault injection methodologies and LEON3 reliability analyses. Section 3 details the proposed MCU fault injection framework and experimental setup. Section 4 presents and discusses the results of the emulation campaigns across the three benchmarks. Section 4 presents and analyzes the experimental results for MulMatrix, FFT, and AES benchmarks. It then provides a contextual comparison with prior SEU-centric studies and a detailed workload-based comparative analysis. Section 5 concludes with key findings, design implications, and future research directions.

2. Related Work

The LEON3 soft-core processor has become a widely adopted reference standard in aerospace and satellite systems owing to its modular design, open availability, and proven track record in space missions. LEON3 is highly susceptible to radiation-induced faults, which may compromise reliability under harsh environments. To address these challenges, numerous studies have investigated its vulnerability and mitigation through diverse fault injection (FI) methodologies.

Early simulation-based approaches include the work of Abbasitabar et al. [8], who analyzed error propagation in flip-flops, registers, register files, and L1/L2 caches of the LEON3 processor using VHDL simulation, and the work of Mansour and Velazco [9], who extended this analysis with a direct fault injection (DFI) method targeting register-level vulnerabilities. At a higher level of abstraction, SystemC-based transaction-level modeling has been adopted to accelerate fault injection campaigns using dynamic binary instrumentation (DBI) techniques. This approach enables faster evaluation of the system-level impacts of transient faults [10].

Several works have also explored hybrid fault-tolerant implementations. For instance, Lindoso et al. [11] combined single-error correction and double-error detection (SEC/DED), software validation, and configuration memory scrubbing in a LEON3 processor implemented on Artix-7 FPGAs, demonstrating enhanced resilience with modest overhead. Similarly, in our previous work [12,13], we performed an exhaustive SEU injection campaign into the SRAM of LEON3, providing detailed insights into workload sensitivity, error propagation mechanisms, and overall processor vulnerability, which informed the design of more robust mitigation techniques.

More recent studies have introduced sophisticated FI platforms and fault models. For example, Sari et al. [14,15] developed an open-source RapidSmith-based FI framework capable of injecting SEUs in configuration memory, while leveraging on-chip microcontrollers and performance counters to analyze fault effects. In a different context, Yuce et al. [16] analyzed pipeline-level vulnerabilities of a 7-stage LEON3 running AES on a Spartan-6 FPGA, demonstrating how FI attacks can compromise cryptographic applications. Similarly, Chekmarev et al. [17,18] proposed an IP-core based FI infrastructure integrated directly into system-on-chips (SoCs), simplifying the environment and enabling long autonomous campaigns on LEON3.

To address emerging fault models such as single-event multiple transients (SEMTs), Nethula et al. [19] proposed a formal framework using satisfiability modulo theories (SMT) to evaluate layout-aware SEMT vulnerabilities in a LEON3 ASIC core, reporting propagation probabilities of 30% for SEMT3 and 54% for SEMT4. On the other hand, Travessini et al. [20] used simulation-based fault injection into CPU registers, revealing that a small subset of registers accounts for the majority of critical failures, and showing that a partial TMR strategy covering only these registers achieved >99% SBU tolerance with marginal area overhead.

Complementing these works, Mannos et al. [21] developed a hardware-accelerated gate-level FI platform on Xilinx UltraScale boards, achieving a $\sim 7200\times$ speedup compared to simulation. Their study, focused on AES-128 execution on LEON3, uncovered 22 distinct faulty behaviors, including key and plaintext leakage through the serial port, with delay faults proving the most effective at inducing exploitable leakage. This underscores the importance of efficient FI platforms and highlights the security implications of subtle fault mechanisms.

Building further, our recent work [22] presented a comprehensive register transfer-level (RTL) FI campaign on the program counter (PC) of LEON3 using the enhanced NETFI+ framework. Over four million SEUs, multiple-bit upsets (MBUs), and single event transients (SETs) were systematically injected across all pipeline stages. The analysis revealed stage-dependent vulnerabilities: early stages (Fetch, Decode) were highly prone to halts and traps, intermediate stages (Register Access, Execute) produced more silent data corruptions, while later stages (Memory, Exception) benefited from fault masking. SETs were mostly transient but occasionally induced control-flow anomalies. This study highlighted the need for pipeline-stage-specific hardening strategies and demonstrated the diagnostic value of simulation-based FI for early design validation.

These works demonstrate the breadth of FI methodologies applied to LEON3, ranging from simulation-based and formal models to hardware-accelerated and attack-oriented platforms, providing both dependability insights and security assessments. Table 1 summarizes fault injection methodologies and findings on the LEON3 processor.

While the studies summarized in Table 1 provide valuable insights into LEON3 fault tolerance, none address the combined impact of (i) multiple-cell upsets (MCUs), (ii) external SDRAM as the fault injection target, and (iii) diverse real-world workloads. Most prior efforts focus on single-bit faults (SEUs/SETs) in internal structures (registers, caches, configuration memory) and often use synthetic or single-application benchmarks. In contrast, this work is the first to emulate spatially correlated two-bit MCUs directly in the SDRAM subsystem of LEON3 under three architecturally distinct workloads (numerical, signal-processing, cryptographic), enabling a workload-dependent vulnerability analysis that reveals new fault manifestations (e.g., alignment traps in FFT, control-flow exceptions in AES) and quantifies the limitations of SEU-centric models for system-level reliability assessment.

The workload-dependent vulnerability patterns identified in this study stem primarily from SDRAM physics (cell density, charge sharing) and application memory access behavior, rather than LEON3-specific microarchitecture. Consequently, these findings are expected to generalize to other embedded soft-core processors, including RISC-V, ARM Cortex-R, and MicroBlaze, that similarly rely on external SDRAM for data-intensive workloads in safety-critical systems.

Beyond LEON3-specific studies, FI has been broadly employed as one of the most effective methodologies to evaluate the resilience of FPGA-based systems against transient and permanent faults. At the simulation level, netlist- and HDL-based tools (e.g., the frameworks of Fibich et al. and Monopoli et al.) introduce controlled perturbations into logic descriptions, enabling systematic analysis with detailed observability [26,27]. Higher-level platforms such as RASP-FIT automate design modifications to accelerate large-scale FI campaigns [28,29]. Hardware-based approaches, on the other hand, leverage physical disturbances including heavy-ion irradiation [30], laser beams [31], or voltage glitches [32], offering high fidelity at the cost of limited accessibility. In between these two extremes, FPGA-accelerated emulation methods, such as the approaches of López-Ongil et al. [33] and Entrena et al. [34], provide a practical compromise by injecting faults directly in hardware while executing realistic workloads, thus combining realism with scalability.

Table 1. Survey of fault injection approaches applied to the LEON3 processor.

Ref./Year	Methodology	Fault Model	Target	Main Findings
[8]/2012	VHDL simulation + FPGA emulation	SETs, SEUs, MBUs, METs	Flip-flops, Register file, Registers, Cache memories,	Characterized soft error propagation and masking in integer and multiplier unit.
[9]/2013	FPGA emulation campaigns	SEUs	Register-level units	Analyzed vulnerability of registers under SEUs
[10]/2015	Structural pipeline analysis combined with experimental biased FI attacks	Fault attacks modeled via biased fault injection	7-stage pipeline LEON3 (Spartan-6 FPGA)	Identified most vulnerable pipeline stages for AES; highlighted risks in cryptographic software
[11]/2017	Hybrid design (SEC/DED codes, hardware monitors, software validation, scrubbing) on Artix-7	SEUs + control-flow errors	Full LEON3 system	Demonstrated improved reliability with multi-layer mitigation
[12,13]/2018	Large-scale FPGA-based emulation (>200 k injections)	SEUs	Xilinx SRAM-based FPGA hosting LEON3 focuses SRAM memories	Provided detailed characterization of error propagation, workload sensitivity, and limits of detection mechanisms
[14,15]/2017	Open-source FI platform (RapidSmith) with on-chip microcontroller injects/monitors	SEUs injected at configuration bit/frame level	Xilinx SRAM-based FPGA hosting LEON3 focuses on configuration memory and per-component sensitivity	Estimates soft-error sensitivity per bit/frame
[16]/2015	Structural pipeline analysis combined with experimental biased FI attacks	DFIA + clock glitch	7-stage pipeline LEON3 (Spartan-6 FPGA)	Identified most vulnerable pipeline stages for AES; highlighted risks in cryptographic software
[17,18]/2015	Embedded IP-core fault injector inside SoC	SEUs	LEON3 in SoC	Simplifies fault injection setup; enables autonomous, long campaigns
[19]/2022	Formal SMT-based framework with layout-aware analysis	SEMTs (SEMT3, SEMT4)	LEON3 ASIC core	Estimated propagation probability. 30% (SEMT3), 54% (SEMT4); rigorous vulnerability assessment
[20]/2018	Simulation-script-based FI	SBUs	LEON3 CPU registers	Identified small set of highly vulnerable registers
[21]/2019	Gate-level FI using synthesizable saboteur + scan chain; hardware-accelerated on UltraScale FPGAs	Static faults (incl. delay faults)	LEON3 running AES-256	Identified 22 faulty behaviors incl. crypto/memory leakage; delay faults most effective; ~7200× faster than simulation
[22]/2025	Netlist fault injection based simulation	SEUs, MBUs, SETs	LEON3 Program Counter across pipeline stages	Early/middle stages (FE–EX) highly vulnerable; ME/XC resilient; SET mostly masked but risky.

Within this framework, SRAM-based FPGAs have received particular attention due to their extreme sensitivity to radiation. The configuration memory, composed of millions of SRAM cells, controls all functional aspects of the device. Early emulation tools such as FLIPPER [35] enabled high-throughput SEU injection in configuration bits, while later frameworks like ACME [36] or ACME-2 [37] introduced selective injection targeting essential bits, reducing experimental overhead without sacrificing fault representativeness. The Xilinx soft error mitigation (SEM) IP [38] and ICAP-based approaches [4–8] further advanced this field by supporting runtime fault injection and correction, enabling non-intrusive experimentation. These developments have established FPGA-based emulation as a cost-effective and flexible alternative to beam testing [39,40]. A comprehensive tutorial about emulation-based fault injection platforms for SRAM-based FPGAs can be found in Ref. [41].

While most FI studies initially focused on single-bit upsets (SBUs), recent evidence from radiation tests shows that MCUs, where a single particle strike flips several adjacent bits, occur frequently in modern high-density technologies. This has motivated several MCU-aware fault injection strategies. For instance, Pérez-Celis et al. [25] reproduced realistic MCU spatial patterns observed in beam experiments, demonstrating significantly higher error rates than single-bit fault models. Other studies have proposed clustered injection strategies to emulate correlated upsets in configuration memory, while dual-circuit modeling has been employed to evaluate MCU effects in user memories such as register files and caches [42,43]. These contributions highlight the importance of extending fault injection beyond SEUs to accurately capture the impact of radiation-induced phenomena.

Despite this progress, relatively few studies have explicitly investigated MCU effects in the context of LEON3. Existing LEON3 fault injection campaigns have predominantly focused on SEUs and SETs, characterizing error propagation and evaluating mitigation techniques [44] such as TMR [45,46], ECC [47], and memory scrubbing [48]. However, the susceptibility of LEON3's SDRAM memories, especially under spatially correlated MCU events, remains largely unexplored. Addressing this gap, the present work introduces an emulation-based MCU fault injection methodology for the LEON3 processor, combining dynamic modeling of two-bit upsets in adjacent SDRAM cells with benchmark-driven evaluation. This approach provides new insights into the resilience of LEON3 under realistic MCU scenarios and contributes to the broader understanding of fault tolerance in FPGA-based aerospace processors.

3. MCU Fault Injection Methodology on SDRAM LEON3

While SEUs assume isolated bit flips, modern high-density SDRAM is increasingly prone to spatially correlated MCUs due to technology scaling, charge sharing, and reduced critical charge [23,49]. Empirical beam testing by Pérez-Celis et al. [25] demonstrated that over 70% of radiation-induced memory errors in nanoscale devices involve exactly two adjacent bits, with multi-bit clusters dominating the error profile. By modeling spatially correlated two-bit upsets directly in external SDRAM, our methodology expands fault coverage beyond traditional SEU models, which fail to capture these realistic, correlated multi-bit disturbances that overwhelm conventional single-error correction codes [23,49].

3.1. The LEON3 Soft-Core Processor and Its SDRAM Subsystem as a Case Study

The LEON3 processor, developed by Cobham Gaisler® as part of the GRLIB IP library, is a synthesizable 32-bit soft-core CPU implementing the SPARC V8 architecture. It is widely adopted in space, avionics, and other safety- and security-critical domains thanks to its configurability, open-source availability, and radiation-hardened implementations [50,51]. As a fully synthesizable design, the LEON3 can be mapped onto FPGAs, which makes it

a suitable platform for emulation-based fault injection studies where accurate, real-time observation of fault effects is essential.

At the heart of LEON3's architecture lies its memory hierarchy, which combines both on-chip SRAM-based storage and external memory components:

- Instruction and data caches, typically configurable between 1 KB and 64 KB, implemented using FPGA block RAMs (BRAMs). These accelerate memory access by storing recently used instructions and data.
- Register file, holding general-purpose and control registers, implemented as SRAM-based structures in FPGA realizations.
- Scratchpad or tightly coupled memories (TCMs), optional on-chip SRAM memories used for deterministic data access in real-time or safety-critical applications.
- External memory subsystem, managed by the LEON3 memory controller (MCTRL), which supports several memory devices such as SRAM, PROM, and SDRAM.

The MCTRL plays a crucial role in LEON3's memory organization by mapping external memories into specific address regions, as illustrated in Figure 1. Its configuration enables flexible partitioning of system memory across chip-select (CS) regions. Of particular importance in this work is the main memory region (SDRAM mapped to CS0), which spans the address range 0x40000000 to 0x7FFFFFFF. This region typically hosts program data, heap, and stack segments during benchmark execution. As such, it contains the majority of runtime variables that are directly exposed to soft errors, including state arrays, intermediate matrices, and key data structures. The SDRAM represents the most fault-prone component due to its dynamic cell architecture, high density, and reduced critical charge in modern process technologies.

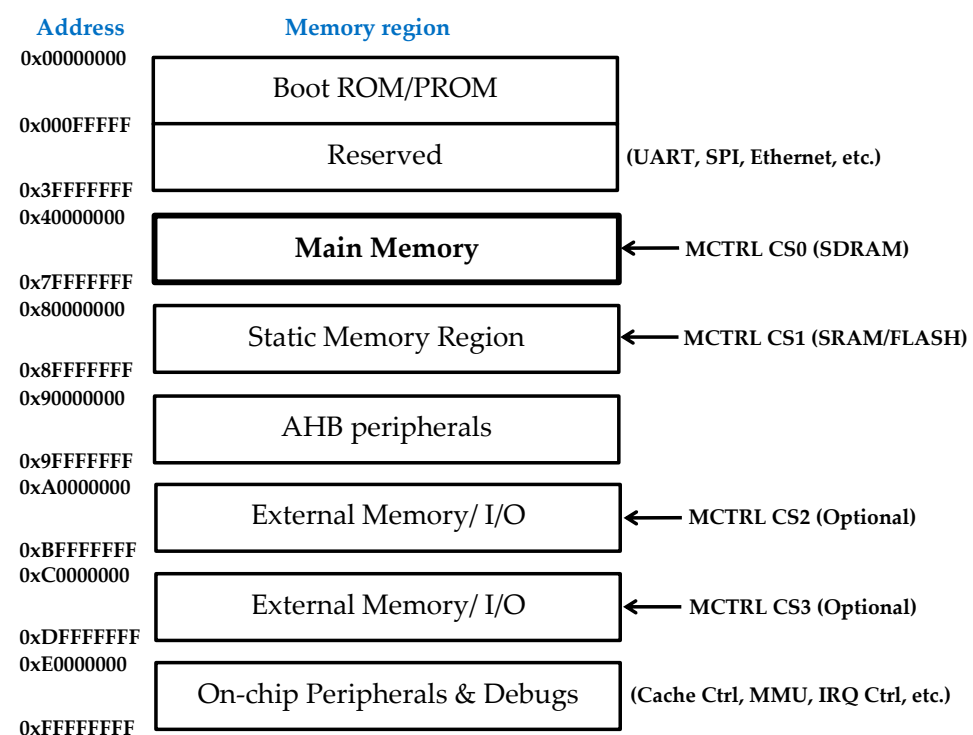


Figure 1. LEON3 processor memory map.

Due to technology scaling, SDRAM cells have become increasingly sensitive to radiation-induced effects such as SEUs and MCUs. While SEUs affect isolated bits, MCUs simultaneously flip multiple adjacent cells, an effect exacerbated by charge sharing and shrinking cell capacitance in nanometer-scale DRAM technologies. In FPGA-based LEON3 systems, this vulnerability is particularly critical: although on-chip SRAM structures

(caches, scratchpads) benefit from spatial isolation and lower density, the external SDRAM remains exposed to high-probability multi-bit corruption events.

3.2. Benchmark Selection

To capture the full spectrum of workload-dependent MCU vulnerability in embedded systems, three representative benchmarks were selected to fundamentally represent distinct computational archetypes commonly found in safety- and mission-critical applications:

- **Matrix Multiplication** (30×30 character-type matrices): an arithmetic- and memory-intensive workload that stresses sustained data movement and compute density. This benchmark is representative of numerical kernels in aerospace, scientific computing, and real-time control systems. In this benchmark, intermediate matrix elements are stored in SDRAM and repeatedly retrieved during computation, making the memory subsystem the dominant source of vulnerability.
- **Fast Fourier Transform (FFT)**: a signal-processing workload characterized by iterative feedback, regular memory access patterns, and high sensitivity to address alignment. FFT is widely used in radar, communications, and sensor signal processing, where small perturbations can significantly alter output integrity. Even minor errors in intermediate results can propagate rapidly through the computation pipeline, making FFT an effective vehicle for exposing error propagation mechanisms.
- **AES-128 encryption**: a control-intensive, security-critical workload that stresses control logic, stateful round-based execution, and irregular memory access patterns through lookup tables (S-boxes). As a cryptographic standard in secure communications, AES is highly sensitive to data corruption, even single-bit errors in key schedule or state arrays can compromise correctness and enable fault-based attacks [52]. Both the key schedule and state arrays reside in SDRAM during execution, directly exposing security-critical data to memory faults.

This deliberate selection ensures that observed fault manifestations reflect generalizable vulnerability patterns across major embedded application domains, numerical, signal-processing, and cryptographic, rather than idiosyncrasies of a single algorithm [53,54]. The benchmarks are written in C and compiled using the SPARC GCC cross-compiler for LEON3, generating compatible executable applications.

3.3. Proposed MCU Fault Injection Methodology

This study introduces a dedicated fault injection framework specifically designed to emulate multiple-cell upsets in the SDRAM of the LEON3 soft-core processor. Figure 2 illustrates the experimental architecture, composed of the following key components:

- **Host computer**: defines and controls injection campaigns, manages benchmark execution, and collects output data.
- **Fault injection controller (FIC)**: injects MCU events into specified SDRAM addresses according to configurable parameters (e.g., fault type, address range, fault location).
- **Monitor/Checker module**: tracks processor activity during execution to identify erroneous outputs, exceptions, or system stalls.
- **Benchmark suite**: numerical and cryptographic workloads running on the LEON3 to represent distinct application domains.

The setup is implemented on a Xilinx Virtex-5 FPGA (ML507), where the LEON3 core and its off-chip SDRAM act as the primary targets. All experiments were conducted on a hardware emulation platform based on a single Xilinx FPGA device, with the LEON3 hosted as the design under test (DUT). A host computer connected via a JTAG interface controlled the injections through the JTAG configuration manager (JCM), which handled FPGA configuration, fault activation, and monitoring.

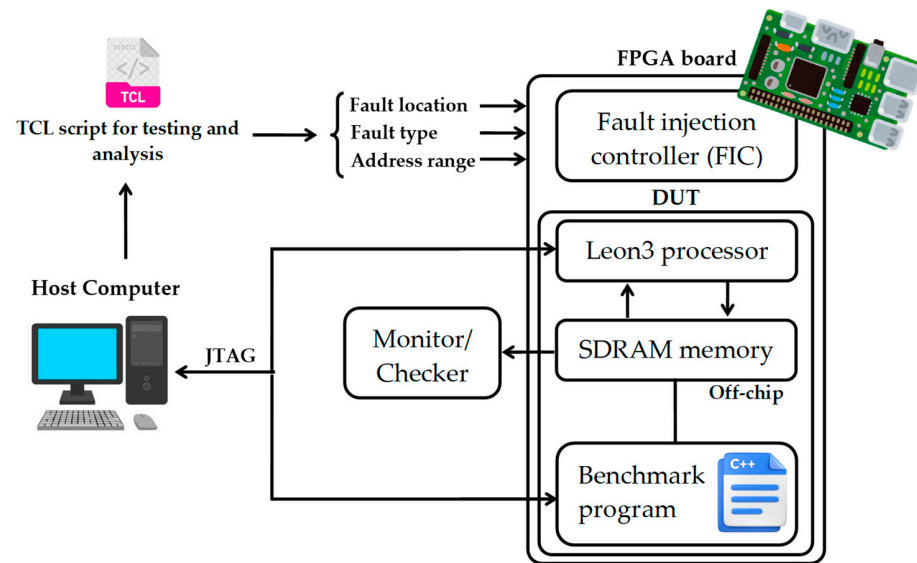


Figure 2. Architecture of emulation fault injection.

The fault injection campaigns were performed under controlled conditions with the following parameters:

- **Timing:** Faults were injected during active benchmark execution, after the completion of the initialization phase and before final output generation, ensuring perturbations affect only the computational core of each workload. Injections occurred across multiple clock cycles to capture timing-dependent vulnerabilities.
- **Spatial location:** The data segment resides in the SDRAM address range 0x40000000–0x7FFFFFFF, where each address corresponds to a 128-bit memory size. Each SDRAM address was partitioned into four words. Within each 128-bit, one bit was flipped in the least significant bit (LSB) position of the first word and one bit in the LSB of the second adjacent word, emulating a spatially correlated two-bit MCU within the same physical memory access unit. This pattern was applied exhaustively across all valid data-segment addresses.
- **Total injection events:** The experiments comprised 661,870 injections for MulMatrix, 662,870 for FFT, and 661,888 for AES, ensuring comprehensive coverage of the active memory footprint under each workload.

The MCU injection mechanism operates through a two-stage scripting process. In the first stage, the memory layout is dynamically mapped during benchmark loading, with the runtime location of the data segment determined through allocation analysis. In the second stage, the injector flips one bit in each of two physically adjacent SDRAM cells, exhaustive emulating a two-bit upset.

As shown in Figure 3, each campaign follows a cycle of injection and evaluation: after a fault is injected, the benchmark resumes execution, and the ‘END INJ’ signal initiates post-injection analysis. Input vectors are applied, the resulting outputs are stored in text files and compared against expected results, and discrepancies are logged before the process advances to the next injection site. This iterative methodology enables a systematic and large-scale exploration of MCU-induced error behaviors under representative workloads.

In a limited number of campaign cases, the last event of the injected fault consistently forced the processor into a stopped mode, requiring manual reconfiguration of the FPGA to resume testing. This recurring behavior highlights a practical limitation of emulation-based fault injection, where halted states reduce throughput and complicate automation in large-scale campaigns. The described procedure is used to conduct the fault injection emulation experiments described in the next section.

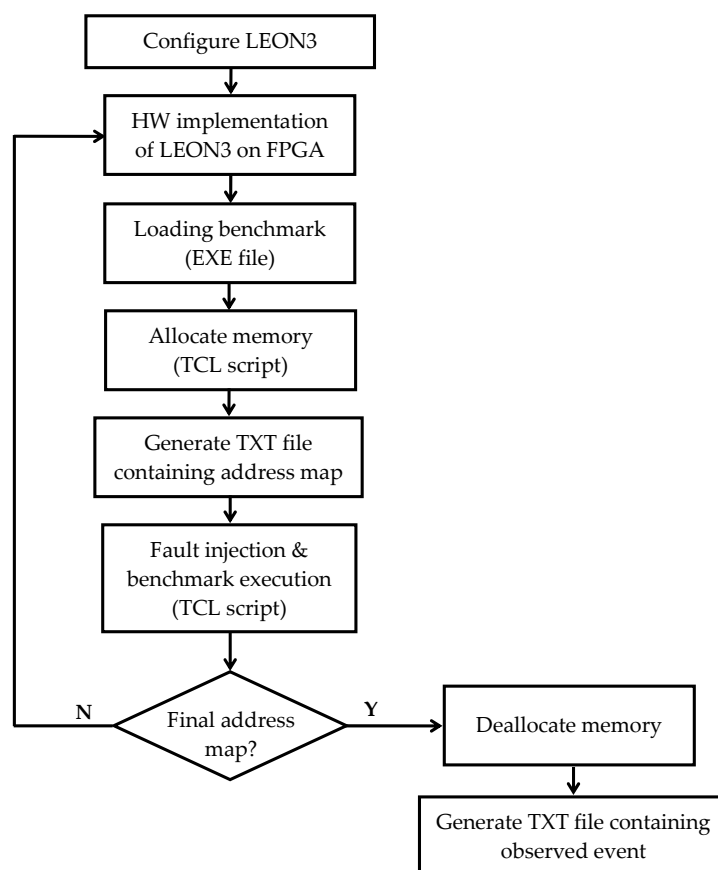


Figure 3. Fault injection procedure.

4. Experimental Results and Analysis

4.1. MCU Fault Manifestations in MulMatrix Execution

This section presents the classification and analysis of MCU fault injection results obtained during the execution of the matrix multiplication benchmark (30×30 char type) on the LEON3 soft-core processor. Table 2 shows the fault injection results by executing the MulMatrix benchmark.

Result classification revealed that the LEON3 processor exhibits a pronounced vulnerability to MCUs in the SDRAM. Across all campaigns, only a single masked fault was observed, while the remainder produced visible system effects. This extremely low masking rate contrasts with typical SEU fault injection studies [13], where natural masking by application behavior or hardware redundancy often results in much higher tolerance.

Table 2. MulMatrix benchmark: fault injection results.

FI Campaign (Total FI: 661,870)	Masked Faults	Silent Faults	Data Store Errors	Stopped Mode
116,497 (Series 1)	0	0	116,496	1
53,058 (Series 2)	1	0	53,056	1
59,970 (Series 3)	1	0	59,968	1
116,364 (Series 4)	1	0	116,362	1
55,490 (Series 5)	1	0	55,488	1
116,362 (Series 6)	1	0	116,360	1
144,129 (Series 7)	1	0	144,128	0

The experimental campaigns revealed that data store errors overwhelmingly dominated the fault outcomes, representing nearly 99.99% of all observed failures. In the LEON3 architecture, such traps occur when the processor receives an AMBA error response for a memory location that is about to be accessed. Specifically, in this study, the failures correspond to the trap “IU in error mode (tt = 0x2B, data store error)”, which indicates a write buffer exception. This condition can arise during sub-word store operations that trigger a read–modify–write cycle, where the EDAC mechanism reports an error on the read phase [55,56]. The high concentration of store-related corruptions aligns with the characteristics of the executed benchmark, which is dominated by frequent and regular memory write operations. Consequently, corrupted data are rapidly propagated and amplified through subsequent arithmetic computations.

In contrast, stopped mode faults were consistently observed as the last injection for each series, reflecting either fatal control-flow corruption or unhandled traps. Similar limitations in handling halted states have been reported in our previous work [13], which emphasized the challenges of maintaining automated injection workflows on FPGA platforms without integrated reset mechanisms.

Overall, the results confirm that the LEON3 processor exhibits minimal resilience to MCU-induced faults when executing memory-bound workloads. The near absence of masked faults indicates that virtually all injected upsets propagate to observable failures. These results are consistent with earlier SEU studies on LEON3 and comparable architectures, which did not quantify the proportion of faults leading to system-level failures [13]. Table 3 summarizes the observed fault categories, including:

- data store errors ($\approx 99.99\%$ in each campaign);
- stopped mode and masked faults (negligible for each type).

Table 3. MulMatrix benchmark: classification traps and faults by categories.

Fault Category	Description	Detection Mechanisms	Example Behavior	Observed Frequency
Data store error	Incorrect value written to memory, Write buffer error	ECC, parity	Corrupted data in memory	Predominant ($>99.99\%$ of faults in each series)
Stopped mode	Processor enters halted state	Watchdog, exception flags	System stops execution	Always present (1 per campaign, negligible)
Masked fault	Fault that does not affect program output	N/A	No visible effect	Always present (1 per campaign, negligible)

This distribution demonstrates the strong bias of MCU effects toward immediate, detectable corruption, with minimal opportunities for natural fault tolerance. Such outcomes reaffirm that while MCUs present a severe threat to system reliability, dedicated architectural and coding-level countermeasures can significantly enhance resilience in radiation-prone or safety-critical environments.

4.2. MCU Fault Manifestations in FFT Execution

This section presents the classification and analysis of MCU fault injection results obtained during the execution of the FFT benchmark on the LEON3 soft-core processor. The experimental outcomes are summarized in Table 4, which details the distribution of fault manifestations and their impact on system behavior.

Table 4. FFT benchmark: fault injection results.

FI Campaign (Total FI: 662,870)	Masked Faults	Silent Faults	Data Access Exception	Data Store Errors	Mem. Address Not Aligned	Stopped Mode
10,622 (Series 1)	146	3	1086	556	8830	1
66,818 (Series 2)	830	0	0	65,987	0	1
78,018 (Series 3)	2424	0	1	75,592	1	1
90,930 (Series 4)	830	0	0	90,099	0	1
87,490 (Series 5)	830	0	0	86,659	0	1
67,650 (Series 6)	830	0	0	66,819	0	1
67,778 (Series 7)	830	0	0	66,947	0	1
91,858 (Series 8)	830	0	0	91,027	0	1
74,434 (Series 9)	830	0	0	73,603	0	1
27,272 (Series 10)	830	0	0	25,441	0	0

In the first campaign (series 1), the FFT benchmark exhibited a diverse fault profile. A relatively high number of masked faults (146 cases) and a few silent faults (3 cases) were observed, alongside substantial occurrences of data access exceptions (1086 cases). These exceptions, reported as “IU in error mode (tt = 0x09, data access exception)”, occur when the processor receives an AHB error response during load or store operations. If the corrupted word corresponds to the pipeline’s requested data, a trap is triggered; otherwise, the cache memory controller suppresses the valid bit without raising an exception [56]. In addition, a large number of unaligned memory access exceptions (8830 cases) were recorded, indicated by “IU in error mode (tt = 0x07, mem address not aligned)” [55]. These errors point to vulnerabilities in memory addressing consistency, attributed to MCU-induced disturbances in address registers. Their occurrence strongly depends on the timing of injections relative to pointer or index updates. Interestingly, these misalignment errors disappeared in later campaigns, suggesting that such events are highly sensitive to runtime conditions such as temporal coincidence and memory allocation patterns [56].

In subsequent campaigns (series 2–10), the fault behavior shifted dramatically, becoming dominated by data store errors, which consistently accounted for more than 97% of all injected faults (e.g., 90,099 in series 4 and 91,027 in series 8). Each campaign ultimately terminated in a stopped mode, requiring a manual reset of the emulation platform, thereby underscoring the disruptive impact of MCUs on sustained FFT execution.

Masked faults remained present in later campaigns, typically averaging around 830 per run (≈ 0.9 – 3.1% of injections, depending on campaign size). By contrast, silent faults, initially observed in the first campaign, did not reappear in subsequent runs. This suggests that once the FFT benchmark reached a steady-state execution, MCU effects were either detected through traps or manifested as store errors, with virtually no silent data corruptions escaping detection.

The overall distribution indicates that FFT, as a computation- and memory-intensive benchmark, is particularly sensitive to data corruption, where memory writes dominate the execution profile and propagate errors rapidly through the computation pipeline. The rarity of alignment and access exceptions beyond the first series suggests that these were more likely tied to initial memory setup and pointer handling rather than to the steady-state FFT benchmark itself. Table 5 summarizes the observed fault categories and traps, including:

- data store errors (~ 93 – 99% of faults in Series 2–10; $\sim 5\%$ in Series 1);
- stopped mode faults, silent faults (negligible for each type);

- masked faults (~1–3%) and;
- misaligned memory/data access exception (83% misaligned, 10% access in Series 1).

Table 5. FFT benchmark: classification traps and faults by categories.

Fault Category	Description	Detection Mechanisms	Example Behavior	Observed Frequency
Data store error	Corruption during write operations to SDRAM, leading to inconsistent or invalid data values	Detected through execution mismatches and exception handling	Incorrect FFT output values, rapid propagation of numerical errors	Dominant (~93–99% of faults in Series 2–10; ~5% in Series 1)
Stopped mode	System halts due to unrecoverable error or watchdog trigger	Requires manual reset of FPGA platform	Emulation stopped; execution cannot proceed	Always present (1 per campaign, negligible fraction)
Masked fault	Faults that do not alter program state or output	Inherent masking through algorithm redundancy or unused resources	Execution proceeds normally without errors	Low (~1–3% across campaigns)
Silent faults	Errors that propagate without detection	Not detected by hardware/software mechanisms	Incorrect results without triggering exceptions	Rare (only 3 cases, ~0.03% in Series 1)
Data access exceptions/misalignment memory	Invalid memory addressing due to MCU-induced corruption	Exception handlers in LEON3 pipeline	Addressing trap, memory misalignment	Significant in Series 1 (83% not aligned, 10% access), absent later

This distribution confirms that MCU effects during FFT execution overwhelmingly manifest as data store corruptions, with only marginal contributions from masked or silent faults. The results underline the FFT benchmark’s susceptibility to memory-related disturbances.

4.3. MCU Fault Manifestations in AES Execution

To evaluate the resilience of the LEON3 processor against MCUs in security-critical workloads, fault injection campaigns were conducted during the execution of the AES-128 encryption algorithm. Because cryptographic routines are highly sensitive to even minor data corruptions, MCU-induced errors can lead to severe consequences, including incorrect ciphertext generation or, in adversarial contexts, side-channel vulnerabilities that expose secret keys. Thus, evaluating the LEON3 soft-core processor under AES execution provides critical insights into its fault tolerance in secure embedded applications.

The fault injection experiments were carried out using the same emulation-based platform as in the matrix multiplication benchmark, with the LEON3 soft-core processor running AES on an FPGA device. Faults were injected exhaustively into the processor’s SDRAM during AES execution, and system responses were classified according to their impact. Table 6 provides a breakdown of the fault injection results obtained while running the AES benchmark.

The analysis revealed that, unlike the matrix multiplication workload, the AES application exhibited a diverse set of fault manifestations. Data store errors remained the dominant category in the first series; their frequency was somewhat diminished (11.67%) compared to the high percentage (>97%) seen in FFT and matrix multiplication benchmarks. Although halted executions (i.e., “stopped mode”) occurred in only a small number of injections (one case in our campaign), they reflect the algorithm’s

control-intensive nature: corrupted loop indices or state variables can disrupt AES's round-based structure, preventing further progress.

Table 6. AES benchmark: fault injection results.

FI Campaign (Total FI: 661,888)	Masked Faults	Silent Faults	Illegal Instruction	Data Access Exception	Data Store Errors	Mem. Address Not Aligned	Instruction Access	Stopped Mode
83,139 (Series 1)	4561	82	1	1273	77,212	8	1	1
578,695 (Series 2)	578,695	0	0	0	0	0	0	0

In addition to the dominant store errors and halted executions, rare exceptions were observed during the campaigns. Instruction-related traps were recorded only once each, specifically instruction access and illegal instruction events. These anomalies are most likely attributable to MCU-induced corruption of control words or instruction fetch addresses, which subsequently result in invalid opcode decoding. The instruction access exception is reported as “IU in error mode (tt = 0x01, instruction access exception)”. This trap occurs when the processor receives an AHB error response while fetching instructions. In normal operation, if the error arises during a prefetch of an instruction that is not immediately required, the cache controller simply avoids setting the corresponding valid bit in the instruction cache tag. Should the IU later attempt to execute from that address, a cache miss forces a retry, thereby triggering the instruction access exception [55,56]. Illegal instructions, in contrast, were identified by the message “IU in error mode (tt = 0x02, illegal instruction)”, corresponding to the decoding of an invalid opcode.

Another category of faults consisted of data access exceptions, which accounted for approximately 0.2% of all experiments, with a slightly higher rate (1.53%) observed in the first injection campaign. While relatively infrequent, these faults can cause severe disruptions in program execution and, in the case of AES, compromise the correctness of cryptographic outputs. Notably, no such exceptions occurred during the matrix multiplication benchmark, reinforcing the observation that control-intensive workloads expose distinct vulnerability patterns compared to purely arithmetic ones.

A final, particularly rare class of errors involved unaligned memory access exceptions, occurring in approximately 0.01% of cases during the first injection series.

Interestingly, the AES campaign exhibited a high proportion of non-propagating faults (88.12%), compared to matrix multiplication (<0.001%). However, this apparent “masking” is largely attributable to faults injected in unused SDRAM regions or transient intermediate states that do not influence the final ciphertext, rather than intrinsic error resilience of the algorithm.

Overall, the AES workload highlights that MCU effects in cryptographic applications not only threaten functional correctness but also carry broader security implications. While the majority of observable outcomes were immediate and detectable (e.g., corrupted ciphertexts, halted execution, or processor exceptions), the near-absence of silent data corruptions does not guarantee safety. Although AES exhibited a high proportion of non-propagating faults under MCU injection, even a very small fraction of unmasked errors can have severe security implications. Transient memory corruptions in SDRAM, such as those observed in data store errors, may alter intermediate cryptographic states, including the round key or state matrix, leading to faulty ciphertexts. Such deviations enable fault-based cryptanalysis, notably differential fault analysis (DFA). As demonstrated by Piret and Quisquater [57], a single-byte fault injected during the final or penultimate round of AES-128 allows recovery of the full 128-bit secret key with only two faulted encryptions, reducing the effective security from 2^{128} to approximately 2^{32} operations. This

confirms that MCU-induced faults in external memory are not merely reliability concerns but constitute actionable vulnerabilities in security-critical applications. These results reinforce that MCU-aware fault modeling is essential when evaluating cryptographic processors like LEON3, as SEU-only injection campaigns significantly underestimate the risks posed by multi-bit upsets. Furthermore, they emphasize the necessity of combining robust multi-bit ECC mechanisms with architectural protections, such as temporal redundancy or instruction-level integrity checks, to ensure both reliability and security in MCU-prone environments.

Table 7 summarizes the observed fault categories, including:

- data store errors (~12% of total faults; ~93% in Series 1);
- masked faults (~88% of total faults; 100% in Series 2);
- data access exception (~0.2%);
- silent faults, illegal instructions, memory address not aligned, instruction access faults, and stopped mode (negligible for each type).

Table 7. AES benchmark: classification traps and faults by categories.

Fault Category	Description	Detection Mechanisms	Example Behavior	Observed Frequency
Masked faults	Fault that does not affect program output	N/A	AES output remains correct; fault overwritten or unused memory	Overwhelmingly prevalent (100% of faults in Series 2)
Silent faults	Faults that occur without detection	N/A	Corrupted AES computation may produce incorrect cipher-text without errors	Extremely low (~0.01% of faults in Series 1)
Illegal instructions	Fault triggers in-valid opcode execution, UNIMP or other un-implemented instruction	CPU exception	Processor executes invalid instruction; may halt or trigger fault handler	Rare (only 1 case)
Data access exceptions	Memory access violation, Access error during load or store instruction	CPU exception, MMU	Program halts or triggers exception; AES execution interrupted	Very low (~2% of faults in Series 1)
Data store errors	Incorrect value written to memory, Write buffer error	ECC, parity	AES key or state corrupted; incorrect ciphertext generated	Predominant (~93% of faults in Series 1)
Memory address not aligned	Memory access to un-aligned address	CPU exception	Program halts due to misaligned memory access	Negligible (only 8 cases in Series 1)
Instruction access faults	Fault in instruction fetch, Error during instruction fetch	CPU exception, ECC	AES execution may halt or incorrect instructions executed	Rare (only 1 case)
Stopped mode	Processor halts completely	Watchdog, exception flags	AES execution stops entirely	Rare (only 1 case)

UNIMP: Unimplemented Instruction.

In our earlier study, SEU injections on LEON3 revealed a natural masking rate above 93%, with only about 5.19% of injected faults resulting in processor halt states [13]. The present results indicate that MCUs create a distinctly harsher fault profile: although the raw masking rate under MCUs (88.12%) appears comparable, it primarily reflects injections into non-critical memory regions, unlike the intrinsic masking observed under SEUs. Consequently, MCU-induced faults that do affect active data are far more likely to produce observable errors, revealing a harsher effective fault profile despite the high overall non-propagation rate. These results emphasize that evaluations limited to SEUs may underestimate the true vulnerability of cryptographic applications, underscoring the necessity of MCU-aware fault models when assessing the robustness of secure embedded processors.

4.4. Contextual Comparison with Prior SEU-Centric Studies

While a direct comparison under identical memory targets (i.e., SEU vs. MCU both in external SDRAM) was not feasible, as no prior work has injected SEUs into LEON3's external SDRAM (see Table 1), our results can be contextualized against the closest available baselines: multiple SEU studies targeting internal SRAM structures under the same LEON3 configuration.

Abbasitabar et al. [8] and Mansour & Velazco [9] reported high masking rates in caches and registers due to architectural isolation. Our own SEU campaign [13] observed >93% masking in on-chip SRAM, with only ~7% of injections causing observable errors. Travessini et al. [20] found that partial TMR could achieve >99% tolerance in CPU registers, confirming strong inherent SEU resilience in internal units.

In stark contrast, our MCU injections in external SDRAM, a resource without ECC or redundancy, yield dramatically higher observable error rates:

- <0.01% masking in MulMatrix (vs. >0.0099% in SRAM [13]),
- ~2% masking in FFT (vs. low detection ~0.259% in SRAM [13]),
- 88% non-propagating faults in AES, but only because injections occurred in inactive memory, faults in active regions caused security-critical exceptions absent in SEU models.

Furthermore, MCUs introduced new fault classes not observed under SEUs, such as alignment traps in FFT (from corrupted address registers), reflecting the spatial correlation of multi-bit upsets. Given that external SDRAM lacks the protections of internal SRAM, even a hypothetical SEU injection in SDRAM would show also worse resilience similar to Ref. [13]. The fact that microcontrollers already cause almost total activation of faults confirms that multi-bit disturbances are a significant threat, analogous to single-bit errors in unhardened external memory. Table 8 summarizes this contextual comparison.

Table 8. Contextual comparison of MCU (this work) vs. SEU study in LEON3 [13].

Workload	SEU in SRAM (Our Prior Work)	MCU in SDRAM (Our Work)
MulMatrix	>0.0099% masking	<0.01% masking
FFT	~0.259% masking	~2% masking; alignment traps emerge
AES	93.34% masking	88% non-propagating (inactive memory), but active faults enable DFA

4.5. Comparative Analysis: MulMatrix, AES and FFT

Figure 4 presents a comparative visualization of the fault classification results for three benchmark applications: MulMatrix, FFT and AES-128 encryption. The bar chart summarizes the distribution of observed fault outcomes, offering an intuitive comparison of each benchmark's vulnerability profile.

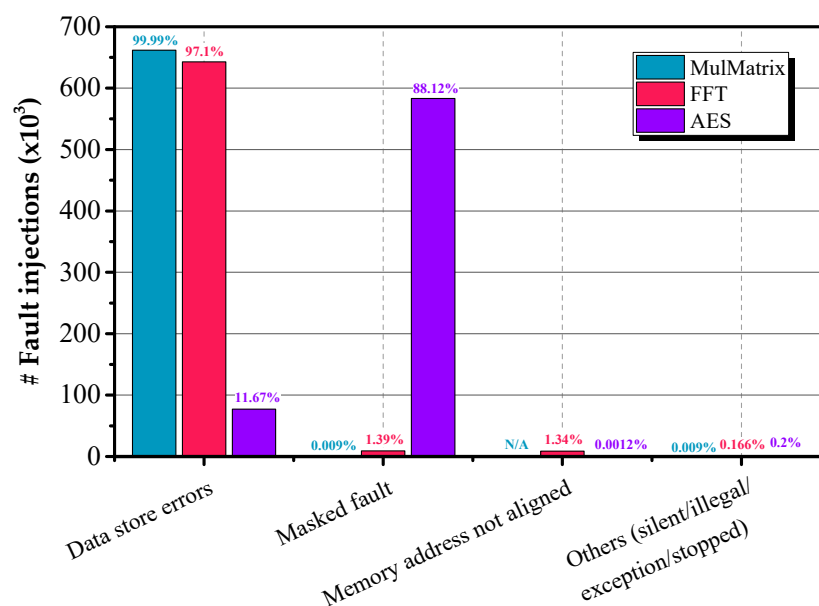


Figure 4. Benchmark-wise distribution of MCU fault outcomes: MulMatrix, AES, and FFT benchmarks.

The experimental outcomes provide three main insights. First, the matrix multiplication benchmark on LEON3 was overwhelmingly dominated by data store errors (~99%), with halted states and masked events occurring only marginally. This indicates that memory-intensive numerical workloads are highly susceptible to direct corruption of stored data, offering minimal inherent resilience or masking. Second, the AES workload exhibited a markedly different fault distribution: approximately 88% of injected faults did not lead to observable errors. However, this apparent resilience is largely attributable to injections in non-active memory regions (e.g., unused SDRAM) or transient intermediate values that do not influence the final ciphertext, rather than intrinsic fault masking by the algorithm itself. Nevertheless, even a small fraction of unmasked errors in cryptographic workloads is critical, since they can compromise system security through fault-based attacks. Finally, the FFT benchmark revealed yet another fault profile: subsequent runs were almost entirely dominated by data store errors (>97%), accompanied by a consistent but small fraction of masked faults (~1–3%). This indicates that FFT, like matrix multiplication, is strongly memory-bound and thus highly sensitive to MCU-induced corruption, though with a slightly higher but still limited degree of masking.

These results emphasize the workload-specific impact of MCUs. Numerical applications such as matrix multiplication and FFT are dominated by catastrophic memory corruption, leaving little natural fault tolerance. In AES, many faults do not affect the output because they occur in inactive memory. However, the absence of true error masking means that even a single fault in critical data or control structures can compromise security. Consistent with earlier SEU-centric studies [8,13,22], our results further show that MCUs amplify reliability threats in memory-intensive computations and security vulnerabilities in cryptographic workloads. Consequently, protection strategies must be tailored to workload characteristics: lightweight redundancy may suffice for numerical workloads, whereas cryptographic and signal-processing workloads demand dedicated multi-bit error detection and correction mechanisms to safeguard both functional correctness and security in radiation-rich or adversarial environments.

5. Conclusions and Future Work

This work presents a comprehensive evaluation of the impact of multiple-cell upsets (MCUs) on the LEON3 soft-core processor, using emulation-based fault injection targeting its SDRAM subsystem during execution of three representative benchmarks: MulMatrix (numerical), FFT (signal processing), and AES (cryptographic). By classifying and analyzing the observed fault manifestations, the study provided new insights into the MCU-induced vulnerability of LEON3 and demonstrated the strong workload dependency of fault outcomes.

The results reveal clear distinctions across workloads. MulMatrix was highly sensitive to memory corruptions, with fault outcomes overwhelmingly dominated by data store errors and limited evidence of masking or recovery. FFT displayed yet another fault profile: early campaigns revealed a mix of data access exceptions, misaligned memory address traps, and silent faults, but as execution stabilized, the workload became dominated by data store errors (>98%) with only a small fraction of masked outcomes. This behavior reflects FFT's iterative, memory-intensive nature, where injection timing and address alignment play a decisive role in determining fault visibility. AES, by contrast, exhibited a higher proportion of non-propagating faults, primarily because injections occurred in inactive memory regions; AES itself provides no intrinsic fault masking due to its avalanche effect. When faults did affect active data or control structures, they typically manifested as store errors or access exceptions, carrying critical security implications.

The contributions of this study are twofold. First, it shifts the focus from traditional SEU- and SET-based analyses toward MCU-specific fault injection, reflecting the growing importance of multi-cell disturbances in nanoscale SDRAM technologies. Second, it introduces a framework that combines FPGA-based emulation with fine-grained fault taxonomy, offering both the realism of hardware campaigns and the diagnostic precision required to assess resilience in embedded processors.

The analysis also points to practical directions for enhancing dependability. The classification of fault manifestations enables the design of selective fault mitigation strategies: lightweight ECC or parity can be prioritized for AES-critical memory regions, algorithm-based fault tolerance (ABFT) techniques may better protect numerical workloads like MulMatrix, while address-alignment checks or improved memory access monitoring could enhance FFT resilience. Furthermore, identifying workload-specific masking patterns provides opportunities to optimize reliability without incurring unnecessary performance or energy overheads.

This work delivers a novel methodology for evaluating MCU-induced effects on embedded processors and provides actionable insights for improving fault tolerance in LEON3-based systems. As future work, the fault model will be extended to include MBUs, representing a more realistic failure mode in advanced technology nodes. Expanding the benchmark suite to encompass broader application classes will further enable the exploration of adaptive resilience mechanisms, where protection strategies dynamically adjust to workload profiles. Such approaches hold strong potential to significantly enhance the reliability and efficiency of space- and safety-critical embedded systems.

While this study was conducted on a Xilinx Virtex-5 FPGA and a specific LEON3 configuration, representative of legacy space-grade systems, the observed workload-dependent vulnerability patterns are expected to scale to newer FPGA generations. Modern FPGAs (e.g., Xilinx UltraScale+, Intel Agilex) feature smaller feature sizes (<16 nm), higher integration density, and 3D stacking, all of which increase susceptibility to spatially correlated MCUs.

Author Contributions: Conceptualization, A.K. and S.S.; methodology, A.K. and S.S.; software, A.K.; validation, A.K., S.S. and H.G.; formal analysis, A.K. and S.S.; investigation, S.S. and H.G.; resources, A.K. and S.S.; data curation, A.K. and S.S.; writing—original draft preparation, A.K. and S.S.; writing—review and editing, A.K., S.S. and H.G.; visualization, A.K. and S.S.; supervision, S.S. and H.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Sehmi Saad was employed by the company RF2S Spectrum Solutions. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript (alphabetic order):

3D	Three-Dimensional
ABFT	Algorithm-Based Fault Tolerance
AES	Advanced Encryption Standard
AHB	Advanced High-Speed Bus
AMBA	Advanced Microcontroller Bus Architecture
ASIC	Application-Specific Integrated Circuit
ARM	Advanced RISC Machines
BRAM	Block RAM
CPU	Central Processing Unit
CS	Chip-Select
DBI	Dynamic Binary Instrumentation
DED	Double Error Correction
DFA	Differential Fault Analysis
DFI	Direct Fault Injection
DFIA	Differential Fault Intensity Analysis
DRAM	Dynamic Random-Access Memory
DUT	Design Under Test
ECC	Error Correction Code
EXE	Executable File
FI	Fault Injection
FIC	Fault Injection Controller
FPGA	Field-Programmable Gate Array
GCC	GNI Compiler Collection
HDL	Hardware Description Language
I/O	Input/Output
ICAP	Internal Configuration Access Port
IP	Intellectual Propriety
IRQ	Interrupt ReQuest
IU	Integer Unit
JCM	JTAG Configuration Manager
JTAG	Joint Test Access Group
LSB	Least Significant Bit
MBU	Multiple-Bit Upset
MCTRL	Memory Controller

MCU	Multiple-Cell Upset
MET	Multiple-Event Transient
MMU	Memory Management Unit
N/A	Not Applicable or Not Available
PC	Program Counter
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RTL	Register Level Transfer
S-Box	Substitution Box
SBU	Single Bit Upset
SDRAM	Synchronous Dynamic Random-Access Memory
SEC	Single Error Correction
SEM	Soft Error Mitigation
SEMT	Single-Event Multiple Transient
SET	Single-Event Transient
SEU	Single-Event Upset
SMT	Satisfiability Modulo Theory
SoC	System-on-Chip
SPARC	Scalable Processor Architecture
SPI	Serial Peripheral Interface
SRAM	Synchronous Random-Access Memory
TCL	Tool Command Language
TCM	Tightly Coupled Memories
TMR	Triple Modular Redundancy
TXT	Text
UART	Universal Asynchronous Receiver Transmitter
UNIMP	Unimplemented Instruction
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

References

1. Aranda, L.A.; Ruano, O.; Garcia-Herrero, F.; Maestro, J.A. Reliability analysis of asic designs with xilinx sram-based fpgas. *IEEE Access* **2021**, *9*, 140676–140685. [\[CrossRef\]](#)
2. Shukla, S.; Ray, K.C. A low-overhead reconfigurable RISC-V quad-core processor architecture for fault-tolerant applications. *IEEE Access* **2022**, *10*, 44136–44146. [\[CrossRef\]](#)
3. Hamada, E.; Igarashi, Y.; Ueno, K. A Hybrid Scrubber Based on the SEM and the PicoBlaze for Artix-7 FPGAs in the COMET Read-out Electronics. *IEEE Trans. Nucl. Sci.* **2024**, *18*, 240–248. [\[CrossRef\]](#)
4. Siecha, R.T.; Alemu, G.; Prinzie, J.; Leroux, P. Analysis of the SEU Tolerance of an FPGA-Based Time-to-Digital Converter Using Emulation-Based Fault Injection. *Electronics* **2025**, *14*, 2176. [\[CrossRef\]](#)
5. Lee, D.; Nam, T.; Park, D.; Kim, Y.; Na, J. Enhanced soft error rate estimation technique for aerospace electronics safety design via emulation fault injection. *Appl. Sci.* **2024**, *14*, 1470. [\[CrossRef\]](#)
6. Gruwell, A.; Zabriskie, P.; Wirthlin, M. High-speed FPGA configuration and testing through JTAG. In Proceedings of the IEEE AUTOTESTCON, Anaheim, CA, USA, 12–15 September 2016; pp. 1–8.
7. Ferlini, F.; Viel, F.; Seman, L.O.; Pettenghi, H.; Bezerra, E.A.; Leithardt, V.R.Q. A methodology for accelerating FPGA fault injection campaign using ICAP. *Electronics* **2023**, *12*, 807. [\[CrossRef\]](#)
8. Abbasitabar, H.; Zarandi, H.R.; Salamat, R. Susceptibility analysis of LEON3 embedded processor against multiple event transients and upsets. In Proceedings of the 2012 IEEE 15th International Conference on Computational Science and Engineering, Paphos, Cyprus, 5–7 December 2012; pp. 548–553.
9. Mansour, W.; Velazco, R. SEU fault-injection in VHDL-based processors: A case study. *J. Electron. Test.* **2013**, *29*, 87–94. [\[CrossRef\]](#)
10. Da Silva, A.; Sanchez, S. LEON3 ViP: A virtual platform with fault injection capabilities. In Proceedings of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, Lille, France, 1–3 September 2010; pp. 813–816.
11. Lindoso, A.; Entrena, L.; García-Valderas, M.; Parra, L. A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA. *IEEE Trans. Nucl. Sci.* **2017**, *64*, 374–381. [\[CrossRef\]](#)

12. Kchaou, A.; Youssef, W.E.H.; Tourki, R.; Bouesse, F.; Ramos, P.; Velazco, R. A deep analysis of SEU consequences in the internal memory of LEON3 processor. In Proceedings of the 17th Latin-American Test Symposium (LATS), Foz do Iguacu, Brazil, 6–8 April 2016; p. 178.
13. Kchaou, A.; Youssef, W.E.H.; Velazco, R.; Tourki, R. An exhaustive analysis of SEU effects in the SRAM memory of soft processor. *J. Eng. Sci. Technol.* **2018**, *13*, 58–68.
14. Sari, A.; Psarakis, M. A fault injection platform for the analysis of soft error effects in FPGA soft processors. In Proceedings of the IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Kosice, Slovakia, 20–22 April 2016; pp. 1–6.
15. Sari, A.; Psarakis, M. A flexible fault injection platform for the analysis of the symptoms of soft errors in FPGA soft processors. *J. Circuits Syst. Comput.* **2017**, *26*, 1740009. [[CrossRef](#)]
16. Yuce, B.; Ghalaty, N.F.; Schaumont, P. Improving fault attacks on embedded software using RISC pipeline characterization. In Proceedings of the IEEE Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), Saint-Malo, France, 13 September 2015; pp. 97–108.
17. Chekmarev, S.A.; Khanov, V.K.; Antamoshkin, O.A. Modification of fault injection method via on-chip debugging for processor cores of systems-on-chip. In Proceedings of the IEEE International Siberian Conference on Control and Communications (SibCon), Omsk, Russia, 21–23 May 2015; pp. 1–4.
18. Chekmarev, S.A.; Khanov, V.K. Fault injection via on-chip debugging in the internal memory of systems-on-chip processor. *IOP Conf. Ser. Mater. Sci. Eng.* **2015**, *94*, 012020. [[CrossRef](#)]
19. Nethula, S.; Bansal, V.; Hamad, G.B.; Ait Mohamed, O. Layout-based Vulnerability Analysis of LEON3 Processor to Single Event Multiple Transients using Satisfiability Modulo Theories. In Proceedings of the 23rd International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 9–10 April 2022; pp. 1–6.
20. Travessini, R.; Villa, P.R.; Vargas, F.L.; Bezerra, E.A. Processor core profiling for SEU effect analysis. In Proceedings of the IEEE 19th Latin-American Test Symposium (LATS), Sao Paulo, Brazil, 12–14 March 2018; pp. 1–6.
21. Mannos, T.J.; Dziki, B.; Sharif, M. Fault testing a synthesizable embedded processor at gate level using ultrascale FPGA emulation. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, USA, 24–26 February 2019; p. 116.
22. Kchaou, A.; Saad, S.; Garrab, H.; Machhout, M. Reliability of LEON3 Processor's Program Counter Against SEU, MBU, and SET Fault Injection. *Cryptography* **2025**, *9*, 54. [[CrossRef](#)]
23. Tsiligiannis, G.; Dilillo, L.; Bosio, A.; Girard, P.; Pravossoudovitch, S.; Todri, A.; Virazel, A.; Puchner, H.; Frost, C.; Wrobel, F.; et al. Multiple cell upset classification in commercial SRAMs. *IEEE Trans. Nucl. Sci.* **2014**, *61*, 1747–1754. [[CrossRef](#)]
24. Alles, M.L. Radiation hardening by process. In *Extreme Environment Electronics*; CRC Press: Boca Raton, FL, USA, 2013; pp. 287–298.
25. Perez-Celis, A.; Wirthlin, M.J. Statistical method to extract radiation-induced multiple-cell upsets in SRAM-based FPGAs. *IEEE Trans. Nucl. Sci.* **2019**, *67*, 50–56. [[CrossRef](#)]
26. Fibich, C.; Rössler, P.; Tauner, S.; Taucher, H.; Matschnig, M. A netlist-level fault-injection tool for FPGAs. *e & i Elektrotech Informat.* **2015**, *132*, 274–281.
27. Monopoli, M.; Biondi, M.; Nannipieri, P.; Moranti, S.; Fanucci, L. RADSAFiE: A Netlist-Level Fault Injection User Interface Application for FPGA-Based Digital Systems. *IEEE Access* **2025**, *13*, 28809–28823. [[CrossRef](#)]
28. Khatri, A.R.; Hayek, A.; Börcsök, J. Development and Verification of Serial Fault Simulation for FPGA Designs using the Proposed RASP-FIT Tool. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 35–40. [[CrossRef](#)]
29. Khatri, A.R. Validation of Proposed Test Set Reduction Hybrid Compaction Schemes for FPGA-Based Designs. *J. Hunan Univ. Nat. Sci.* **2023**, *50*, 228–235. [[CrossRef](#)]
30. Cai, C.; Fan, X.; Liu, J.; Li, D.; Liu, T.; Ke, L.; Zhao, P.; He, Z. Heavy-ion induced single event upsets in advanced 65 nm radiation hardened FPGAs. *Electronics* **2019**, *8*, 323. [[CrossRef](#)]
31. Viera, R.; Dutertre, J.M.; Silva Lima, R.; Pommies, M.; Bertrand, A. Tampering with the flash memory of microcontrollers: Permanent fault injection via laser illumination during read operations. *J. Cryptogr. Eng.* **2024**, *14*, 207–221. [[CrossRef](#)]
32. Jiang, W. Machine Learning Methods to Detect Voltage Glitch Attacks on IoT/IloT Infrastructures. *Comput. Intell. Neurosci.* **2022**, *1*, 6044071. [[CrossRef](#)]
33. Lopez-Ongil, C.; Garcia-Valderas, M.; Portela-Garcia, M.; Entrena, L. Autonomous fault emulation: A new FPGA-based acceleration system for hardness evaluation. *IEEE Trans. Nucl. Sci.* **2007**, *54*, 252–261. [[CrossRef](#)]
34. Entrena, L.; Garcia-Valderas, M.; Fernandez-Cardenal, R.; Lindoso, A.; Portela, M.; Lopez-Ongil, C. Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection. *IEEE Trans. Comput.* **2010**, *61*, 313–322. [[CrossRef](#)]
35. Alderighi, M.; Casini, F.; D'Angelo, S.; Mancini, M.; Pastore, S.; Sechi, G.R.; Weigand, R. Evaluation of single event upset mitigation schemes for SRAM based FPGAs using the FLIPPER fault injection platform. In Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007), Rome, Italy, 26–28 September 2007; pp. 105–113.

36. Aranda, L.A.; Sánchez-Macián, A.; Maestro, J.A. ACME: A tool to improve configuration memory fault injection in SRAM-based FPGAs. *IEEE Access* **2019**, *7*, 128153–128161. [[CrossRef](#)]
37. Aranda, L.A.; Ruano, O.; Garcia-Herrero, F.; Maestro, J.A. ACME-2: Improving the extraction of essential bits in Xilinx SRAM-based FPGAs. *IEEE Trans. Circuits Syst. II Express Br.* **2021**, *69*, 1577–1581. [[CrossRef](#)]
38. The LogiCORE™ IP Soft Error Mitigation. In *Product Guide (PG036)*; Xilinx: San Jose, CA, USA, 2015.
39. Ebrahimi, M.; Mohammadi, A.; Ejlali, A.; Miremadi, S.G. A fast, flexible, and easy-to-develop FPGA-based fault injection technique. *Microelectron. Reliab.* **2014**, *54*, 1000–1008. [[CrossRef](#)]
40. Farooq, U.; Mehrez, H. Pre-silicon verification using multi-FPGA platforms: A review. *J. Electron. Test.* **2021**, *37*, 7–24. [[CrossRef](#)]
41. Ruano, Ó.; García-Herrero, F.; Aranda, L.A.; Sánchez-Macián, A.; Rodriguez, L.; Maestro, J.A. Fault injection emulation for systems in fpgas: Tools, techniques and methodology, a tutorial. *Sensors* **2021**, *21*, 1392. [[CrossRef](#)]
42. Hubert, G.; Velazco, R.; Federico, C.; Cheminet, A.; Silva-Cardenas, C.; Caldas, L.V.E.; Pancher, F.; Lacoste, V.; Palumbo, F.; Mansour, W.; et al. Continuous High-Altitude Measurements of Cosmic Ray Neutrons and SEU/MCU at Various Locations: Correlation and Analyses Based-On MUSCA SEP³. *IEEE Trans. Nucl. Sci.* **2013**, *60*, 2418–2426. [[CrossRef](#)]
43. Franco, F.J.; Fabero, J.C.; Mecha, H.; Rezaei, M.; Hubert, G.; Clemente, J.A. Best-Fit Techniques to Estimate SBU/MCU Cross Sections From Radiation-Ground Tests in Memories. *IEEE Trans. Nucl. Sci.* **2025**, *72*, 1403–1411. [[CrossRef](#)]
44. Kasap, S.; Wächter, E.W.; Zhai, X.; Ehsan, S.; McDonald-Maier, K. Survey of soft error mitigation techniques applied to LEON3 soft processors on SRAM-based FPGAs. *IEEE Access* **2020**, *8*, 28646–28658. [[CrossRef](#)]
45. Wirthlin, M.J.; Keller, A.M.; McCloskey, C.; Ridd, P.; Lee, D.; Draper, J. SEU mitigation and validation of the LEON3 soft processor using triple modular redundancy for space processing. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey CA, USA, 21–23 February 2016; pp. 205–214.
46. Keller, A.M.; Wirthlin, M.J. Benefits of complementary SEU mitigation for the LEON3 soft processor on SRAM-based FPGAs. *IEEE Trans. Nucl. Sci.* **2016**, *1*, 519–528. [[CrossRef](#)]
47. Learn, M.W. *Evaluation of the Leon3 Soft-Core Processor Within a Xilinx Radiation-Hardened Field-Programmable Gate Array (No. SAND2012-0454)*; Sandia National Laboratories (SNL): Albuquerque, NM, USA, 2012.
48. Sari, A.; Psarakis, M. Scrubbing-based SEU mitigation approach for systems-on-programmable-chips. In Proceedings of the 2011 International Conference on Field-Programmable Technology, New Delhi, India, 12–14 December 2011; pp. 1–8.
49. Lotfi-Kamran, P.; Sarbazi-Azad, H. Introduction to data prefetching. In *Advances in Computers*; Elsevier: Cambridge, MA, USA, 2022; Volume 125, pp. 1–17.
50. Losa, B.; Parra, P.; Da Silva, A.; Polo, Ó.R.; Tejedor, J.I.G.; Martínez, A.; Sánchez, J.; Sánchez, S.; Guzmán, D. Memory management unit for hardware-assisted dynamic relocation in on-board satellite systems. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *59*, 6923–6939. [[CrossRef](#)]
51. Losa, B.; Parra, P.; Da Silva, A.; Polo, Ó.R.; Tejedor, J.I.G.; Martínez, A.; Sánchez, S.; Guzmán, D. Hardware-assisted virtualization extensions for LEON processors in mixed-criticality systems. *Microprocess. Microsyst.* **2025**, *112*, 105130. [[CrossRef](#)]
52. Mannos, T.; Plusquellic, J.; Dziki, B. *Information Leakage Analysis Using Accelerated Fault Injection Emulation of a RISC-V Microprocessor, No. SAND2020-0503C*; Sandia National Lab. (SNL-NM): Albuquerque, NM, USA, 2020.
53. Vijay, M.; Mittal, R. Algorithm-based fault tolerance: A review. *Microprocess. Microsyst.* **1997**, *21*, 151–161. [[CrossRef](#)]
54. Bosilca, G.; Delmas, R.; Dongarra, J.; Langou, J. Algorithm-based fault tolerance applied to high performance computing. *J. Parallel Distrib. Comput.* **2009**, *69*, 410–416. [[CrossRef](#)]
55. Aeroflex Gaisler. *Ut699 Leon 3ft/Sparctm v8 Microprocessor. Functional Manual*; Aeroflex Gaisler: Gothenburg, Sweden, 2014.
56. Aeroflex Gaisler. *Handling of External Memory Edac Errors in Leon/Grlib Systems*; Application note, Doc. No GRLIB-AN-0004; Aeroflex Gaisler: Gothenburg, Sweden, 2017.
57. Piret, G.; Quisquater, J.J. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Cologne, Germany, 8–10 September 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 77–88.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.