Avilés, Pablo M., et al. Evaluating reliability through soft error triggered exceptions at ARM Cortex-A9 microprocessor.
In: *Microelectronics reliability, vol. 126, 114323 (Special issue: Proceedings of ESREF 2021, 32th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis), November 2021, 6 p.*

DOI: https://doi.org/10.1016/j.microrel.2021.114323

# Evaluating reliability through soft error triggered exceptions at ARM Cortex-A9 microprocessor

Pablo M. Aviles, Almudena Lindoso, Jose A. Belloch, Luis Entrena

*University Carlos III of Madrid, Electronic Technology Department, 28911 Madrid, Spain*

**Abstract**

The ARM Cortex-A9 is a high-end microprocessor present in multiple programmable System-on-Chip that are used in many application fields including high reliability requirements. This work aims to understand better the behaviour of a high-end microprocessor in a harsh environment through the analysis, evaluation, and classification of exceptions. Exceptions triggered by radiation effects may end up in an unresponsive state of the microprocessor which is undesirable in high reliability scenarios. Experimental results of both fault injection and irradiation presented in this work provide an extensive analysis of the exceptions and their relationship with the executed code and microprocessor architecture.

## 1. Introduction

COTS (Components Off The Shelf) have recently raised as a considerable alternative for the design of spacecraft electronics due to their advantages in cost, performance, and miniaturization [1]. However, manufacturers do not provide an extensive portfolio of qualified devices that can assure its reliability under a harsh environment. In fact, they are designed with newer technologies than qualified components. Usually, COTS present smaller feature size and higher frequencies which could compromise their reliability. Radiation effects in this type of components must be studied and assured for the specific parameters of the space missions they will be used in. Not only space applications must assure high reliability, but neutrons can also affect both aeronautic electronics and terrestrial applications.

When COTS behaviour does not meet the required specifications, they can be hardened to increase their reliability. In such a case, with COTS we face the problem of lack of information about the hardware design, limiting our choices. Without the possibility of changing the circuit, hardening usually relies in redundancy. An example of this approach can be found in CNES DMT and DT2 systems intended for the usage of COTS microprocessors in space [2]. The usage of several functional units increases size, cost, and weight. When microprocessors are the circuit under study, we have also the possibility of software hardening [3]. In this case, hardware is not modified but we must assume penalties in performance and memory size. When it is possible, combinations of hardware and software approaches can present a good trade-off between reliability, performance, and size [4]. Errors due to radiation can trigger exceptions that may end in an unresponsive state of the microprocessor (hang). This type of errors is usually controlled by a watchdog protection (external or internal). The recovery of the system requires a reset or even a power cycle. It is critical for space electronics to reduce this type of intervention to the minimum.

Most of the works in the literature are based on redundancy to detect and correct errors but few of them tackle with the exception analysis. In [5], the authors propose a fast and flexible fault injector framework, called OVPSim-FIM that enables to identify errors and exceptions according to different criteria and classifications. Exception's handling is also studied in [6] where Pthreads and Openmp parallel frameworks are evaluated for the dual core cortex A9. We can also find other works that present hardening techniques [7], evaluate operating systems behaviour [8] or explore specific parts of the microprocessors [9-10] but to the best of our knowledge no detailed exception behaviour analysis has been presented before.

Our work aims to understand better the complex microprocessor behaviour and study how radiation effects are related with microprocessor exceptions. We have selected a high-end microprocessor as a case study to carry out our analysis, the dual core

* Corresponding author. alindoso@ing.uc3m.es
Tel: +34916248888

ARM cortex A9 [11]. To evaluate the system behaviour, we have conducted fault injection and irradiation campaigns. In our experiments, we have extracted and analysed relevant information of the system state when exceptions are triggered. The results of this work can apport deep knowledge of the microprocessor behaviour in a harsh environment. Moreover, our results can be taken into consideration when hardening strategies are designed for microprocessors.

The remaining of this paper is organized as follows, section 2 explains the architecture under study and section 3 presents the experimental setup and results of the injection and irradiation campaigns, and finally section 4 provides the conclusions of this work.

## 2. ARM Cortex A9 processor

ARM cortex A9 is a high-end microprocessor whose key features are:

1) Multicore architecture (up to 4 cores). In this work we will focus on dual core architecture.
2) Superscalar Out-of-order speculative execution, with 8-stage pipeline. Including dynamic branch prediction.
3) NEON SIMD (Single Instruction Multiple Data) coprocessor that is intended for parallelization of instruction execution. With a NEON instruction, the same instruction can be performed simultaneously over several data.
4) High performance VFPv3 floating point unit.
5) L1 Cache per core, providing instruction and data support. Shared L2 cache memory for both cores.
6) Harvard L1 memory controlled by a Memory Management Unit (MMU).

As any microprocessor, the Cortex A9 uses exceptions to handle events [12]. When an exception occurs, the microprocessor must save its state, and execute the corresponding exception handler. All exceptions are handled in a privileged execution mode. While the microprocessor is in those privileged modes, some standard registers are replaced with specific registers related to the exception. In those are included, the Stack Pointer (SP) and Link Register (LR). When the exception appears, the link register associated to the processor mode stores the address of the next instruction in execution (return address). For ARM architecture, exceptions are vectorized at pre-defined memory locations (exception vectors). Table 1 summarizes the types of exceptions that are supported by ARM architecture. Table 1 also shows the relationship between the exception type and the processor mode. ARM microprocessors have 2 additional processor modes apart from the list of Table 1, user mode and system mode. User mode is the only non-privileged mode and is the regular mode for applications execution.

Table 1: ARM exceptions

| Exception type | Processor Mode |
|---|---|
| Reset | Supervisor |
| Software interrupt | Supervisor |
| Interrupt | IRQ |
| Fast interrupt | FIQ |
| Undefined instruction | Undefined |
| Prefetch Abort | Abort |
| Data Abort | Abort |

In this work, we explore the behaviour of the microprocessor for exception modes that are not related with interrupts, excluding from the analysis Interrupts and Fast Interrupts. Our analysis is centred in: Undefined Instruction, Prefetch Abort and Data Abort.

Undefined Instruction exception is triggered when the system tries to execute an unknown instruction (not recognised instruction encoding). It could be related with an error in the instruction affecting the encoding that identifies the instruction or with the accessed address. If the error changes the memory address of the instruction to read, the architecture can reach a memory area in which there are no instructions, causing this type of exception.

Invalid access to data or addresses generates abort exceptions. Abort can happen internally in the MMU or L1 and L2 cache memories, or externally in AXI interface. ARM has two different types of aborts: Prefetch abort and Data abort. Additionally, aborts can be precise (during the execution of an instruction an illegal access takes place and the abort exception is triggered). Imprecise aborts occur during the execution of instructions that are not attempting an illegal access, they can be subsequent to the one with incorrect access. Prefetch abort is generated when an instruction fetched cannot be executed. Data aborts are related with problems with data and can be related with the MMU (alignment faults, translation faults, access bit faults, domain faults or permission faults).

## 3. Experimental results

The experiments have been carried out with the zynq 7000 APSoC that contains a hard-core ARM

cortex A9 with 2 cores [13]. We have performed fault injection and irradiation campaigns which are described in sections 3.1 and 3.2 respectively.

### *3.1. Fault injection campaign*

For the injection campaign we developed a software injector based in the injector described in [7]. In our experiments, the injector injects faults in the register bank of one of the cores of the microprocessor and no additional external host is required to control the DUT (Device Under Test) and collect the results. In order to collect the results of the experiments, the DUT is connected to a computer through a serial connection. Information about the fault injection and errors is gathered during the experiments for subsequent analysis.

When a fault injection takes place, the system generates a bit flip randomly allocated in one of the cores. The flipped bit, the register of the register bank, and the execution instant are also selected randomly. Moreover, we have modified the injector mechanism to increase the observability of the system within the experiments. The information collected for every detected error includes relevant information related with the application execution and the microprocessor state. This way, we are able to detect when an exception takes place and which type of exception it is. All different events are identified with a specific code which is sent through serial port. The injection campaign duration depends on the execution time of the benchmark, but our injector is able to inject approximately 14,000 faults per hour.

For the injection experiments we have utilized two benchmarks:

- Matrix multiplication benchmark, with matrices of 20x20 elements running in both cores without optimization (-O0).
- AES (Advanced Encryption Standard) encryption algorithm running in both cores without optimization (-O0). In the experiments, AES encrypted a data value of 16 bytes using a key length of 128 bits and 10 iterations.

The utilized benchmarks, execute the same application in both cores and perform a comparison of the results to detect errors. Even though the application code is the same, each core has its own program located in a specific memory part. That means that each core is accessing different memory addresses.

Table 2 reports the summary of the injection campaign for bare metal implementation of Matrix multiplication and AES benchmarks. The last column of Table 2 reports the percentage with respect to the total injected faults.

For each benchmark, more than 20,000 faults were injected and around 9% of the injected faults resulted in errors. For matrix multiplication benchmark 52.47% of the observed errors resulted in exceptions, while for AES benchmark it was the 51.62%.

Table 2
Bare metal benchmarks injection campaign summary

| Matrix multiplication | | |
|---|---|---|
| Type of event | Number of events | Percentage |
| Injected faults | 20129 | 100% |
| Observed errors | 1946 | 9.67% |
| Exceptions | 1021 | 5.07% |
| **AES** | | |
| Type of event | Number of events | Percentage |
| Injected faults | 21326 | 100% |
| Observed errors | 1941 | 9.10% |
| Exceptions | 1002 | 4.70% |

Table 3 summarizes the experimental results of the injection campaign regarding exceptions for both benchmarks including information about the core that triggered the exception. According to the results of both benchmarks, the total number of exceptions is distributed nearly equally between the two cores.

Table 3
Bare metal injection campaign: exceptions

| Matrix Multiplication | | | |
|---|---|---|---|
| Exceptions | Core 0 | Core 1 | Total |
| Data Abort | 210 | 153 | 363 |
| Prefetch Abort | 144 | 143 | 287 |
| Undefined | 147 | 224 | 371 |
| Total | 501 (49.07%) | 520 (50.93 %) | 1021 (100.00%) |
| **AES** | | | |
| Exceptions | Core 0 | Core 1 | Total |
| Data Abort | 270 | 106 | 376 |
| Prefetch Abort | 154 | 163 | 317 |
| Undefined | 141 | 168 | 309 |
| Total | 565 (56.39%) | 437 (43.61%) | 1002 (100.00%) |

Comparing both benchmarks, in the case of matrix multiplication the most common exception is Undefined, while for AES is Data Abort. AES algorithm is more complex and manages more data than matrix multiplication, Table 3 reports that both Data Abort and Prefetch Abort are the most common exceptions for AES benchmark (69.16%) which is in agreement with the computational complexity of the application.

We have analysed the influence of the injected register in triggering an exception. Table 4 shows the registers that present the highest percentage of triggering exceptions.

Results of Table 4 show that the most problematic register is LR (Link Register). It must be noted that LR register injection can be considered equivalent to PC (Program Counter) injection, because this register stores the return address during the injection process. Bit flips in the address of the next instruction to be fetched trigger exceptions in 57.46% and 52.20% for matrix multiplication and AES respectively.

Table 4
Bare metal injection: registers with highest percentage of triggering exceptions

| Matrix multiplication | | | |
|---|---|---|---|
| Exceptions | CPSR | LR | FP |
| Data Abort | 75 | 239 | 12 |
| Prefetch Abort | 89 | 197 | 0 |
| Undefined | 21 | 350 | 0 |
| Total exceptions | 185 | 786 | 12 |
| Total injections | 1280 | 1368 | 1344 |
| Percentage (%) | 14.45 | 57.46 | 0.89 |
| AES | | | |
| Exceptions | CPSR | LR | FP |
| Data Abort | 82 | 270 | 21 |
| Prefetch Abort | 128 | 180 | 9 |
| Undefined | 21 | 284 | 4 |
| Total exceptions | 231 | 734 | 34 |
| Total injections | 1445 | 1406 | 1438 |
| Percentage (%) | 15.99 | 52.20 | 2.36 |

Registers CPSR (Current Program Status Register) and FP (Frame Pointer) trigger exceptions but more moderately. CPSR contains the flags (Negative, Zero, Carry, Overflow, Cumulative) and additional state bits. Injection in this register affects to the current microprocessor state and triggers more frequently Prefetch Abort exception (unsuccessful execution of a fetched instruction). This effect is more remarkable for AES benchmark and it is linked with the highest complexity of the executed code.

We have analysed in detail the behaviour of CPSR register to find a relationship between injected bits and exception triggering. For bare metal, the most critical bits are: B0-B3, B5, B9 and B24. It must be noted that all Undefined exceptions of CPSR were related to B5 injection. In CPSR, B0-B3 contain the microprocessor mode and are relevant for Data Abort and Prefetch exceptions. B5, B9 and B24 are reserved and not in use in the architecture but must contain a logic 0 according to ARM documentation. We have found that from all reserved bits of CPSR only these provoke a high amount of exceptions.

Comparing both benchmarks, we can observe a similar behaviour in the total number of exceptions per register, LR is the most problematic and we can observe the increased number of exceptions in AES benchmark for CPSR register injections. Analysing the exception type per register injection, we can appreciate that all of them behave in the same manner for both benchmarks. For these benchmarks, the most common exception is Undefined for LR, Prefetch Abort for CPSR and Data Abort for FP. Frame Pointer keeps track of the stack in which data is stored, injections in this register generate wrong data access and Data abort exceptions. It can be observed in the results that the number of exceptions related to FP is increased with AES benchmark which matches the benchmark characteristics.

We have performed another injection campaign with an OS-based implementation of both benchmarks (matrix multiplication and AES). We have selected FreeRTOS Operating System. FreeRTOS is a market-leading lightweight open-source real time Operating System that provides robustness with small footprint.

The injection campaign summary is reported in Table 5 for OS-based implementations of matrix multiplication and AES. For both benchmarks the total number of injected faults is around 20,000. Comparing the summary of the campaign for bare metal versions (Table 2) and OS-based versions (Table 5) we can appreciate a slight decrease in the total number of observed errors and exceptions with the utilized OS implementations. However, the percentage of exceptions with respect to the total number of errors has increased to 55.9% for OS-based matrix multiplication and decreased to 49.68% for OS-based AES. In both OS-based benchmarks we can observe the decrease in exceptions, but for AES it is more remarkable.

Table 5
OS-based injection campaign summary

| OS-Based Matrix multiplication | | |
|---|---|---|
| Type of event | Number of events | Percentage |
| Injected faults | 20165 | 100% |
| Observed errors | 1599 | 7.93% |
| Exceptions | 894 | 4.43% |
| **OS-based AES** | | |
| Type of event | Number of events | Percentage |
| Injected faults | 20113 | 100% |
| Observed errors | 1703 | 8.47% |
| Exceptions | 846 | 4.21% |

Table 6
OS-based injection campaign: exceptions

| OS-Based Matrix multiplication | | | |
|---|---|---|---|
| Exceptions | Core 0 | Core 1 | Total |
| Data Abort | 168 | 178 | 347 |
| Prefetch Abort | 171 | 73 | 244 |
| Undefined | 120 | 183 | 303 |
| Total | 459 (51.34%) | 434 (48.55 %) | 894 (100.00%) |
| **OS-Based AES** | | | |
| Exceptions | Core 0 | Core 1 | Total |
| Data Abort | 163 | 147 | 310 |
| Prefetch Abort | 88 | 89 | 177 |
| Undefined | 168 | 191 | 359 |
| Total | 419 (49.53%) | 427 (50.47%) | 846 (100.00%) |

Injection campaign detailed results for OS-based implementations are presented in Table 6. For both benchmarks we can notice that the total number of exceptions are distributed equally between the two cores. The exception type distribution in this experiment is similar for both benchmarks: being the most common exceptions Data Abort and Undefined and the least common Prefetch Abort. A different behaviour for both OS implementation is observed when compared with results reported in Table 3. OS-based AES presents a nearly regular distribution of all types of exceptions among the two cores while matrix multiplication presents some irregularities for Prefetch Abort and Undefined. The last one is linked to the benchmark presenting a common behaviour with results reported in Table 3.

For this campaign we have also analysed the exceptions regarding the injected register. We have summarized the results in Table 7.

When compared with bare metal results it can be observed the same behaviour for LR, being the register with higher exception occurrence. CPSR also generates a high quantity of exceptions, although for OS implementations the distribution of exception types for CPSR is similar for both benchmarks. CPSR injection triggered exceptions have decreased considerably for OS-based benchmarks while FP exceptions have increased slightly. Analysing CPSR injected bits, we have found that for OS benchmarks the critical bits are only reserved bits (B5, B9 and B24) which justify the decrease in the number of exceptions. Taking into account the exception type, it is relevant that for all OS benchmarks when injecting in CPSR: B24 triggered all Prefetch Abort exceptions and B5 triggered all Undefined exceptions.

Table 7
OS-Based injection campaign: registers with highest percentage of triggering exceptions

| OS-Based Matrix multiplication | | | |
|---|---|---|---|
| Exceptions | CPSR | LR | FP |
| Data Abort | 83 | 227 | 23 |
| Prefetch Abort | 7 | 237 | 0 |
| Undefined | 38 | 265 | 0 |
| Total exceptions | 128 | 729 | 23 |
| Total injections | 1395 | 1282 | 1363 |
| Percentage (%) | 9.18 | 56.86 | 1.69 |
| **OS-Based AES** | | | |
| Exceptions | CPSR | LR | FP |
| Data Abort | 78 | 189 | 18 |
| Prefetch Abort | 11 | 160 | 6 |
| Undefined | 29 | 328 | 2 |
| Total exceptions | 118 | 677 | 26 |
| Total injections | 1389 | 1307 | 1369 |
| Percentage (%) | 8.5 | 51.80 | 1.90 |

### *3.2. Irradiation campaign*

We have performed an irradiation campaign with protons in CNA (Centro Nacional de Aceleradores) in Seville, Spain. Due to the limited availability of the beam, we performed the irradiation campaign only with matrix multiplication benchmark.

For irradiation campaign, we had to add an external host, that was not exposed to the beam, to control the DUT and collect the results of the experiments. The external host and the DUT were connected through serial communication. The external host does a power cycle when the DUT does not respond a predetermined amount of time (external watchdog). We used an energy of 15 MeV.

Table 8 summarizes the observed events for the irradiation campaign with matrix multiplication bare metal benchmark. For this experiment, the flux was of $8.5 \cdot 10^{8}$ p/cm$^2$/s and the total fluence was of $4.9 \cdot 10^{11}$ p/cm$^2$. Table 8 reports the type of exception (Data Abort, Prefetch Abort and Undefined) triggered per Core. There is an additional category Und. Core that reports exceptions triggered that were collected during the experiments, but the information collected made not possible to identify the core properly. The total number of observed errors were 429, of them 80 were exceptions (18.65%). The total cross-section is $5.6 \cdot 10^{-10}$ cm$^2$.

Table 8
Irradiation campaign results: Matrix multiplication bare metal

| | Core 0 | Core 1 | Und. Core | Total |
|---|---|---|---|---|
| Data Abort | 20 | 21 | 3 | 44 |
| Prefetch Abort | 4 | 1 | 1 | 6 |
| Undefined | 20 | 9 | 1 | 30 |
| Total exceptions | 44 | 31 | 5 | 80 |
| Total observed errors | | | | 429 |

Comparing results for both injection and irradiation we can see a decrease in the percentage of exceptions in irradiation campaign results. It must be noted that the injector is only injecting faults in the register bank, while irradiation affects to the whole system. All exceptions in the irradiation campaign provoked a microprocessor hang, to continue the experiment a power cycle of the DUT was carried out by the external host.

In Table 8 it is shown that the distribution of exception types for both cores is similar, excepting the undefined type. We can appreciate a higher exception occurrence in Core 0 in irradiation. It must be noted that this core is handling the communication with the external host that controls the experiment. It can be observed a different behaviour in irradiation for Prefetch Abort exception, which is much less frequent than in the injection campaign.

We have performed another irradiation experiment with OS-based Matrix multiplication benchmark, results are presented in Table 9. In this case the total fluence was of $5.5 \cdot 10^{11}$ p/cm$^2$ and the total cross-section was of $4.6 \cdot 10^{-10}$ cm$^2$.

Table 9
Irradiation campaign results: OS-based Matrix multiplication

| | Core 0 | Core 1 | Und. Core | Total |
|---|---|---|---|---|
| Data Abort | 18 | 16 | 0 | 34 |
| Prefetch Abort | 3 | 3 | 1 | 7 |
| Undefined | 8 | 7 | 0 | 15 |
| Total exceptions | 29 | 26 | 1 | 56 |
| Total observed errors | | | | 269 |

Table 9 shows similar results to the irradiation results for bare metal implementation (Table 8) but with some peculiarities. Exceptions are also distributed equally between both cores but in this case Undefined exceptions are not larger for Core 0.
The total number of errors, cross-section and total number of exceptions has decreased for OS implementation, which is in agreement with injection campaign results. With this benchmark the percentage of exceptions with respect to the total observed errors has increased slightly (20.82%) in comparison with bare metal irradiation experiment. This behaviour was also observed in the injection campaign, but the increase is more moderate with irradiation.
The distribution of the different exception types is following the same behaviour for both irradiation experiments: Data Abort is the most common exception, followed by Undefined exception and the least probable is Prefetch Abort. If we consider the percentage of one exception type with respect to the total number of exceptions, for OS-based we can appreciate and increase in Data Abort (60.71% for OS-based and 55.00% per bare metal) and in Prefetch Abort (12.50% for OS-based and 7.5% for bare metal) and a considerable decrease of Undefined exceptions for OS-based implementation (26.79% for OS-based implementation and 37,50% for bare metal).

## 4. Conclusions

In this work we analyse how the exceptions triggered by soft errors can affect the reliability of a high-end microprocessor. Experimental results for both fault injection and irradiation campaigns are provided considering two benchmarks with both bare metal and OS-based implementations.

In both injection and irradiation experiments, we can observe a high quantity of Data Abort exceptions and it is more remarkable for irradiation experiments. Irradiation experiments also show a considerable decrease of Undefined exceptions for OS-based benchmark.

In the injection campaign, we analysed the relationship between the injected register and the exception occurrence. For all the experiments, the registers that trigger more exceptions are LR, CPSR and FP. By protecting those 3 registers we can decrease notably the exceptions and improve the system reliability. In bare metal experiments, when the benchmark complexity is increased, CPSR increases considerably the number of triggered exceptions. CPSR bits analysis has shown very critical bits related with exception triggering and some of them vary their criticality when OS is used.

For OS-based experiments the number of total exceptions has been reduced with respect to the equivalent bare metal version, but it has increased the probability of exceptions occurrence with respect to the number of observed errors.

This analysis can be useful for understanding deeply the microprocessor behaviour in a harsh environment and to investigate new hardening strategies to decrease exceptions and system power cycles associated to them.

## Acknowledgements

This work has been supported in part by the Spanish Ministry of Science and Innovation under project PID2019-106455GB-C21 and by the Community of Madrid under project no. 49.520608.9.18.

## References

[1] G. Furano and A. Menicucci, “Roadmap for on-board processing and data handling systems in space,” Dependable Multicore Architectures at Nanoscale, Springer, 2018, pp. 253–281.

[2] M. Pignol, "DMT and DT2: two fault-tolerant architectures developed by CNES for COTS-based spacecraft supercomputers," 12th IEEE International On-Line Testing Symposium (IOLTS'06), 2006.

[3] H. Quinn, Z. Baker, T. Fairbanks, J. L. Tripp and G. Duran, "Software Resilience and the Effectiveness of Software Mitigation in Microcontrollers," in IEEE Transactions on Nuclear Science, vol. 62, no. 6, pp. 2532-2538, Dec. 2015.

[4] Michael Nicolaidis, “Soft Errors in Modern Electronic Systems”, Springer, 2010.

[5] F. Rosa, F. Kastensmidt, R. Reis and L. Ost, "A fast and scalable fault injection framework to evaluate multi/many-core soft error reliability," 2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), 2015, pp. 211-214.

[6] G. S. Rodrigues, F. L. Kastensmidt, R. Reis, F. Rosa and L. Ost, "Analyzing the impact of using pthreads versus OpenMP under fault injection in ARM Cortex-A9 dual-core," 2016 16th European Conference on Radiation and Its Effects on Components and Systems (RADECS), Bremen, 2016, pp. 1-6.

[7] Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, S. Philippe, Y. Morilla, P. Martin-Holgado, “PTM-based hybrid error-detection architecture for ARM microprocessors”, Microelectronics Reliability, Volumes 88–90, 2018, pp 925-930.

[8] G. S. Rodrigues, F. Rosa, Á. B. de Oliveira, F. L. Kastensmidt, L. Ost and R. Reis, "Analyzing the Impact of Fault-Tolerance Methods in ARM Processors Under Soft Errors Running Linux and Parallelization APIs," in IEEE Transactions on Nuclear Science, vol. 64, no. 8, pp. 2196-2203, Aug. 2017.

[9] L. A. Tambara, P. Rech, E. Chielle, J. Tonfat and F. L. Kastensmidt, "Analyzing the Impact of Radiation-Induced Failures in Programmable SoCs," in IEEE Transactions on Nuclear Science, vol. 63, no. 4, pp. 2217-2224, Aug. 2016.

[10] A. Lindoso, M. Garcia-Valderas, L. Entrena, Analysis of neutron sensitivity and data-flow error detection in ARM microprocessors using NEON SIMD extensions, Microelectronics Reliability, Volumes 100–101, 2019, 113346.

[11] ARM, “Cortex-A9 Technical Reference Manual”, r4p1, 2012.

[12] K.C. Wang “Embedded and Real-Time Operating Systems”, Springer, Cham, 2017.

[13] Xilinx Inc., “Zynq-7000 SoC Data Sheet: Overview”, DS190, 2018.