International Journal of Computational and Electronic Aspects in Engineering
Volume 4, Issue 4, October 2023, pp 100 - 109
https://doi.org/10.26706/ijceae.4.4.20231002

Dependability is Measured with Soft Error Caused Exceptions on the Arm Cortex-A9 CPU

Zaidoon Khalaf Mahmood

University of Kirkuk, Kirkuk, Iraq.

Corresponding Author Email: zaidoon.kh.mahmood@uokirkuk.edu.iq

Abstract - In embedded systems, ARM processors are the market leaders, offering high-performance computation, low power consumption, and low cost. As a result, there is a lot of excitement about its potential in the aerospace business. The Xilinx Zynq-7000 FPGA features dual-core ARM Cortex-A9 hard-core processors. are protected from soft failures thanks to a lockstep technique described in this study. The lockstep is an error detection and righting system that is based on both hardware and software. At the software level, it uses checkpoints and rollback methods. On the hardware level, checker circuits and processor replication are also included. The proposed method demonstrates how to execute the plan successfully in two core processors to improve system stability in radiation environments. The results reveal a link between the size of the application and the checkpoint and rollback strategies in terms of usability overhead. as well as a dependence on the number of checkpoints.

Keywords: ARM, embedded processors reliability, Exceptions, Radiation effects, Fault injection

Article – Peer Reviewed
Received: 20 July 2023
Accepted: 15 September 2023
Published: 12 October 2023

Cite this article: Zaidoon Khalaf Mahmood, “Dependability is Measured with Soft Error Caused Exceptions on the Arm Cortex-A9 CPU”, International Journal of Computational and Electronic Aspects in Engineering, RAME Publishers, vol. 4, issue 4, pp. 100-109,2023.
https://doi.org/10.26706/ijceae.4.4.20231002

1. Introduction

Soft mistakes caused by radiation can impair embedded systems in aeronautical applications. causing people to act in ways that aren't planned Ionizing atoms from the atmosphere can interact with silicon, causing transitory pulses in sensitive locations and bit-flips in sequential circuits, resulting in growing difficulties and crashes. Soft mistakes can also be seen at ground level as a result of neutrons interacting with silicon, causing secondary ionising particles that cause transitory faults. Because soft errors change the values stored in memristors, they have an impact on processors. This flaws could cause the CPU to improperly execute a programme, causing SDC or system delays and failures. Soft mistakes in FPGAs can affect the setup stream of bits, causing changes in the device's logic, function, and effectiveness [1, 2].

SRAM-dependent Field-Programmable Gate arrays are used in high-performance processors with dual-core ARM Cortex-A9 processors like the Xilinx Zynq-7000 and Altera Cyclone-V. (FPGAs). These FPGAs are called All Programmable Systems-on-Chip because they have embedded CPUs and storage (APSoC). DSP elements, and a customizable matrix made up of Customized Processing Elements (CLBs). The addition of a larger integrated CPU to SRAM-based FPGAs boosted their performance dramatically. Due to the millions of SRAM cells required to construct all of the synthetically processing, embedded processing, DSPs, and memory, SRAM-based FPGAs are extremely radiation vulnerable [3].

To ensure the reliability of developed processors, a variety of fault-tolerance technologies are available. software-based methods, Hardware-based methods, and hybrid strategies are the three types. The initial stage is to change the design of the system and include checker components and resilience, including such TMR and error usually verification. They have a huge area above them in most cases.

Soft defects are checked and corrected using software-based solutions, which supply extra instructions and data redundancy to the processor. They frequently have a significant overrun in the execution time. To increase mistake identification and correction while reducing area and time overhead, hybrid approaches combine hardware-based and software-based methods. DCLS (Dual-core Lockstep) is a hybrid error detection and repair system that combines software checkpoints and rollback mechanisms with hardware processor replication and inspection circuits.

A lockstep technique using a hard-core PowerPC CPU was built using the Xilinx Vertex II-Pro target device. The approach was demonstrated to be capable of removing 97 present of CPU register errors. The authors designed a lockstep technique to secure a Xilinx Virtex chip's soft-core Leon2 processor. The technique can detect and rectify 99 percent of mild mistakes recorded in the processor's pipeline registers in 17 to 54 milliseconds, depending on the amount of data. By generating a lockstep, the authors showed that the method has a 300 present area resource overrun and a high fault retrieval rate in a modified 8-bit soft-core PIC16 processor. This research presents an enhanced lockstep technique based on two soft-core Micro Blaze processors integrated on a Xilinx Virtex-5 FPGA.

This quantity of data is affected by time resources varying from 17 to 54 seconds. The approach could discover and correct 99 99% mild defects put in the processor's pipeline entrance. This technology is unique in that it incorporates a fault-tolerant Configuration Engine that detects the malfunctioning core and allows for partial reconfiguration to restore it. To our understanding [4], There has been no work done on ARM processors to implement lockstep in the dual-core Cortex-A9, the most common CPU in APSoCs. Instead, ARM offers processors like the Cortex-R5 that have built-in lockstep and can be tweaked for application stability.

2. Related Work

Because of their efficacy and versatility, Mobile consumer products, such as cell phones, ipads, and computers, have all embraced Arm CPU architectures. Newly , there has been a rise in computing performance [5]. Microarchitecture-level modelling with fault injection CPU models and laser testing on the final actual CPU chip are two well-established ways for measuring a microprocessor's soft error dependability at various stages of the design cycle [6]. Because of their efficiency and adaptability, Arm CPU architectures are frequently utilised in portable consumer products including as mobile phones, iPads, and laptops. Because of Arm's unique ability to adapt the design for a specific application, Arm-based systems are currently beneficial in HPC (High Performance Computational) activities and driverless cars [5].

The usefulness and efficiency of a dual-core lockstep as a fault-tolerance device operating on top of FreeRTOS programmes are investigated in this study. The strategy was based on the Zynq7000 APSoC's dual-core ARM Cortez-A9 CPU [7]. Because of their excellent performance and density, System-on-chips (SoCs) are being used in more and more dependable applications. Furthermore, as the structure of SoC shrinks, it becomes more susceptible to radiation-induced soft mistakes [8]. In the design of computing systems, reliability1assessment has always been an important consideration. As a result of the evaluation's findings, As a result, early and precise reliability assessment is crucial [9]. Improvements are highlighted and guided, sparking redesign cycles. This research proposes SyRA, a framework cross-layer innovative reliability test tool for radiation-induced soft errors in microprocessor-based storage arrays. The target system was modelled using a multi-level hybrid Bayesian model. A set of dependability measures are generated using Bayesian inference [10]. This paper presents a comparison of soft error sensitivity in the ARM Cortex-A9 single core embedded microprocessor, which is frequently used in protection systems and runs embedded software built for a bare metal environment and with an os [11].

In this paper, they use a hierarchical analysis to show how soft errors impact the reliability of a PID controller and its DLC equivalent. Single-bit-flip injections in foundation hardware and time-series data data collection from various abstraction layers (ALs) are carried out on a virtual prototype system based on an ARM Cortex-A9 CPU with a PID controller and accompanying recurrent neural network (RNN) built DLC [12]. DNNs are employed in IoT devices with restricted resources, such as those with low-performance CPUs and memory size footprints. While DNN effectiveness and efficiency are important, trained models that provide long-term durability at a reasonable cost are also important [13]. On parallel systems running the Linux operating system, this study [14] investigates the effectiveness of classic fault-tolerant approaches. The fault tolerance of successive approximation algorithms is investigated using two fault injection and reliability test approaches presented in this paper.

Even if the processed input is erroneous, this type of approximate computing technique has built-in fault tolerance, allowing it to converge to a final correct output [15]. An embedded software library of approaches was created using the ARM Cortex A9 processor found on the Xilinx Zynq-7000 series board. One of the tests [16] involves exposing the decapsulated CPU to laser beams focused on the memory storage region, as well as mimicking register file fault entries. With 15 workloads distributed among 8 production technique nodes, the effects of multi-bit upsets on six key hardware components of an ARM Cortex-A9 CPU based on Gem5 micro architecture simulator are investigated [17]. A mathematical approach for predicting microprocessor fault tolerance under radiation is provided [18]. Improved virtual platform frameworks have made soft error assessment of more genuine multicore systems easier [19].

The trend of scaling transistor size and system complexity continues. Soft system flaws, such as the CPU core, are projected to become a severe reliability concern as a result [20]. Under fault injection, this paper compares successive approximate solutions with conventional methods. Following approximation methodologies are implemented as a calculation loop that comes closer to the final result with each loop iteration [21]. The impacts of parallel processing and redundancies at the thread level are investigated in four TMR software implementations, and an unique application for TMR on Linux-based applications is proposed [22]. the initialization step of the flight computers is handled by boot software during space missions. While not all devices are radiation-tolerant, the CPU module in which it runs is [23]. This paper shows how to detect errors in ARM microprocessors using the trace architecture.

The Instrument and the Program Control-flow and data-flow anomalies are detected using the ARM Core Sight technology's Trace Macro cells [24]. To merge privacy and protection essential software systems on a single hardware platform, trustworthy isolation is needed [25]. On the Simics simulation platform, this study presents an automatic approach for injecting faults into the processor and collecting time series data [26]. For high-performance embedded devices, The ARM processor family is steadily gaining traction as the industry standard. Combining a run-time reconfigurable FPGA and an ARM CPU on a single chip delivers exceptional performance and versatility in this application [27]. A charge controller and readout device was created to read data from a range of specialised wearable sensors while also monitoring the battery voltage of the power source [28].

3. Proposed Method

A typical DCLS system includes two processors that are comparable and perform the same application at the same time. Because both CPUs share the same resources and run symmetrically replicated software, they should perform the identical activities, enabling for monitoring system, addressing, and bus management. On a regular basis, a checker module examines the data outputs of both processors, validates the data, and determines the next actions. Every disparity in the data outputs reveals the presence of mistakes, which are then dealt with via a recovery technique. If the data outputs are identical, the processors are presumed to be error-free, and the system's true continuous state is recorded. If an error occurs, the most recent saved proper state is recovered.

The checkpoint procedure is used to save the system's current state. It saves the context of the processors in a secure and error-free storage system. All data resources employed in execution of a program are considered context sensitive data. It's critical to establish the context of the processors so that the system can be recovered properly when it's needed.

Both CPUs have an external DDR memory that acts as both the processor's instructions memory and an alternate secure storage location for the system's checkpoint. Soft faults can occur in both BRAM and DDR storage (bit-flips, also known as SEU). Despite the fact that each has EDAC capabilities, we are considering recreating the processors' constant state on DDR memory to improve system reliability. As a result, we have a variety of options for storing the processors' consistent states. Because DDR memory access times are typically slower than BRAM memory access times, the amount of stores in the DDR should be carefully considered so that system performance is not jeopardised.

The checker component (Checker IP) is a one-of-a-kind HDL IP created in the Zynq's PL. The checker has easy accessibility to both BRAMs and monitors the activities of the two ARM CPUs. The Checker examines the CPUs' data outputs by accessing both BRAM memory and determining whether or not a checkpoint or rollback operation is required. Each ARM CPU receives an interrupt from Checker, notifying them that they must complete a task.

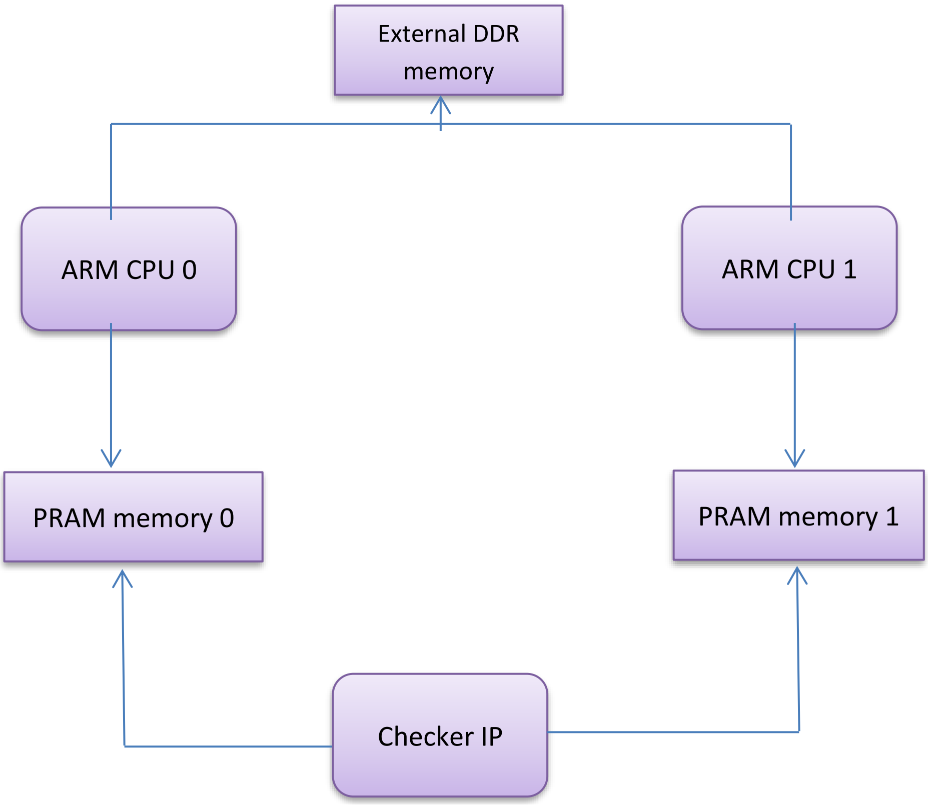

Figure 1. ARM Cortex-A9 dual-core DCLS design proposal

If both CPUs are fault-free, the Checker issues a disruption demand to each of them. Providing processors with access to and preservation of their context. The CPUs conduct the following activities when they process an interruption:

1) The actual thread's performance has been halted.

2) The processor's context (all of the processor's logical registers in our case) is stored to the stack.

3) The procedure for termination is carried out.

The overall (R0 through R12) and specific (SP, LR, and PC) variables, as well as the data RAM used in our technique, are used to assess the processor's environment (all the application data).

When the two most recent iterations compute points that are very close to each other, the algorithm judges the estimate to be "good enough." It indicates that the algorithm has already reached, or is very close to, the best possible outcome.

$$xn + 1 \;=\; xn \;-\; f\,(xn)/f\,(xn) \qquad (1)$$

The trapezoid rule is a strategy for determining the integral of a function. It estimates the areas of various trapezoids that are close in size to the area of the curve. Each trapezoid k has a base length of Δxk = Δx = ba /N if we examine N equally spaced trapezoids specified between point a and point b of the functional. Using the trapezoid technique, Eq. 2 determines the cumulative estimate.

$$\int \, f(x)dx \;\approx\; \Delta x\;/2\sum N\; k = 1\,(f\;(xk-1)\;+\;f\;(xk)) \qquad (2)$$

Simpson's Rule: The Simpson's Rule is yet another method for quantitatively calculating a function fundamental. Unlike the Trapezoid, the Simpson's Rule estimates the area of parabolas rather than trapezoids. In comparison to the Trapezoid rule, in fewer repetitions, this approach calculated the result with greater precision. Equation 2 shows the Simpson estimate for an integration with h = (b- a)/2 step size.

$$\int \, f(x)dx \;\approx h/3[\,f\;(a)\;+\;4f(a+b/2)\;+\;f\;(b)] \qquad (3)$$

These methods are used as benchmarks in this project. Three different variations are proposed for each one. The amount of iterations (and thus accuracy) varies by version, but the technique remains the same.

The fault injection flow employed in this study is depicted in Figure 2. The Platform Setup specifies the fault campaign's parameters, information as the investigated application and the target architecture The Gold Conceptual Framework is then used to extract the functional data appropriately, yielding a reference that is error-free. Another phase that requires human interaction is Fault Injection Setup. As a result of this pressure, A statistical model was used to calculate the number of bit-flips that would be performed. As a result, a list of possible fault injection events and register bits is created.

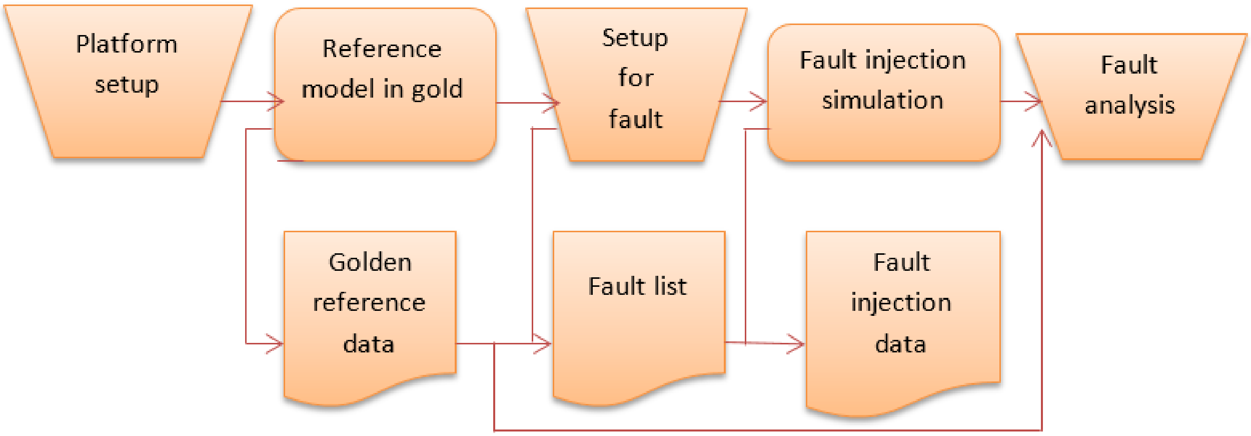

Figure 2: The FI flow covers everything from platform setup to fault analysis

The faults are organised according to a fault categorization that the user might define. The authors of categorise output flaws as accurate output, tolerable faults, substantial faults, and crashes, which is the pattern followed in this study. Accurate outcomes are ones in which the output data (for example, output probability) matches what was obtained by flawless execution (i.e., golden reference data). When it comes to allowed flaws, the output varies from the gold reference points.

They do, however, have a top-tier categorization, which is equivalent to flawless execution. On the other hand, critical defects lead to erroneous probability and no output estimates. Lastly, a failure occurs when an application fails to complete or terminates abnormally with an error message, prompting a pre-emptive removal after a certain amount of time has passed.

3.1 Metrics for Evaluation

Reliability measures must be utilised to adequately analyse the impact of soft error reliability on a given system. The mean work to failure (MWTF) statistic is employed in this investigation. The MWTF also provides the average quantity of effort that an application can perform before failing, in addition to fault classification (i.e., higher values are better). When comparing or analysing the effectiveness of different countermeasures, this is an excellent metric to utilise.

$$MWT\ F\ =\ 1\ /\ (execution\ time\ \times\ AV\ F\ Critical\ Faults\ ) \tag{4}$$

The Architecture Vulnerability Factor (AVF) is a metric for determining the likelihood of a problem causing an error.

4. Experimental Result

The area's and performance's outcomes are as follows:

On a Zynq-7000 ApSoC's dual-core ARM Cortex-A9 processor, the case study system performs bare-metal mathematical calculations. The ARM CPU0 is connected to BRAM storage for distributed applications and an outside DDR memory for programme instructions in the unhardened arrangement. The unhardened code programme also includes three matrix multiplication sequences.

Table 1. The following is a breakdown of our NEON RTL implementation's use

Unit of Function	Primitive FPGA		
	LUT	DSP	BRAM
– arithmetic-ops	18968	263	1
– NEON ALU	520	83	1
– Boolean-ops	388	2	1
– comparison-ops	789	1	1
– shift-ops	762	1	1
– multiply-ops	918	77	1
– miscellaneous-ops	9769	214	1
– NEON Register File	1	1	2
– NEON Unit	12380	263	2

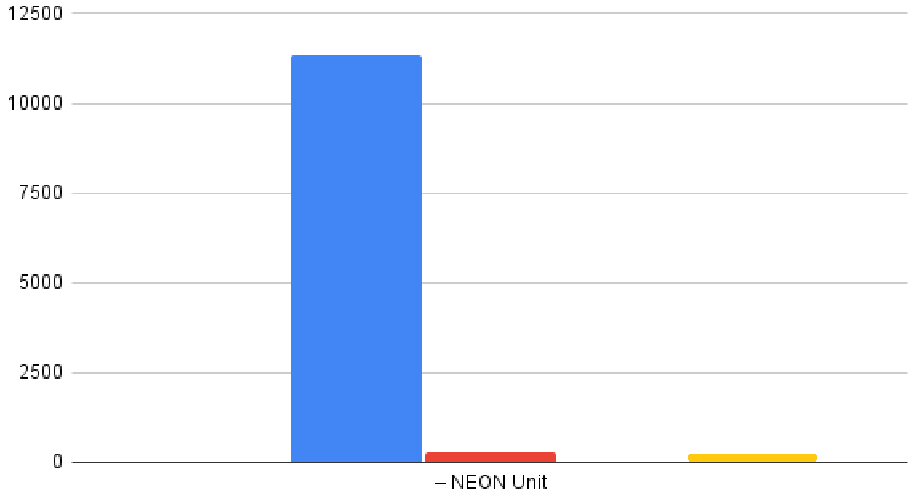

Figure 3. NEON Utilization

It is decided how much NEON ISE will be used by media and protection apps. For portability, the methodologies used in the assessments are not directly ARM/NEON-particular. Rather, we tell the GCC translator to automatically win the application's code. According to a static examination of GCC code, the benchmarks required on median 13% of the NEON ISE total library.

Table 2. The properties of our hardened NEON unit and RTL NEON implementations a in terms of space and latency

Resources for FPGAs				Operation frequency	
HARDENED NEON		NEON RTL		Hardened NEON	NEON RTL
LUT	DSP	LUT	DSP	MHz	MHz
1038	10	1139 (12.7×)	329 (33.2×)	645	167 (2.6×)

Latency Gap: Table 1 summarises the difference in delay among our RTL NEON approach and the protected NEON design.

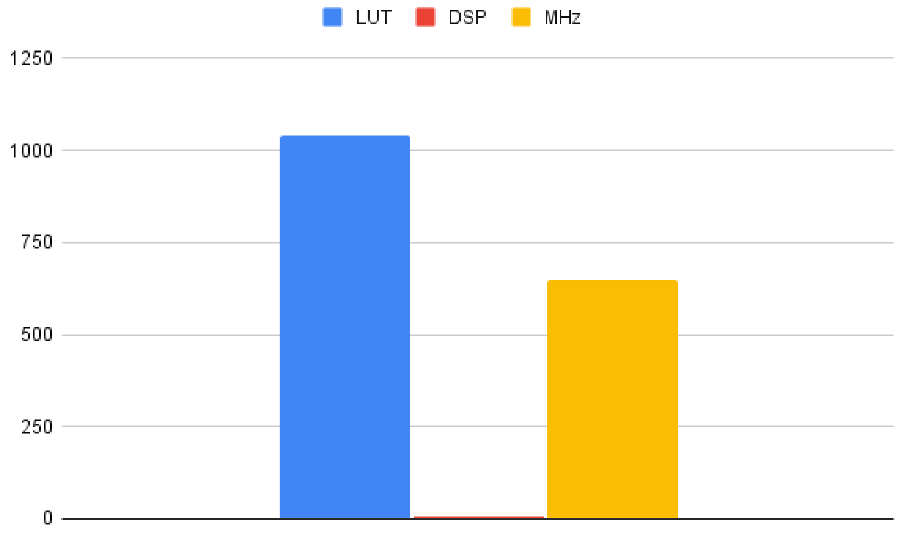

Figure 4. Hardened NEON Gap Analysis CPU Interlay

They measured the distance between other systems in order to define and minimise the effects of this adjustment, this is necessary since we wish to use a CPU interaction instead of the strengthened NEON.

Table 3. The NEON unit and a Dual-Core ARM Cortex-A9 processor are used in this part

Unit of Function	Primitive FPGA Equivalent		
	LUT	DSP	BRAM
This CPU is made up of two ARM Cortex A-9 cores	10040	72	35
ARM Cortex A-9 processor with a single core	5620	35	15
There are two NEON units.	3060	14	6
NEON Unit with a Single	1020	6	2

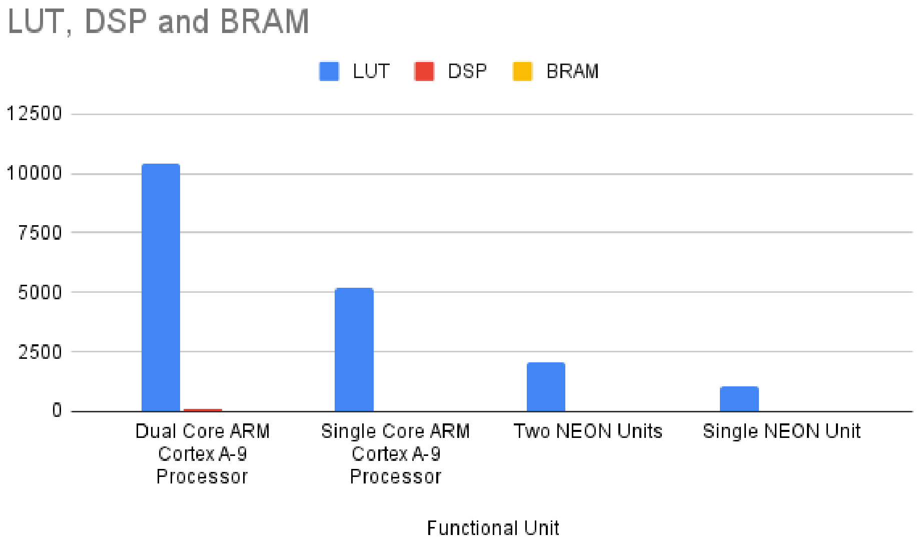

Figure 5. Use of NEON unit and a Dual-Core ARM Cortex-A9 processor

The simplest way for calculating the number of FPGA components that could also fit in the space occupied by the ARM processor is to analyse the Zynq chip's configuration using Xilinx EDA tools.

Table 4. Outcome of proton radiation experiments.

Type of error	#Errors	%Errors
Hang	6	2.60%
Comm	2	0.90%
Det. SW	168	71.20%
Det Hang	60	24.42%
Total	236	100.00%

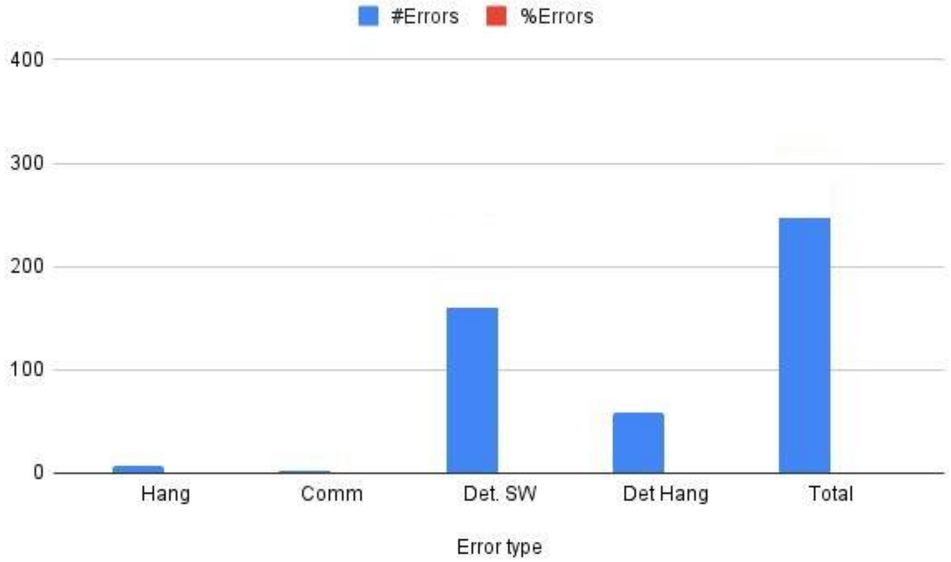

Figure 6. Outcome of proton radiation experiments

After running the application software to estimate its completion time, the maximum distance for randomly selected inserted immediate generation is calculated. The inserted bit and location are also chosen at random. To ensure that estimated values are provided with bias, unique seeds are developed elsewhere and sent to the machine when it is reactivated.

The ARM processor has a lot of registers. It makes use of cached copies of some integers, with the implementation mode selecting the existing register. Each SIMD and floating-point processor architecture has its unique set of stackable registers. Using our method, faults can be placed into any register. To avoid infusing errors in registers that were not used, fault insertion was limited to the ARM core variables at the input validation view. Because of their functionality, many registrants must be perceived separately.

5. Conclusion

In this research, a DCLS approach for reducing radioactive material interferences in hard-core computing in constrained locations is proposed. The recommended DCLS was built on Xilinx's Zynq-7000 APSoC to safeguard the embedded dual-core ARM Cortex-A9 CPU. As a result, they determined that using the strategy had no effect on the framework that focuses while dealing with huge demands. In a future study, they aim to investigate the trade-off between the number of checkpoints and the amount of time it takes for a mistake to happen. which has a big impact on productivity overhead. We will also create a fault injection mechanism that will inject soft faults into the processor's registers to improve the method's reliability.

Reference

[1] X. Xiong, X. Du, B. Zheng, Z. Chen, W. Jiang, S. He, *et al.*, "*Electronics,* vol. 11, p. 3844, 2022. http://dx.doi.org/10.3390/electronics11233844

[2] A. Ahmed and H. Demirel, "FPAA based on Floating Current Source Analog system design," *Design Engineering,* pp. 10953-10969, 2021. https://doi.org/10.1016/j.fmre.2023.01.006

[3] M. A. Ahmed, M. Z. Khalaf, and D. Hüseyin, "Study of finfet transistor. Critical and literature review in finfet transistor in the active filter," *3 c TIC: cuadernos de desarrollo aplicados a las TIC,* vol. 12, pp. 65-81, 2023. https://doi.org/10.1016/j.micpro.2019.102875

[4] M. Alisawi, A. Al-Dawoodi, Y. M. Wahab, L. Hammood, A. Y. Nawaf, and A. Ghazi, "Developing the Real Estate Rental Sector in Third World Countries Using Blockchain Technology: Iraq as Case Study," in *Blockchain Technologies for Sustainable Development in Smart Cities*, ed: IGI Global, 2022, pp. 87-109. http://dx.doi.org/10.4018/978-1-7998-9274-8.ch006

[5] P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "The impact of soc integration and os deployment on the reliability of arm processors," in *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2021, pp. 223-225. https://doi.org/10.1016/j.microrel.2023.115010

[6] A. Chatzidimitriou, P. Bodmann, G. Papadimitriou, D. Gizopoulos, and P. Rech, "Demystifying soft error assessment strategies on arm cpus: Microarchitectural fault injection vs. neutron beam experiments," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 26-38. http://dx.doi.org/10.1109/DSN.2019.00018

[7] Á. B. de Oliveira, G. S. Rodrigues, and F. L. Kastensmidt, "Analyzing lockstep dual-core ARM cortex-A9 soft error mitigation in FreeRTOS applications," in *Proceedings of the 30th Symposium on Integrated Circuits and Systems Design: Chip on the Sands*, 2017, pp. 84-89. http://dx.doi.org/10.1109/LASCAS.2017.7948063

[8] X. Du, D. Luo, K. Shi, C. He, and S. Liu, "FFI4SoC: a fine-grained fault injection framework for assessing reliability against soft error in SoC," *Journal of Electronic Testing,* vol. 34, pp. 15-25, 2018. https://link.springer.com/article/10.1007/s10836-018-5757-2

[9] A. Chatzidimitriou, M. Kaliorakis, D. Gizopoulos, M. Iacaruso, M. Pipponzi, R. Mariani, *et al.*, "Rt level vs. microarchitecture-level reliability assessment: Case study on arm (r) cortex (r)-a9 cpu," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2017, pp. 117-120. http://dx.doi.org/10.1109/DSN-W.2017.16

[10] A. Vallero, A. Savino, A. Chatzidimitriou, M. Kaliorakis, M. Kooli, M. Riera, *et al.*, "Syra: Early system reliability analysis for cross-layer soft errors resilience in memory arrays of microprocessor systems," *IEEE Transactions on Computers,* vol. 68, pp. 765-783, 2018. http://dx.doi.org/10.1109/TC.2018.2887225

[11] L. G. Casagrande and F. L. Kastensmidt, "Soft error analysis in embedded software developed with & without operating system," in *2016 17th Latin-American Test Symposium (LATS)*, 2016, pp. 147-152. http://dx.doi.org/10.1109/LATW.2016.7483355

[12] T. Liu, Y. Fu, Y. Zhang, and B. Shi, "A Hierarchical Assessment Strategy on Soft Error Propagation in Deep Learning Controller," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, 2021, pp. 878-884. http://dx.doi.org/10.1145/3394885.3431573

[13] G. Abich, J. Gava, R. Garibotti, R. Reis, and L. Ost, "Applying lightweight soft error mitigation techniques to embedded mixed precision deep neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 68, pp. 4772-4782, 2021. http://dx.doi.org/10.1109/TCSI.2021.3097981

[14] 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," *IEEE Transactions on Nuclear Science,* vol. 64, pp. 2196-2203, 2017. http://dx.doi.org/10.1109/TNS.2017.2706519

[15] G. S. Rodrigues, Á. Barros de Oliveira, F. L. Kastensmidt, V. Pouget, and A. Bosio, "Assessing the reliability of successive approximate computing algorithms under fault injection," *Journal of Electronic Testing,* vol. 35, pp. 367-381, 2019. https://link.springer.com/article/10.1007/s10836-019-05806-y

[16] F. R. da Rosa, L. Ost, and R. Reis, *Soft Error Reliability Using Virtual Platforms: Early Evaluation of Multicore Systems*: Springer Nature, 2020. http://dx.doi.org/10.1109/PRIME-LA.2018.8370388

[17] A. Chatzidimitriou, G. Papadimitriou, C. Gavanas, G. Katsoridas, and D. Gizopoulos, "Multi-bit upsets vulnerability analysis of modern microprocessors," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, 2019, pp. 119-130.

[18] A. Serrano-Cases, L. M. Reyneri, Y. Morilla, S. Cuenca-Asensi, and A. Martínez-Álvarez, "Empirical mathematical model of microprocessor sensitivity and early prediction to proton and neutron radiation-induced soft errors," *IEEE Transactions on Nuclear Science,* vol. 67, pp. 1511-1520, 2020. https://doi.org/10.1109/TNS.2020.2993637

[19] F. R. da Rosa, R. Garibotti, L. Ost, and R. Reis, "Using machine learning techniques to evaluate multicore soft error reliability," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 66, pp. 2151-2164, 2019. http://dx.doi.org/10.1109/TCSI.2019.2906155

[20] G. S. Rodrigues and F. L. Kastensmidt, "Evaluating the behavior of successive approximation algorithms under soft errors," in *2017 18th IEEE Latin American Test Symposium (LATS)*, 2017, pp. 1-6. http://dx.doi.org/10.1109/LATW.2017.7906764

[21] B. Wibowo, A. Agrawal, and J. Tuck, "Characterizing the impact of soft errors across microarchitectural structures and implications for predictability," in *2017 IEEE International Symposium on Workload Characterization (IISWC)*, 2017, pp. 250-260. http://dx.doi.org/10.1109/LATW.2017.7906764

[22] G. S. Rodrigues, F. Rosa, F. L. Kastensmidt, R. Reis, and L. Ost, "Investigating parallel TMR approaches and thread disposability in Linux," in *2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2017, pp. 393-396. http://dx.doi.org/10.1109/ICECS.2017.8292013

[23] Ó. R. Polo, J. Sánchez, A. da Silva, P. Parra, A. M. Hellín, A. Carrasco*, et al.*, "Reliability-oriented design of on-board satellite boot software against single event effects," *Journal of Systems Architecture,* vol. 114, p. 101920, 2021. https://doi.org/10.1016/j.sysarc.2020.101920

[24] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. García-Valderas, Y. Morilla, and P. Martín-Holgado, "Online error detection through trace infrastructure in ARM microprocessors," *IEEE Transactions on Nuclear Science,* vol. 66, pp. 1457-1464, 2019. http://dx.doi.org/10.1109/TNS.2019.2921767

[25] Y. Shen and K. Elphinstone, "Microkernel mechanisms for improving the trustworthiness of commodity hardware," in *2015 11th European Dependable Computing Conference (EDCC)*, 2015, pp. 155-166. http://dx.doi.org/10.1109/EDCC.2015.16

[26] Z. Dai, Y. Fu, T. Liu, and H. You, "The Internal Fault Detection of Processor Based on Bayesian Network," in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 2018, pp. 775-778. http://dx.doi.org/10.1109/TASE.2016.2574875

[27] T. Xia, J.-C. Prévotet, and F. Nouvel, "Mini-nova: A lightweight arm-based virtualization microkernel supporting dynamic partial reconfiguration," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, 2015, pp. 71-80. https://www.mdpi.com/2079-9292/6/4/93/htm#

[28] J. F. Martínez-Osuna, F. J. Ocampo-Torres, L. Gutiérrez-Loza, E. Valenzuela, A. Castro, R. Alcaraz*, et al.*, "Coastal buoy data acquisition and telemetry system for monitoring oceanographic and meteorological variables in the Gulf of Mexico," *Measurement,* vol. 183, p. 109841, 2021. https://doi.org/10.1016/j.measurement.2021.109841