Aviles, Pablo M.; Lindoso, Almudena; Belloch, Jose A.; García-Valderas, Mario; Morilla, Yolanda; Entrena, Luis (2022). Radiation Testing of a Multiprocessor Macrosynchronized Lockstep Architecture With FreeRTOS. *IEEE Transactions on Nuclear Science*, 69(3), pp.: 462-469.

DOI: https://doi.org/10.1109/TNS.2021.3129164

Radiation testing of a multiprocessor macro-synchronized lockstep architecture with FreeRTOS

Pablo M. Aviles, Almudena Lindoso, Jose A. Belloch, Mario Garcia-Valderas, Yolanda Morilla, and Luis Entrena

***Abstract*— Nowadays, high performance microprocessors are demanded in many fields, including those with high reliability requirements. Commercial microprocessors present a good trade-off between cost, size and performance, albeit they must be adapted to satisfy the reliability requirements when they are used in harsh environments. This work presents a high-end multiprocessor hardened with a macro synchronized lockstep and additional protections. A commercial dual core ARM cortex A9 has been used as case study and a complete hardened system has been developed. Evaluation of the proposed hardened system has been accomplished with exhaustive fault injection campaigns and proton irradiation. The hardening approach has been accomplished for both baremetal applications and OS-based. The hardened system has demonstrated high reliability in all performed experiments with error coverage up to 99.3% in the irradiation experiments. Experimental irradiation results demonstrate a cross-section reduction of two orders of magnitude.**

***Index Terms*—**Microprocessors, ARM, FreeRTOS, fault tolerance, lockstep, proton, soft error.

I. Introduction

RELIABLE high-end microprocessors are currently required in many fields. Aerospace applications are demanding increased computational capacity that cannot be provided by radhard microprocessors. The problem is not only performance but also, size, weight and power consumption. COTS (Commercial Off The Shelf) microprocessors are a suitable alternative but their reliability must be evaluated. In fact, COTS microprocessors are playing an important role in the New Space era, in which a growing number of actors apart from Space Agencies and Governments are involved. Increased efforts are being taken to study commercial parts to test and improve, if possible, their reliability [1-2].

In a harsh environment, electronic circuits are exposed to radiation that can lead to a misbehavior of the electronics. Usually, the type of errors we can observe are classified into hard errors, or non-recoverable errors, and soft errors. Soft errors are recoverable errors, in this case the electronic circuit is not permanently damaged and the system may be recovered. When microprocessors are considered, there exist different kinds of techniques that can provide protection against soft errors. On the one hand, we can highlight the software-based techniques that do not require architectural modification but imply overheads in memory and execution time [3-4]. Commonly, software techniques utilize duplication techniques [5], signature-based control flow [6] and assertions [7]. On the other hand, hardware-based techniques rely on external hardware modules that monitor the execution of the instructions by means of an available interface, such as a memory buses [8-9] or the trace interface [10]. This last choice has been successfully applied to LEON3 microprocessor [11] and ARM A9 [12]. Other Hardware approaches use watchdog processors to observe the microprocessor behavior [13-14]

High-end COTS microprocessors architecture can be explored to propose reliable solutions that meet the new requirements and provide high computational capability [15]. CNES (French National Centre for Space Studies, *Centre National d'Études Spatiales*) hardening architectures based in redundancy are a good example of it [16]. They propose two different approaches for hardening COTS microprocessors based on: temporal redundancy (DMT, Duplex Multiplexed in Time) or spatial redundancy (DT2, Dual Duplex Tolerant to Transients). Another choice consists in using lockstep microprocessors which contains two cores that are micro synchronized. In this case, both cores are always in the same execution point and in the case a discrepancy is found, an error is detected. Micro synchronization requires specific hardware support, which is not present in most architectures. ARM Cortex-R microprocessors can support this behavior, such as the Cortex-R5 used by Texas Instruments in Hercules microcontroller [17].

In order to explore solutions that are not limited by the usage of a microprocessor with lockstep architectural support, we present in this work a multiprocessor hardened system with self-recovery capabilities. Our approach effectively combines a series of techniques that can detect errors and implement several levels of recovery actions to optimize availability without external intervention. The proposed approach is

This work was 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.

P. M. Aviles, A. Lindoso, J.A. Belloch, M. Garcia and L. Entrena are with the Electronic Technology Department, Universidad Carlos III de Madrid, Leganes, 28911 SPAIN (e-mail: paviles@ing.uc3m.es, alindoso@ing.uc3m.es, jbelloc@ing.uc3m.es, mgvalder@ing.uc3m.es, entrena@ing.uc3m.es).

Y. Morilla is with the Centro Nacional de Aceleradores (CNA), Centro Nacional de Aceleradores, CSIC, JA, Universidad de Seville, E-41092 Seville, Spain, SPAIN (e-mail: ymorilla@us.es).

flexible, providing several levels of rollback capabilities.

As a case study, we have developed a hardened system based on a dual core cortex A9. Cores in this architecture accomplish not only verification and rollback processes but also software reset when needed. Additional reliability actions have been considered at system level such as memory protection, watchdog, program delimitation and special exception handling, providing a flexible and reliable low-cost COTS-based architecture. Thanks to this combination of techniques we achieve high reliability, with an error coverage up to 99.3% and a reduction in cross section of two orders of magnitude.

In addition, we evaluate the impact of using an operating system in comparison to a bare metal implementation of the proposed approach. In particular, we implement our technique with FreeRTOS operating system [18], which is a market-leading light weight Real-Time operating system. The characteristics of this Operating System make it a good candidate for safety critical applications.

This paper is organized as follows. Section II summarizes the related work in this field. Section III describes the proposed hardened system. Section IV describes the experiments that have been performed to validate this approach and presents the experimental results for both fault injection campaign (subsection IV.A) and irradiation campaign (subsection IV.B). Finally, section V summarizes the conclusions of this work.

II. Related work

Redundant execution has been regarded as an effective approach for error detection, either by means of time or space [19-20]. Solutions based on lockstep are popular and used to provide functional correctness. For example, the authors of [21-22] proposed a lockstep approach to protect the soft-core Leon2 processor implemented into a Xilinx Virtex device. The technique detected and corrected 99% of soft errors injected in processor's pipeline registers, with a time overhead ranging from 17% to 54% depending on the amount of data. Dual-core Lockstep is proposed in [23] and in [24] using two soft-core MicroBlaze processors and two hard-core processors, respectively. A non-invasive approach for the implementation of processors embedded in FPGAs, using lockstep in conjunction with checkpoint and rollback recovery is presented in [25]. A modified lockstep scheme that detects and eliminates the internal temporary configuration upsets without interrupting normal functioning is proposed in [26].

We can find works in the literature that try to emulate a lock-step behavior within architectures that are not specifically prepared for it. Specifically, the authors in [27] propose a hypervisor-based replication approach that can be applied to commodity hardware allowing virtual lockstep execution. In [28], the authors utilize spatial redundancy approaches with micro synchronization for PowerPC. Moreover, some works use external IPs for observing the microprocessor cores behavior. This is the case of [29] in which an external IP verifies the state of two cores and is able to trigger interrupts in a multicore system. In [30] a relaxed synchronization is presented to stablish hardening strategy for Cortex-A9. In this case, multiple threads with their own stack are used, but without the support of an Operating System (OS). Additionally, the approach in [30] is hybrid and uses an external IP connected to the microprocessor's trace interface to control the execution flow.

III. Proposed approach

Our hardened system has self-recovery capabilities that are supported with a software-based macro synchronization approach. We use spatial redundancy, so that application execution is running in parallel in several cores but without micro synchronization (i.e., it is not required that the cores execute the same instruction at the same time exactly). After the task or program is completed on all functional cores, the results must be checked to decide if errors have occurred. Our approach performs the verification process directly in software without the utilization of an external verification IP, which differs from [28-29].

As case study we have selected the dual core ARM Cortex-A9 microprocessor of the Xilinx Zynq family [31]. The utilized dual core is a high-end ARM microprocessor [32]. Cortex-A9 is a 32 bits superscalar microprocessor with out of order execution. Among its characteristics, it stands out the 8 stages pipeline, 2 level cache for instructions and data and the floating-point unit and SIMD (Single Instruction Multiple Data) coprocessor.

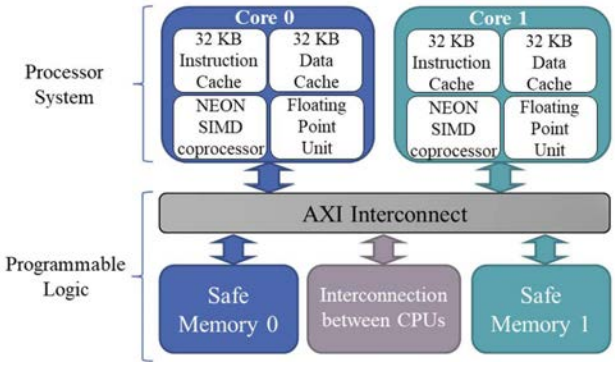

Fig. 1. Architecture of the proposed hardened system.

Fig. 1 shows the architecture of the proposed system where we can appreciate the hard core cortex A9 with its two cores (Core 0 and Core 1) and 3 external modules located in Programmable Logic:

- Safe Memory 0 and 1. These blocks are two external protected memories that store the required information to recover the system. We have used a dedicated memory per core.
- Interconnection block. Every core can use this block to share computation results with all the other cores present in the architecture. In our architecture, we implement this module with Xilinx Mailbox [33]. This type of information interchange is needed to share information between the cores at certain points of the application execution.

It must be noted that the 3 external modules are located in Programmable Logic of the Zynq device, but the hardening strategy could have used other memory resources that are inside the hard core. In such a case, we have to assign specific dedicated memory sections for each of the modules.

In the following subsections are described in detail the hardening approaches that our technique uses.

A. Safe recovery data storage

We implement a secure memory in programmable logic to store microprocessor contexts using the Xilinx IP logic core: Memory Block Generator [34]. This memory supports the built-in Hamming Error Correction Capability (ECC), soft Hamming Error Correction (Soft ECC) for data widths of 64 bits or smaller and Hsiao for data width of 128, 64 and 32 bits. With this capability enabled, memory automatically detects single and double errors and corrects single errors.

In our system, each core has a dedicated safe memory, located in the programmable logic. This memory is used to store the core current context (relevant information that defines the core state) and recover it when needed. In our system, the context contains registers and local variables. Cache or external memories are not included in the context. Our memory blocks have ECC enabled with Hsiao algorithm [35].

B. Verification

To perform macro synchronization, application code is divided in blocks in both cores. After the completion of each block, the verification process checks whether the context of the cores is the same. Additionally, when the output of both cores presents no inconsistencies, verification process stores the context of each core in the corresponding safe memory. Our technique requires access to registers in privileged mode that is not available in regular application execution. In ARM microprocessors, interrupts and exceptions are executed in privileged mode. To overcome this situation, our technique triggers software interrupts to perform verification and rollback with privileged mode.

Moreover, our verification scheme uses a signature as proposed in one of the methods of [29]. At each verification point, a signature of the values to be compared is computed. This way, verification overheads can be reduced. Both cores are interconnected to provide the necessary data to complete the verification process, which differs from other approaches. The work presented in [29] uses an external IP which controls multicore CPUs execution and checks the correctness of the system, all the processes are handled with interrupts triggered by the external IP.

In our case, when a core finishes the execution of one application block, shares the signature and halts execution, waiting for the other core to finish and share the signature. Verification takes place when all cores have finished and shared their signature.

C. Rollback

When errors are detected, the rollback process restores the microprocessor from a previous error free state. In contrast to other approaches, our hardening approach permits consecutive rollbacks with flexible depth. If a rollback process is not able to restore the system correctness, it will be tried another rollback to the previous verification point (consecutive rollback). The number of consecutive rollbacks is flexible and can be stablished according to the application needs. For our experiments described in section IV, we have determined experimentally the maximum number of consecutive rollbacks allowed. As described in verification, rollback requires privileged mode and it is obtained with software interrupts.

D. Watchdog Timer

We use a watchdog timer to control the microprocessor architecture. In the case of a hang, the watchdog will restart the system. The used watchdog is associated to core 0 but due to the macro synchronization technique, it controls the whole system. If core 1 hangs, core 0 will be stuck waiting for data to complete the next verification point. After a predetermined amount of time, watchdog timer will trigger the system restart.

E. Additional protections

Our hardened system is able to detect incorrect behaviors and trigger Software reset if needed. For instance, if the maximum selected rollback depth is reached without success, the system will be restored by triggering a software reset.

Additionally, the proposed system is able to restore the system from exceptions. We have modified the exception table in order to handle all exceptions and be able to recover the system from any possible event. When an exception occurs, we try first to recover the system with rollback. If the maximum number of rollbacks is not successful, software reset is triggered. With this modification, the system also implements a rollback from the exception handlers.

To increase the system capabilities, we have also delimited the memory areas in use by the applications, so illegal access will trigger exceptions.

F. Operating System usage

The hardening mechanism based in macro-synchronized lockstep has been implemented for two different versions: baremetal (without operating system) and with the operating system FreeRTOS [18]. This OS (Operating System) is a light real time operating system that has the main advantage of being able to work with threads and pseudo parallelism in a single core processor. FreeRTOS provides separate stacks for each task. The stacks, and each kernel object in general, are located within a memory space defined as heap. FreeRTOS has five sample implementations of memory allocation. FreeRTOS applications can use one of the sample implementations or provide their own. Our system uses Heap 1, which is recommended for commercially critical and safety critical systems. Critical systems often prohibit dynamic memory allocation due to uncertainties associated with non-determinism, memory fragmentation, and failed allocations, but heap 1 is always deterministic and does not fragment memory.

When OS in in use, we use the same architecture described in Fig. 1 but we apply software changes. The application is run in one thread. The verification process is implemented in a

lower priority task than the main tasks associated with the application code. For OS implementations, verification and rollback processes are also implemented with software interrupts. When a verification point is reached in the main task, the OS will execute the verification process task, and once this task is completed, it will return to the main task.

IV. Experimental Results

For the experiments, we have used a Zybo development board [36], which contains a Zynq 7010 device [10] with a hard-core Cortex-A9 [32]. In the experiments, the DUT (Device Under Test) is connected serially to an external host used to collect the results and observe constantly the system behavior. All the information gathered during the experiments is analyzed when the experiments end.

The experiments have been performed with both versions of the proposed macro synchronized lockstep technique: baremetal and OS-based version (using FreeRTOS). Moreover, we have used two benchmarks:

- Matrix multiplication (Mmult). We used matrices of 20x20 elements and data size of 16 bits.
- AES encryption [37] (AES, Advanced Encryption Standard). AES algorithm encrypted a data value of 16 bytes, the utilized key length was of 128 bits and the algorithm was performed with 10 iterations.

Table I summarizes the memory overhead of the proposed system considering non-hardened and hardened versions of all benchmarks. The reported information demonstrates that the proposed technique does not incur in additional overhead in terms of program and data size. However, it must be pointed out that two additional memories are added to the hard-core microprocessor (safe memory 1 and 2, both of 64KB) and execution is performed simultaneously in two cores. Reported data also shows a slightly larger overhead for core 0, this core can be considered as the primary core because it controls the watchdog and also handles the communication with the external host. Results also show the considerable increase in size of the benchmarks for OS versions.

Table II summarizes the performance overhead required for the hardening technique taking in consideration the utilized benchmarks. The microprocessor frequency is 650MHz. The first row reports the overhead in execution time for the verification and rollback processes. We can observe that for all benchmarks execution time of verification & rollback is slightly increased for OS versions, being Mmult execution time much longer than AES. This occurs because of the amount of data used by the two benchmarks, Mmult is using matrices of 20x20 elements.

The second and third rows of Table II show respectively the medium time and the maximum time to restart the system during irradiation experiments. Part of this time, around 13 ms, are devoted to serial communication data transfer that acknowledges the system restart. We have this information only for the benchmarks that were utilized in the irradiation experiments. The last row shows the execution time of one iteration of each benchmark. It must be noted that the system boot has not been considered for the execution time reported in Table II.

TABLE I
MEMORY OVERHEAD

Benchmark		NH Total size (bits)	Hardened Total size (bits)	Overhead
Mmult Baremetal	Core 0	69,540	72,924	1.05
	Core 1	65,752	66,572	1.01
AES Baremental	Core 0	63,000	68,320	1.08
	Core 1	59,212	61,996	1.05
Mmult OS	Core 0	201,748	206,324	1.02
	Core 1	197,332	199,996	1.01
AES OS	Core 0	197,100	201,684	1.02
	Core 1	192,692	195,420	1.01

TABLE II
PERFORMANCE OVERHEAD

	Mmult Baremetal	AES Baremetal	Mmult OS	AES OS
Verification & Rollback (µs)	207	5	215	13
SW Reset (ms)	637	-	594	-
Tmax (3 Rollback+SW Reset) (ms)	661	-	604	-
Execution Time (ms)	6.9	6.8	7.2	7

In both benchmarks the code is divided in 10 blocks which corresponds with 10 verification points. An additional verification point is added at the beginning of the code, which results in a total of 11 verification points for all utilized benchmarks.

The safe memories that store the context of each core have a size of 64KB. We have selected a maximum consecutive rollback depth of 3. This depth has been determined experimentally and it is described in subsection IV.A.

OS benchmarks use Heap 1. Two tasks are created before starting the scheduler: the higher priority task for the application code, and another lower priority task for verification process. Each task has its own unique stack that is set by the programmer when the task is created. We set a stack depth of 1,400 that means a 5,600 bytes stack.

The proposed approach has been evaluated with injection and irradiation campaigns that are described respectively in subsections IV.A and IV.B.

A. Injection campaign

Firstly, an injection campaign was made to study in detail and adjust the system for a subsequent irradiation campaign.

Our injector is based in [38] and generates a random bit-flip in a random register of the register file. It also selects the core and the injection instant randomly. In these experiments, the external host is a computer that is collecting the results, which will be analyzed when the injection campaign ends.

Table III shows the injection campaign results for each version (baremetal and OS) of both benchmarks, Mmult and AES128.

TABLE III
INJECTION CAMPAIGN RESULTS

	Category	Baremetal		OS	
		Mmult	AES128	Mmult	AES128
Faults	Silent faults	44,092 (80.9%)	42,926 (88.9%)	40,841 (77.0%)	45,909 (90.9%)
	Errors	10,413 (19.1%)	5,339 (11.1%)	12,174 (23.0%)	4,573 (9.1%)
	Total faults injected	54,505 (100.0%)	48,265 (100.0%)	53,015 (100.0%)	50,482 (100.0%)
Errors	Undetected errors	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Detected errors	10,413 (100.0%)	5,339 (100.0%)	12,174 (100.0%)	4,573 (100.0%)
Detected errors	Corrected errors	5,356 (51.4%)	1,754 (32.9%)	5,429 (44.6%)	1,705 (37.3%)
	WDT	1,012 (9.7%)	830 (15.5%)	5,454 (44.8%)	1,722 (37.7%)
	SW Reset	4,045 (38.9%)	2,755 (51.6%)	1,291 (10.6%)	1,146 (25.0%)

Table III has three horizontal sections: Faults, Errors and Detected errors. In each section, data is also provided in percentage regarding the total number of events per benchmark. The Faults section provides a summary of each experiment showing the total number of faults injected and the errors observed. Table III shows that for each benchmark we injected around 50,000 faults and the percentage of observed errors ranged from 9.1% to 23.0%.

Table III has also an Error section which shows a summary of the detection capabilities of the proposed technique. Experimental results reported in Table III demonstrate that we achieved full error coverage for all benchmarks.

Finally, the section of Detected errors of Table III shows an analysis of the detected errors for each benchmark. We have used the following error categories:

- Corrected errors: errors in which system is recovered correctly with one or several rollbacks.
- WDT: Watchdog Timer restarts the system due to an abnormal behavior in execution.
- SW Reset: SW Reset is triggered by the proposed technique when it detects that the system cannot be correctly restored.

Fig. 2 shows the distribution of errors for Baremetal benchmarks in the injection campaign. The reported percentage is with respect to the total number of observed errors per benchmark. Results show a higher quantity of corrected errors for matrix multiplication benchmark, which correlates with the benchmark complexity. With AES benchmark we observe a higher percentage of SW reset, this is due to the decrease of corrected errors that produce SW resets. The WDT percentage is similar for both benchmarks.

Fig. 3 shows the distribution of errors for OS benchmarks in the Injection campaign. We can observe that the correction capabilities have decreased for OS versions but again Mmult benchmark behaves better that AES benchmark. Comparing Fig. 2 and Fig. 3 we can observe that when OS is used, the number of WDT is considerably increased, and it changes the trend shown in Fig. 2 having more relevance this effect for Mmult benchmark. This increase in WDT affects the rest of error categories and we can observe a clear decrease in SW Reset for both benchmarks. It must be pointed out that our baremetal technique has extended recovery capabilities when exceptions are triggered. This implies an increase of corrected errors and a decrease of WDT detected errors.

Baremetal SW Resets are more frequent, 3.6 times higher in percentage than OS version for mmult and 2.1 for AES. In the case of WDT reset, OS version has a percentage 4.6 times higher than Baremetal version for mmult and 2.4 for AES.

In order to characterize the utilized benchmark, we did a preliminar fault injection experiment to decide the maximum number of rollbacks allowed. We had enabled 10 possible consecutive rollbacks with their corresponding storage in both safe memories. This experiment was done with 10,845 and 12,043 injections for Baremetal and OS, respectively. Analyzing the results, we could observe that the system was recovered most of the times with up to 3 rollbacks. Only the 0.15% of the total number of rollbacks for baremetal Mmult benchmark required from 4 to 10 rollbacks. In the case of OS, the percentage was even smaller (0.13%). Taking into account this result, we decided to stablish the limit for the utilized benchmarks in 3 consecutive rollbacks.

TABLE IV
NUMBER OF ROLLBACKS DISTRIBUTION

	1 Rollback		2 Rollback		3 Rollback		Total
	Quantity	%	Quantity	%	Quantity	%	
Mmult Baremetal	4,838	90.33	355	6.63	163	3.04	5,356
AES Baremetal	1,418	80.84	282	16.08	54	3.08	1,754
MMmult OS	5,071	93.41	246	4.53	112	2.06	5,429
AES OS	1,339	78.53	337	19.77	29	1.70	1,705

Table IV reports the number of rollbacks distribution of the corrected errors reported in Table III. Percentages are with respect to the total number of recoveries for each benchmark. We can observe that most recoveries of the system were performed with 1 rollback, around 90% per Mmult and around 80% per AES benchmarks. The smallest percentage of recoveries with one rollback is for the OS version of AES. AES benchmarks are the ones with highest percentage of recoveries with 2 rollbacks, ranging from 16% to 19%. In the case of three rollbacks, figures are similar for baremetal (around 3%) and OS versions (around 2%). The higher complexity of AES benchmark and the usage of signatures seems to increase the required number of rollbacks to accomplish a successful recovery.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2021.3129164, IEEE Transactions on Nuclear Science

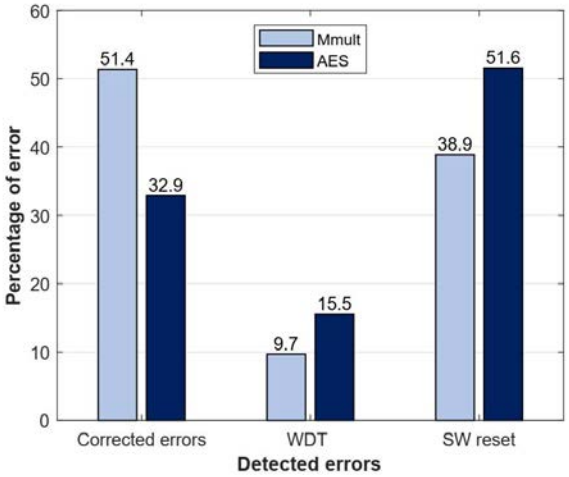

Fig. 2. Distribution of observed errors for Baremetal benchmarks (Injection Campaign).

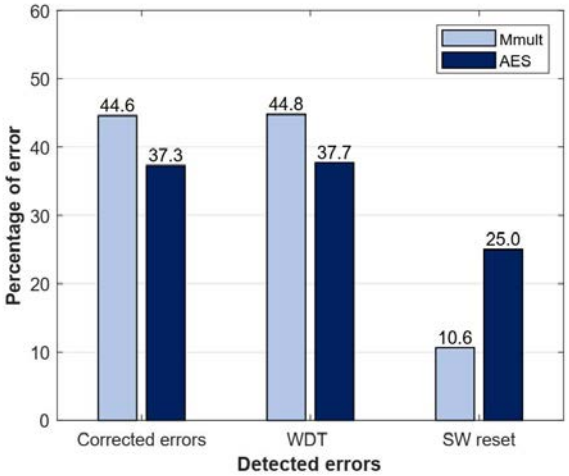

Fig. 3. Distribution of observed errors for OS benchmarks (Injection Campaign).

B. *Irradiation campaign*

The proposed approach was also evaluated in an irradiation campaign that was performed at Centro Nacional de Aceleradores (CNA) in Seville, in January 2021. We have used protons with an energy of 15 MeV.

For the irradiation experiments, an external host was used to control the DUT. This host is connected to the DUT through serial connection and collects the results and information about the system state: data about successful verification points, detected errors and recoveries. In these experiments, the external host can force a power cycle in the case the DUT is not responding and cannot be restarted or recovered by the proposed hardening approach (undetected errors). Fig. 4 shows the experimental setup for the Irradiation experiments.

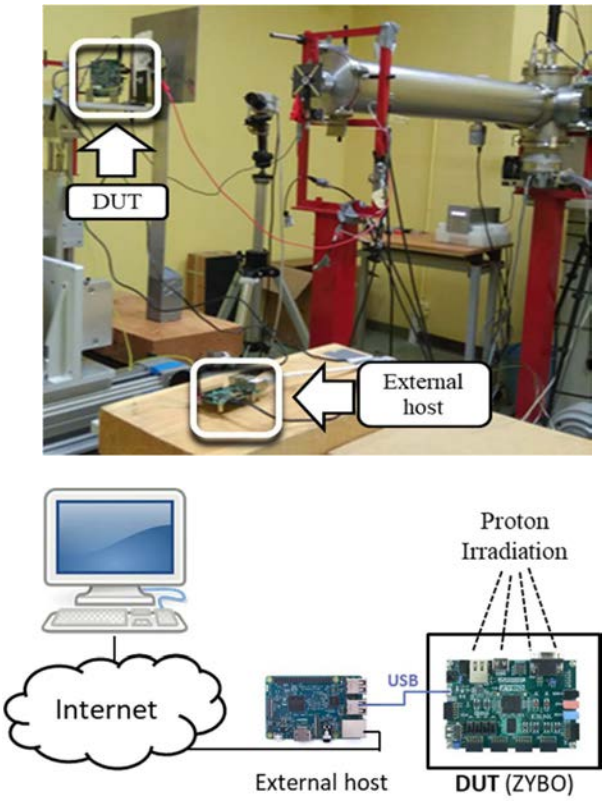

Fig. 4. Experimental setup (Irradiation campaign)

Thanks to the benchmark characterization described in subsection IV.A, we have set a maximum of 3 consecutive rollbacks in all irradiation experiments. This way we can avoid unnecessary delays in the system recovery process. In addition, it allows to reduce the memory space for data redundancy, since only the context of the last 3 verification points is stored.

Experiments were performed with baremetal and OS-based implementations of matrix multiplication benchmark. We have used a fluence of $4.9 \cdot 10^{11}$ p/cm^2 for baremetal version and $5.5 \cdot 10^{11}$ p/cm^2 for OS-based version. The experimental results are shown in Table V, which follows a similar structure to Table III, having three sections: Errors, Detected errors and cross-section. The Errors section reports a summary of the irradiation experiment showing the total errors, detected and undetected errors. Experimental irradiation results demonstrate the high error coverage of the hardened system (up to 99.3%) in accordance with the results of the injection campaign reported in Table III. The following section of Table V, Detected errors, shows an analysis of the detected errors with the same error categories used in Table I. Finally, Table V shows the cross-section for both benchmarks taking into consideration the total number of observed errors (Total errors) and the undetected errors. Cross-section section also reports the confidence interval for all the reported data. Results show an improvement of cross-section up to two orders of magnitude.

In Fig. 5 is shown the cross-section regarding the error categories for both benchmarks.

TABLE V
IRRADIATION CAMPAIGN RESULTS

	Category	Mmult	
		Baremetal	OS
Errors	Undetected errors	2 (0.7%)	5 (2%)
	Detected errors	273 (99.3%)	248 (98%)
	Total errors	275 (100.0%)	253 (100.0%)
Detected errors	Corrected errors	142 (52%)	65 (26.2%)
	WDT	64 (23.4%)	118 (47.6%)
	SW Reset	67 (24.6%)	65 (26.2%)
Cross-section (cm^2)	Total errors	$5.6 \cdot 10^{-10}$ ($4.9 \cdot 10^{-10}$-$6.2 \cdot 10^{-10}$)	$4.6 \cdot 10^{-10}$ ($4.0 \cdot 10^{-10}$-$5.2 \cdot 10^{-10}$)
	Undetected errors	$4.0 \cdot 10^{-12}$ ($4.9 \cdot 10^{-13}$-$1.5 \cdot 10^{-11}$)	$9.1 \cdot 10^{-12}$ ($2.9 \cdot 10^{-12}$-$2.1 \cdot 10^{-11}$)

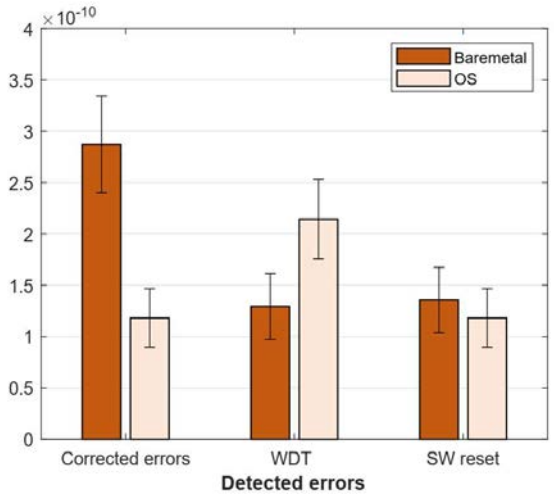

Fig. 5. Cross-section per error category (Irradiation Campaign).

The hardening approach presented in [29] reports an improvement of cross-section of one order of magnitude. Our hardened system achieves a larger improvement in cross-section, being up to two orders of magnitude.

Baremetal results of Table III and Table V are in agreement. The percentage of corrected errors is similar in both experiments, but we can observe a variation for the other two categories of around 10% (with increasing number of WDT Reset and decreased number of SW Reset in radiation experiments).

For baremetal version, 209 rollbacks were made, of which 142 were successful (52%). Of the 142 corrected errors, 134 were with one rollback and 8 with two consecutive rollbacks. In baremetal version, WDT Reset is reduced approximately by a factor of 2 with respect to OS version.

OS version results report a total number of 130 rollbacks, half of which were correctly recovered. Of the 65 corrected errors, 51 were with one rollback and 14 with two consecutive rollbacks. WDT Reset is the most common error for this version, and it is in agreement with injection results. It must be noted that for this benchmark, the usage of 2 rollbacks is effective and contributes in around 21% to the successful recoveries of the system.

Comparing Tables III and V, we can observe a negligible decrease in the error coverage for irradiation experiments (up to 2% smaller). If we analyze the different categories for detected errors in both tables, we can observe that the behavior of baremetal version is similar in percentage for corrected errors category, but slightly increased for WDT reset and decreased for unsuccessful recoveries for irradiation experiments. In the case of OS, results in the injection campaign differ from irradiation results in percentage. Correct recoveries decrease considerably, with the corresponding increase of other ways of restarting the system. This difference in the experimental results could be due to the limitations of the injection technique that only affects to the register file. The complexity of the Operating System could also be related to the increased number of non-recoverable errors for irradiation experiments.

V. CONCLUSIONS

This work presents a hardened high-end multiprocessor system. The proposed hardening approach uses macro synchronization lockstep technique with recovery capabilities and additional system protections. Lockstep is performed at software level without specific hardware support for verification. We have selected a dual core ARM Cortex-A9 as a case study and macro synchronization has been implemented in two different versions: Baremetal and OS-based with FreeRTOS. Fault injection and irradiation campaigns have been carried out to validate the reliability of the system. Experimental results demonstrate the high effectiveness of the proposed approach with full error coverage for injection and up to 99.3% for proton irradiation experiments. The hardened system achieves a reduction in cross-section of two orders of magnitude, which is higher than other proposed approaches.

REFERENCES

[1] R.F. Hodson et Al, “Recommendations on Use of Commercial-Off-The-Shelf (COTS) Electrical, Electronic, and Electromechanical (EEE) Parts for NASA Missions”, NASA/TM−20205011579, NESC-RP-19-01490, 2020.

[2] S. Esposito et al., "COTS-Based High-Performance Computing for Space Applications," in IEEE Transactions on Nuclear Science, vol. 62, no. 6, pp. 2687-2694, Dec. 2015.

[3] E. Chielle et al., "Reliability on ARM Processors Against Soft Errors Through SIHFT Techniques," in IEEE Transactions on Nuclear Science, vol. 63, no. 4, pp. 2208-2216, Aug. 2016,

[4] J. R. Azambuja, S. Pagliarini, L. Rosa, and F. L. Kastensmidt, “Exploring the limitations of software-based techniques in SEE fault coverage,”J. Electron. Test., vol. 27, no. 4, pp. 541–550, Aug. 2011.

[5] M. Rebaudengo, M. S. Reorda, M. Torchiano, and M. Violante, “Soft error detection through software fault-tolerance techniques,” in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst., Nov. 1999, pp.210–218.

[6] N. Oh, P.P. Shirvani, and E. J. McCluskey, “Control-flow checking by software signatures,” IEEE Trans. Rel., vol. 51, no. 1, pp. 111–122, Mar. 2002.

[7] Z. Alkhalifa, V. S. S. Nair, N. Krishnamurthy, and J. A. Abraham, “Design and evaluation of system-level checks for on-line control flow error detection,” IEEE Trans. Parallel Distributed Syst., vol. 10, no. 6, pp. 627–641, Jun. 1999.

[8] P. Bernardi, L. M. V. Bolzani, M. Rebaudengo, M. S. Reorda, F. L. Vargas and M. Violante, "A new hybrid fault detection technique for

systems-on-a-chip," in IEEE Transactions on Computers, vol. 55, no. 2, pp. 185-198, Feb. 2006

[9] J. R. Azambuja, M. Altieri, J. Becker, and F. L. Kastensmidt, "HETA: Hybrid error-detection technique using assertions," IEEE Trans. Nucl. Sci., vol. 60, no. 4, pp. 2805–2812, Aug. 2013.

[10] Entrena L. et al., "Fault-Tolerance Techniques for Soft-Core Processors Using the Trace Interface", in: Kastensmidt F., Rech P. (eds) FPGAs and Parallel Architectures for Aerospace Applications. Springer, Cham, 2016.

[11] A. Lindoso, L. Entrena, M. García-Valderas and L. Parra, "A Hybrid Fault-Tolerant LEON3 Soft Core Processor Implemented in Low-End SRAM FPGA," in IEEE Transactions on Nuclear Science, vol. 64, no. 1, pp. 374-381, Jan. 2017.

[12] M. Peña-Fernandez, A. Lindoso, L. Entrena, M. Garcia-Valderas, Y. Morilla and P. Martín-Holgado, "Online Error Detection Through Trace Infrastructure in ARM Microprocessors," in IEEE Transactions on Nuclear Science, vol. 66, no. 7, pp. 1457-1464, July 2019.

[13] A. Mahmood and E. McCluskey, "Concurrent error-detection using watchdog processors," IEEE Trans. Comput., vol. 37, no. 2, pp. 160–174, Feb. 1988.

[14] S. Bergaoui, P. Vanhauwaert and R. Leveugle, "IDSM: An improved disjoint signature monitoring scheme for processor behavioral checking," 2014 15th Latin American Test Workshop - LATW, 2014, pp. 1-6.

[15] M. Pignol, "Cots-based applications in space avionics," in 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010). IEEE, 2010, pp. 1213–1219.

[16] M. Pignol, "Dmt and dt2: Two fault-tolerant architectures developed by cnes for cots-based spacecraft supercomputers," in 12th IEEE International On-Line Testing Symposium (IOLTS'06). IEEE, 2006, pp. 1-10, July 2006.

[17] K. Greb and D. Pradhan, "HerculesTM microcontrollers: Real-time mcus for safety-critical products," Texas Instruments Inc. white Paper, 2011.

[18] R. Barry, "Mastering the freertos real time kernel," Real Time Engineers Ltd, 2016.

[19] E. Rotenberg, "AR-SMT: A microarchitectural approach to fault tolerance in microprocessors," in Proc. FTC, Madison, WI, USA, 1999, pp. 84–91.

[20] S. Mukherjee, M. Kontz, and S. Reinhardt, "Detailed design and evaluation of redundant multithreading alternatives," in Proc. ISCA, Anchorage, AK, USA, 2002, pp. 99–110.

[21] M. S. Reorda et al., "A low-cost SEE mitigation solution for softprocessors embedded in Systems on Programmable Chips", DATE, pp. 352-357, April 2009.

[22] M. Violante et al., "A Low-Cost Solution for Deploying Processor Cores in Harsh Environments", IEEE Trans. Ind. Electron.I, vol. 58, pp. 2617 - 2626, July 2011.

[23] H.-M. Pham et al., "Low-Overhead Fault-Tolerance Technique for a Dynamically Recongurable Softcore Processor", IEEE Trans. Comput., vol.62, no. 6, pp.1179-1192, June 2013.

[24] Á. B. de Oliveira, L. A. Tambara and F. L. Kastensmidt, "Applying lockstep in dual-core ARM Cortex-A9 to mitigate radiation-induced soft errors," 2017 IEEE 8th Latin American Symposium on Circuits & Systems (LASCAS), 2017, pp. 1-4.

[25] F. Abate, L. Sterpone, C. A. Lisboa, L. Carro and M. Violante, "New Techniques for Improving the Performance of the Lockstep Architecture for SEEs Mitigation in FPGA Embedded Processors," in IEEE Transactions on Nuclear Science, vol. 56, no. 4, pp. 1992-2000, Aug. 2009.

[26] H. Pham, S. Pillement and S. J. Piestrak, "Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor," in IEEE Transactions on Computers, vol. 62, no. 6, pp. 1179-1192, June 2013.

[27] C. M. Jeffery and R. J. O. Figueiredo, "A Flexible Approach to Improving System Reliability with Virtual Lockstep," in IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 1, pp. 2-15, Jan.-Feb. 2012.

[28] F. Abate, L. Sterpone, and M. Violante, "A new mitigation approach for soft errors in embedded processors," IEEE Transactions on Nuclear Science, vol. 55, no. 4, pp. 2063–2069, 2008.

[29] A. B. de Oliveira, G. S. Rodrigues, F. L. Kastensmidt, N. Added, E. L. Macchione, V. A. Aguiar, N. H. Medina, and M. A. Silveira, "Lockstep dual-core arm a9: Implementation and resilience analysis under heavy ion-induced soft errors," IEEE Transactions on Nuclear Science, vol. 65, no. 8, pp. 1783–1790, 2018.

[30] M. Peña-Fernández, A. Serrano-Cases, A. Lindoso, M. García-Valderas, L. Entrena, A. Martínez-Álvarez, S. Cuenca-Asensi, "Dual-Core Lockstep enhanced with redundant multithread support and control-flow error detection", Microelectronics Reliability, Volumes 100–101, pp.1-5, September 2019, 113447.

[31] Xilinx Inc., "Zynq-7000 soc data sheet: Overview," DS190, 2018.

[32] ARM Inc., "Cortex-A9 Technical Reference Manual", r4p1, 2012.

[33] Xilinx Inc., "Mailbox v2.1 LogiCORE IP Product Guide Vivado Design Suite," PG114, 2018.

[34] Xilinx. Inc, "Block memory generator v8.3 logicore ip product guide," PG058, 2017.

[35] M. Y. Hsiao, "A Class of Optimal Minimum Odd-weight-column SEC-DED Codes," in IBM Journal of Research and Development, vol. 14, no. 4, pp. 395-401, July 1970.

[36] Digilent Inc., "Zybo FPGA Board Reference Manual", 2016.

[37] W. E. Burr, "Selecting the Advanced Encryption Standard," in IEEE Security & Privacy, vol. 1, no. 2, pp. 43-52, March-April 2003.

[38] R. Velazco, S. Rezgui, and R. Ecoffet, "Predicting error rate for microprocessor-based digital architectures through C.E.U. (Code Emulating Upsets) injection," IEEE Trans. Nucl. Sci., vol. 47, no. 6, pp. 2405–2411, Dec. 2000.