Received December 6, 2019, accepted January 17, 2020, date of publication January 27, 2020, date of current version February 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2969714

Survey of Soft Error Mitigation Techniques Applied to LEON3 Soft Processors on SRAM-Based FPGAs

SERVER KASAP, (Member, IEEE), EDUARDO WEBER WÄCHTER, XIAOJUN ZHAI, (Member, IEEE), SHOAIB EHSAN, (Senior Member, IEEE), AND KLAUS MCDONALD-MAIER, (Senior Member, IEEE)
School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K.

Corresponding author: Server Kasap (server.kasap@essex.ac.uk)

This work was supported by the UK Engineering and Physical Sciences Research Council under Grant EP/R02572X/1 and Grant EP/P017487/1.

ABSTRACT Soft-core processors implemented in SRAM-based FPGAs are an attractive option for applications to be employed in radiation environments due to their flexibility, relatively-low application development costs, and reconfigurability features enabling them to adapt to the evolving mission needs. Despite the advantages soft-core processors possess, they are seldom used in critical applications because they are more sensitive to radiation than their hard-core counterparts. For instance, both the logic and signal routing circuitry of a soft-core processor as well as its user memory are susceptible to radiation-induced faults. Therefore, soft-core processors must be appropriately hardened against ionizing-radiation to become a feasible design choice for harsh environments and thus to reap all their benefits. This survey henceforth discusses various techniques to protect the configuration and user memories of an LEON3 soft processor, which is one of the most widely used soft-core processors in radiation environments, as reported in the state-of-the-art literature, with the objective of facilitating the choice of right fault-mitigation solution for any given soft-core processor.

INDEX TERMS LEON3 soft-core processor, fault tolerance, spatial redundancy, temporal redundancy, software redundancy, SEE, SEU, soft errors.

I. INTRODUCTION

One of the most significant and complex environmental remediation tasks in the whole of Europe is the cleaning-up process of the legacy nuclear waste, which is projected to cost more than £115bn and perhaps as high as £220bn, over the next 120 years [1]. Cleaning up radioactive waste inside a nuclear power station is too hazardous to be done by human beings due to the severe adverse effects of the ionizing radiation on biological tissues, which abound in these environments primarily in the aftermath of a severe nuclear accident. Consequently, there is a strong motivation and desire to use robots, and consequently, electronic devices to enter radiation facilities, e.g. nuclear power plants, nuclear waste disposal sites.

The associate editor coordinating the review of this manuscript and approving it for publication was Cristian Zambelli.

When using electronic devices, human beings are spared from entering harsh environments. However, this is not a straightforward task because electronic circuits in these robots are also susceptible to the radiation effects; this has become even clearer after the Fukushima Daiichi nuclear power plant in Japan suffered a series of meltdowns as a result of the failure of its safety systems due to a tsunami. Robots dispatched into the accident site to monitor radiation levels and facilitate the cleaning-up process have kept breaking down and failing very soon after entering as their circuits were destroyed by the radiation, thus turning the entrance of the facility into a robot graveyard [2]. Henceforth, if robots are to be deployed in such scenarios, the behaviour of electronic circuits in extreme nuclear radiation environments remains to be thoroughly studied, and radiation effects to be mitigated.

There are two significant cases where electronic equipment can be employed in order to spare human lives. The first one is to manage (move, dispose of) nuclear waste generated by nuclear plants, and the second is the case of a nuclear disaster. In both cases, using robots instead of human beings is still a much more rational solution. Additionally, robotics will play an increasingly important role in in-service maintenance and inspection of the current nuclear fleet to support plant life extension (PLEX). Finally, robotic systems will soon become an essential design element of new-built reactors, as well as helping to build them in the first place, thereby reducing risk and cost while improving safety at the same time.

Soft errors in electronic circuits are consequences of faults induced by energetic particles (e.g. alpha particles, heavy ions, neutrons) or electromagnetic waves (e.g. X-rays, gamma rays) striking the semiconductor substrate of transistors [3] in radiation environments such as nuclear facilities or outer space. These errors have a transient behaviour that does not permanently damage digital circuits. However, they adversely affect the reliability of systems to a great extent, especially if the integration level of the chip is high, leading to a further decrease in dependability [4]. For instance, a number of mission critical applications have been recently implemented in Systems-on-Chips (SoCs) built on SRAM-based FPGAs which offer the benefits of higher flexibility, lower cost, reduced time-to-market and capability of dynamic hardware reconfiguration [5]. Unfortunately, this kind of highly-integrated circuits involving several hard-core or soft-core processors are very prone to transient faults which can easily cause overall system failures. Therefore, developing and employing techniques to mitigate radiation-induced transient faults are of fundamental importance.

Even if soft-core processors can not beat hard-core processors in terms of performance, area, and power consumption, SoCs incorporating soft-core processors are becoming very popular nowadays in the domain of embedded systems; this is due to the fact that soft-core processors can be implemented in any FPGA of any technology/supplier, and can be easily customized for a specific application in order to better accommodate its particular requirements. However, soft-core processors must be seriously protected against ionizing radiation to be an attractive design choice for harsh environments since there are unique fault modes for soft-core processors with respect to hard-core processors –that is, not just the user memory but also the configuration memory controlling both the logic and signal routing circuitry of a soft-core processor on an SRAM-based FPGA is susceptible to radiation-induced faults.

Therefore, adoption of fault-mitigation techniques is the only viable way to use soft-core processors in radiation environments. In this regard, multiple approaches have been proposed in the literature, however, no survey has ever been published till today, to the best of our knowledge, which puts together developed and deployed techniques for the protection of soft-core processors against soft errors arising in their configuration and user memories when implemented in SoCs built on SRAM-based FPGAs. This paper has been written to fill in this gap, and presents a thorough survey of soft error mitigation techniques as applied for protecting the configuration and user memories of LEON3 soft processors deployed on SRAM-based FPGAs. LEON3 is one of the most widely used soft-core processors in radiation environments, e.g. space, as reported by many research papers in the state-of-the-art literature.

While choosing which research papers to include for this survey paper, we searched through all the major relevant research libraries, e.g. IEEE Explore digital library, ACM digital library, ScienceDirect etc., to find all papers published in the last 20 years which discuss fault-mitigation techniques as applied to any soft-core processor against soft errors. When we established that LEON3 is the soft-core processor reported in most of the resulting papers as the target processor on which fault-mitigation techniques are applied, we proceeded with shortlisting LEON3 related papers based on two criteria, i.e the number of citations received by the paper so far and the prominence of its authors in the field, with the objective of including only the most promising research papers in our survey. Note that only soft errors caused by single-event effects have been studied in this paper because they are the most widespread radiation effect for soft-core processors on SRAM-based FPGAs [4]. Other radiation-induced fault types are total ionizing dose and displacement damage, as will be further discussed in Section II-A, and they have been left out of the scope of this paper so as to have more space to delve into techniques dealing with soft errors.

All fault-mitigation solutions presented in this survey employ one or more forms of redundancy [6] (which broadly refers to incorporating additional capabilities into the system which would not be required in a radiation-free environment [7]), such as spatial redundancy, temporal redundancy, software redundancy or information redundancy, along with other techniques in order to empower systems to continue their operation effectively and efficiently even when transient faults occur in their critical components. Furthermore, fault-mitigation techniques described can be readily applied to other soft-core processors implemented in SRAM-based FPGAs, as the ultimate objective of this paper is to facilitate the choice of right fault-mitigation solution for any given soft-core processor.

The paper is organized as follows. Section II provides the effects of radiation on electronics in general, and soft errors it causes in SRAM-based FPGAs in particular. Section III briefly describes the LEON3 processor which is the subject soft-core processor of this paper, and in Section IV, prominent research papers proposing various soft error mitigation solutions for LEON3 processors implemented on SRAM-based FPGAs are presented, and their results are evaluated in many terms, under four subsections depending on the types of redundancy techniques they utilize individually or in conjunction. Finally, conclusions are drawn in Section V.

II. BACKGROUND

Soft-core processors which are implemented in an FPGA fabric are an attractive option for applications to be employed in radiation environments by virtue of their flexibility, relatively-low application development costs, and reconfigurability features applicable when deployed on an SRAM-based FPGA – that is, the FPGA fabric hosting the soft-core processor can be reprogrammed in order to accommodate the evolving mission goals or amend possible errors in the design. Despite the advantages soft-core processors within FPGA fabric can bring to these applications, they are seldom used because they are more sensitive to radiation than their hard-core counterparts which are actually implemented in the silicon as a physical structure.

Striking high-energy particles and electromagnetic waves cause faults arising in the FPGA fabric occupied by the soft-core processor. Faults can occur both in the configuration-memory and user-memory bits of the FPGA. Thus, soft-core processors have unique fault modes in contrast to hard-core processors. Consequently, the adoption of fault-mitigation or fault-tolerance techniques is vital if it is desired to employ soft-core processors in radiation environments.

Before introducing soft-core processor fault tolerance, it is crucial first to discuss what fault-tolerance techniques must protect in a soft-core processor. This section motivates the need for protection with a discussion on fault modes in SRAM-based FPGAs. It begins by discussing the effects of radiation on electronics in general and then identifies the unique fault modes of soft-core processors through a discussion on how faults affect an FPGA's configuration and user memories.

A. RADIATION EFFECTS ON ELECTRONICS

Ionizing radiation can damage electronics in three significant ways. Radiation effects on electronics can either temporarily change the behaviour of a circuit (a soft error), or permanently damage the circuit (a hard error). However, radiation can also cause an error that has characteristics of both a hard and a soft error (a firm error) in FPGA designs.

The first-way radiation damages electronics is called as Total Ionizing Dose (TID) [8] which refers to the cumulative, permanent damage in an electronic device causing it to degrade over time (i.e. hard error). It takes place when charge carriers are implanted into the device's insulators as radiation strikes, where they consequently get trapped altering the electrical characteristics of the integrated circuits [9].

The second major category of radiation-induced adverse effects is generally called as single-event effects (SEE) [8]. Most often, this type of faults is transient (i.e. soft errors) and do not cause permanent damage like TID, but they may still induce unwanted behaviour changes. All these transient effects stem from excess charge carriers generated through the ionization of silicon atoms by radiation. If a sufficient amount of these charges gathers in a certain area, the logic value of a line in that area can be upset; this event is referred to as a single-event transient (SET) which has a brief effect until the excess charge dissipates.

In case of an SET, if a storage device captures the new state of the line, there would be a longer lasting effect on the system output, which is identified as a single-event upset (SEU). Nevertheless, an SEU can be generally amended by restoring all flip-flop (FF) values through a system reset. However, it is not possible to fix some SEUs by a simple reset; these type of SEUs are called as single-event functional interrupts (SEFIs). There is another radiation effect called single-event latch-up (SEL) which takes place when ionizing radiation turns on parasitic transistors in the silicon. These parasitic transistors can keep conducting current until a system reset, causing parts of the device burn in some cases (i.e. hard error) [10].

The third and last significant radiation effect is called displacement damage (DD) [8], which occurs when a high-energy particle displaces silicon atoms out of their positions in the silicon lattice while passing through the device. These displacements cause silicon substrate defects, which has the potential to alter the electrical characteristics of the device. Fortunately, this effect is not often observed in FPGA devices.

Like other electronic devices, radiation has effects on FPGAs the degree of which largely depends on the type of FPGA being employed. FPGAs are divided into three major categories, based on how they deploy configuration data, which include antifuse, flash, and SRAM-based FPGAs. Antifuse FPGAs use non-volatile configuration memory composed of fuses which can be programmed only one time to set each and every configuration bit. Since the configuration data is permanent for antifuse FPGAs, users do not have the convenience of updating or amending the functionality of the device in the field. However, antifuse FPGAs are very reliable against SEUs because their configuration cells made up of simple fuses which are resistant to single-event effects. Nevertheless, antifuse FPGAs are still susceptible to SEUs within their user memory.

On the other hand, non-volatile flash memory cells are employed in flash FPGAs to set the configuration memory bits. As an advantage, flash FPGAs can be reprogrammed in the field for a limited number of times. Furthermore, flash memory cells built with electrically-isolated floating gate offer resistance to SEUs, however, user flip-flops and block memories are vulnerable to SEUs like in the case of antifuse FPGAs. On the downside, flash FPGAs are less tolerant to TID than other FPGA types due to the internal working mechanism of the flash memory, which limits the amount they can be used in harsh environments. The rest of the paper will focus on SRAM-based FPGAs.

B. SOFT ERRORS IN SRAM-BASED FPGAs

An SRAM-based FPGA is an integrated circuit (IC) whose logic and routing matrices can be easily reprogrammed unlimited number of times in the field [5]. The circuitry of an SRAM-based FPGA includes a vast array of reprogrammable logic resources and a rich interconnect system which can be used to create as large and complex digital circuits as a processor. Since designs are implemented on a programmable hardware fabric, SRAM-based FPGAs are extremely flexible and provide a low application development cost for designs. This flexibility and reprogrammability make SRAM-based

FPGAs ideal for a plethora of applications in nuclear power plants [9] and space [11], as they can be reprogrammed even during run-time, thus adapting to the evolving mission goals or rectifying design errors. Nevertheless, as they employ static memory cell, SRAM-based FPGAs are volatile, i.e. they lose their configuration data, thus functionality, once power is removed, and must be reprogrammed from external memory every time they are turned on.

The ability to reconfigure an SRAM-based FPGA is by the virtue of the presence of internal SRAM cells which are collectively called the configuration memory. For instance, signal routing matrices within the FPGA are controlled with some of these SRAM cells in the configuration memory. Changing the routes through a routing matrix is accomplished by changing the values of these configuration memory bits. Bits in the configuration memory also control logic elements such as look-up tables (LUTs), multiplexers, and flip-flop attributes. A large configuration memory, i.e. millions of configuration bits, is required to control all of the routing and logic circuitry within the entire FPGA.

From a reliability point of view, the primary concern for SRAM-based FPGAs is SEUs occurring within their configuration memory. An SEU occurring in an FPGA's configuration memory can cause a firm error in either logic or interconnect part of the design. An upset is classified as a logic upset if it occurs in a region of the configuration memory controlling any logic element or any logic element attribute. An SEU may upset a bit in an LUT which will in return manipulate the logic function the circuit is supposed to perform. Another unique FPGA failure mode stems from upsets occurring in configuration memory regions dictating the signal interconnection. Firm errors in signal routing can be mainly classified either as an open error, i.e. SEU causes an open route, or a short error, i.e. SEU causes two routes to drive a single input.

The configuration memory is not the only FPGA memory sensitive to upsets. User memory is also susceptible to upsets. User memory refers to registers and memories used within a design, i.e. flip-flops or SRAM cells of block RAMs (BRAMs). For example, the user memories in a soft-core processor include the main memory, register file, caches, and pipeline registers. Notably, there are much fewer user memory bits than configuration memory bits.

III. THE LEON3 PROCESSOR

The processor considered in this paper is 32-bit LEON3 [12] soft-core processor from Aeroflex Gaisler. This processor has been chosen for this survey due to its widespread use in harsh environments involving ionizing radiation such as space; LEON3 is one of the processors used by the European space agency [13]. The LEON3 processor is an open-source processor which has performance and efficiency comparable to other well-known soft-core processors while being one of the most configurable soft-core processors [14].

LEON3 complies with the SPARC V8 instruction set architecture (ISA), and has a processing pipeline coupled

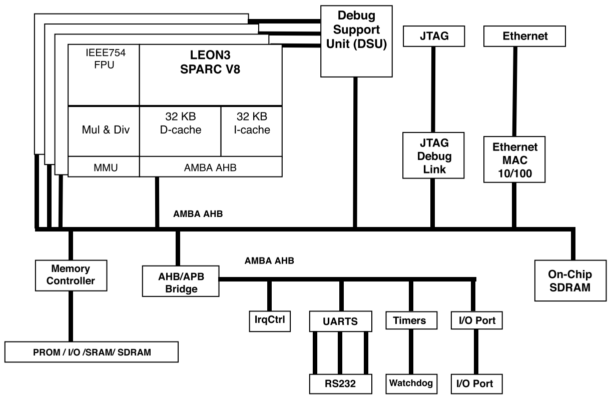


FIGURE 1. The LEON3 processor system design [14].

with separate instruction and data caches, floating-point unit (FPU), co-processor unit, and multiprocessing capability [15]. Both the data and instruction caches can be configured with up to four sets of 1 to 256 KB per set, and 16- or 32-bytes per line. Furthermore, the register file uses windowed registers with up to 32 windows. Each register window consists of 32 registers: eight global registers, eight input registers, eight output registers, and eight local registers, where global registers are shared by all register windows.

Typically, the LEON3 processor includes a seven-stage integer pipeline, a ten-window register file, a hardware multiplier and divider, a memory management unit (MMU), 32 KB direct-mapped, write-through instruction and data caches, and the on-chip memory implementable with Block RAMs (BRAMs) on a Xilinx FPGA [16]. The LEON3 system design also encompasses several auxiliary units such as an AMBA bus, AMBA bus controllers and interfaces, a default memory controller, a default interrupt controller, a debug support unit (DSU) and debug port, a trace buffer, a translation look-aside buffer (TLB) among others, as shown in Fig. 1.

LEON3 processors must be significantly protected against ionizing-radiation to be an attractive design choice for harsh environments when deployed on an SRAM-based FPGA as a soft-core processor. As explained above, the logic and routing within the soft-core processors can be seriously affected by SEEs, since the logic and routing of a soft-core processor is implemented in the configuration memory of an SRAM-based FPGA. The user memory of soft-core processors is also vulnerable to SEEs.

Therefore, the next section discusses various ways of protecting the configuration and user memories for LEON3 processors implemented in SRAM-based FPGAs, as reported in the literature.

IV. SOFT ERROR MITIGATION SOLUTIONS FOR LEON3 PROCESSORS ON SRAM-BASED FPGAs

In this section, most prominent research papers proposing mitigation solutions for soft errors in LEON3 processors implemented on SRAM-based FPGAs are thoroughly evaluated after they are broadly classified into four categories depending which type of redundancy they utilize alone or in conjunction with each other.

A. SPATIAL REDUNDANCY BASED SOLUTIONS

In [17], a non-intrusive fault-tolerance technique has been described for soft processors embedded in SRAM-based FPGAs, which was developed targeting LEON3-based On-Board Computer (OBC) systems to be employed in the satellite construction missions of the Brazilian Institute of Space Research (INPE). The proposed technique has the ability to identify run-time errors through bus monitoring, without any modifications to the application software, with the goal of preventing an error in the OBC processor (i.e. LEON3) to propagate across the whole system.

There are two SRAM-based FPGAs in the proposed OBC board along with the output selection circuit connected, where the proposed technique implements on-line fault detection mechanisms in order to identify run-time errors in one of the FPGAs, thus avoiding error propagation to the rest of the satellite system by masking faults through spatial redundancy.

On each FPGA, a LEON3-based system-on-chip (SoC) has been deployed in which a second (redundant) processor and a specially developed Bus Monitor have been added to the AMBA AHB bus as another instance of spatial redundancy – see Fig. 2. These two processors are executing same instructions over the same data, where the redundant processor does not have the authority to write to the bus, while the bus monitor is tasked with comparing data produced by both processors in order to check whether outcomes of the main processor match those of the redundant one. If there is a mismatch in one of the two SoCs, the error signal of the corresponding Bus Monitor will be asserted to be used by the output selection circuit to disregard the erroneous data coming from that SoC and provide the correct data coming from the other SoC to the rest of the satellite system. On the downside, the proposed technique does not incorporate any mechanism to recover the faulty processor and re-synchronize it with other processors. However, advantages of this technique are twofold:

1) The area overhead per each FPGA is significantly reduced compared to triple modular redundancy (TMR) [18] technique as only one redundant processor is added to the SoC.
2) There is no intrusion to the application software running on the processors, thus avoiding performance losses which could be the case for fault-tolerance techniques interfering with the software.

The proposed fault-mitigation technique was validated through fault injection experiments which are conducted by executing a TCL script within the Mentor ModelSim simulation tool. The simulation script is designed to inject a fault to each and every internal signal of OBC in one-at-a-time fashion through a set of simulations, with faults injected at different simulation times for each internal signal. Authors claim that two Bus Monitors in the overall system were able

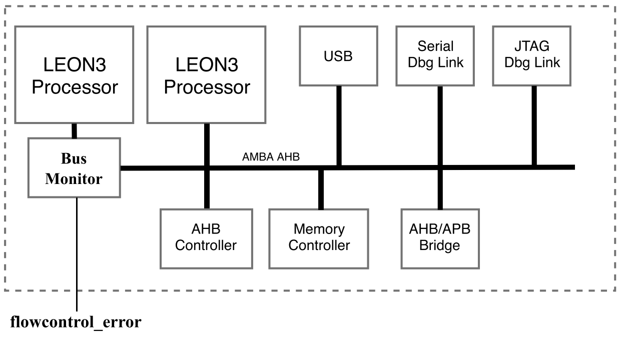


FIGURE 2. LEON3 architecture with a redundant processor [17].

to detect all failures arising from fault injections, therefore all failures could have been masked by the Output Selection Circuit. Note that neither emulation-based fault injection nor accelerated radiation testings have been done. More seriously, no particular FPGA chip or board has been mentioned for the actual implementation of the proposed technique. Consequently, there is no FPGA resource utilization or timing data presented in the paper.

In [19], authors propose a fault-mitigation approach for FPGAs where functional units can be replaced by a hardware spare unit in case of a fault, presenting an example of spatial redundancy. The proposal comprises two sets of Replaceable Functional Units (RFU) and Spare Functional Units (SFU). Critical RFUs are initially hardwired on the device (i.e. they are placed in a non-reconfigurable area), and when found defective, they are replaced by an SFU mapped inside the reconfigurable area. Non-critical RFUs instead are initially mapped inside the reconfigurable area, and are replaced by a software SFU if the reconfigurable area is needed to host a critical spare unit.

An SoC based on LEON3 processor, depicted in Fig. 3, was used as the target in this work where the major objective was to protect the Arithmetic Logic Unit (ALU) within the LEON3 processor. The ALU was identified as a critical RFU, while the utilized Data Encryption Standard (DES) crypto-core was regarded as a non-critical RFU. Consequently, the reconfigurable area was initially occupied by the DES crypto-core. In the case where the ALU embedded in ALU fails, the previously allocated DES would be swapped at run-time with the hardware SFU implementing ALU via dynamic partial reconfiguration. In this scenario, the de-allocated DES hardware core would be replaced with its corresponding software SFU in the LEO3 processor.

The given approach was synthesized and tested on a Xilinx Virtex-4 FPGA. It has been shown that the system can keep working normally after a reconfiguration, but there is an increase, as expected, in the execution time, from 183 ms to 1500 ms, when executing the DES algorithm in software rather than in hardware. Note that the reconfigurable area was allocated with 536 slices which can be regarded as the area overhead of the proposed approach. To verify the correct operation, faults were manually injected into the system through a switch on the board; When the board switch is

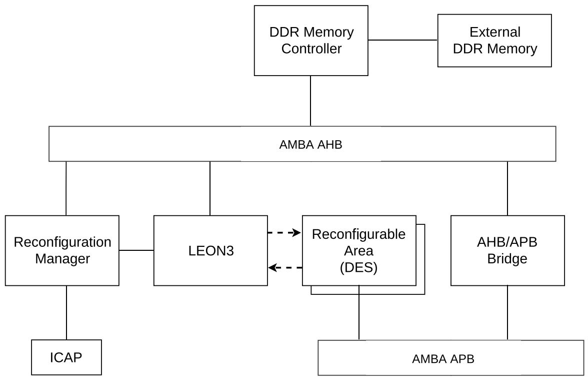


FIGURE 3. Conceptual block diagram of the proposed architecture [19].

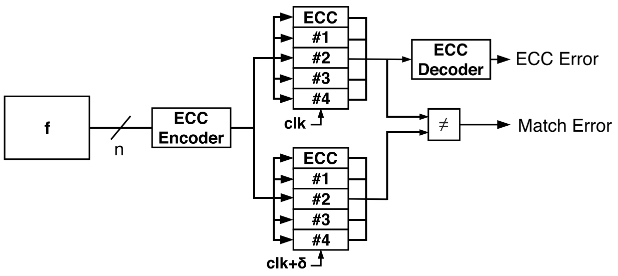


FIGURE 4. Shadow register technique extended with ECC units [20].

asserted, the fault injection mechanism in the system was triggered to emulate a fault. The details of the fault injection campaign was not provided. As another shortcoming, no actual radiation test for this work was reported in the paper. Furthermore, the approach presented in this work merely protects one small module in the processor, i.e. the ALU, while the remaining modules are left unprotected. Moreover, the area overhead is always bounded to the biggest module, which may cause inefficiency in the resource utilization if smaller modules occupy significantly less space.

Boujajila *et al.* [20] has proposed an spatial redundancy based fault mitigation technique for RISC processors in general, while the actual implementation was merely carried out for LEON3 processors. The underlying objective of the proposed approach is to detect and correct any faults occurring in the pipeline registers of the processor. To achieve the given objective, pipeline registers are replicated and then coupled with ECC units as shown in Fig. 4, where register replicas are referred to as shadow registers. The modified version of the pipeline works as follows: whenever the utilised ECC modules supervising pipeline registers detect an error, the pipeline is frozen at the current clock cycle and then the corrupted data is corrected in the second clock cycle deploying the complex mechanism explain in Section III-A of [20]. Therefore, any error will be detected and corrected with a fixed penalty of two clock cycles.

Authors have prototyped this technique on a Xilinx Virtex-IV FPGA, and verified its fault tolerance reliability by introducing instruction execution errors at a variable rate via random fault injection into pipeline registers. To measure the execution overhead, they compared the Cycles per Instruction (CPI) performance of the proposed approach to that of the baseline design. These experiments have shown that for an error rate of 4%, the CPI has increased to 1.09, whose nominal value is 1 under no fault scenario. Furthermore, the proposed approach exhibits an area overhead of 17%, 29% and 28% in terms of Flip-Flops (FFs), LUTs and slices, respectively, with respect to the unmodified design. As a shortcoming, authors have not included any results regarding the clock frequency shift as would be caused by the design modifications; a critical path increase is highly expected since more elements are added between the pipeline registers, which causes a clock frequency reduction. Finally, no radiation test was carried out for this work.

In [21], various combinations of three SEU mitigation and repair techniques, i.e. TMR (a widely used technique based on spatial redundancy), internal block memory scrubbing (BRAM scrubbing) [11] and configuration memory scrubbing (CRAM scrubbing) [11], are utilized in an attempt to study and and evaluate the effects of each technique on the radiation sensitivity of the LEON3-based SoC implemented on SRAM FPGAs. In total, authors have conducted experiments on five combinations of the mentioned techniques:

1) Unmitigated SoC design (reference design)
2) SoC design with TMR
3) SoC design with TMR and BRAM scrubbing
4) SoC design with TMR and CRAM scrubbing
5) SoC design with TMR, BRAM scrubbing and CRAM scrubbing

Same authors previously reported the outcomes of similar experimentation over only one case study, where all the aforementioned SEU mitigation and repair techniques were applied together to the LEON3 soft processor, in [22].

Each of these five LEON3-based system variations or versions incorporates only the core architecture of the LEON3 with a minimal set of peripherals and no cache memory, which obviously prohibits the full coverage of the SEU failure modes and therefore, facilitates simpler experimentation flow. All but the first of these system versions employ full TMR at the fine granularity level, which means the entire design is replicated at the primitive level where triplicated majority voters are placed in the feedback paths. Furthermore, version #3 and version #5 deploy BRAM (used to implement instruction and data memory) scrubbing in the processor systems, where the scrubbing logic goes through each memory address and continuously writes the correct value to that memory locations, as determined by voting between triplicated BRAM copies (see Fig. 5). Moreover, in processor system versions #4 and #5, external read-back CRAM scrubbing is performed over JTAG using a high-speed JTAG controller, where the current configuration of the FPGA is compared bit-by-bit against the golden copy to detect erroneous bits and correct them by writing back the correct bit value via partial reconfiguration.

All five LEON3 processor system variations are implemented one-by-one on the same Xilinx Kintex 7 FPGA chip mounted on the KC705 evaluation board. Since each

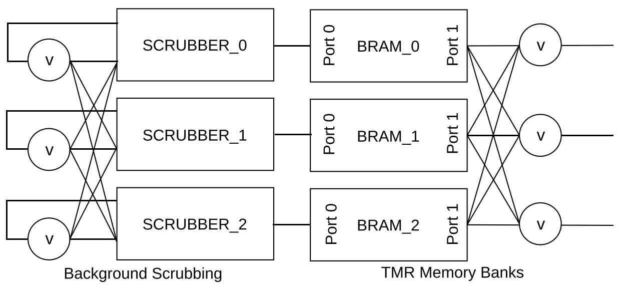


FIGURE 5. Internal BRAM scrubbing [21].

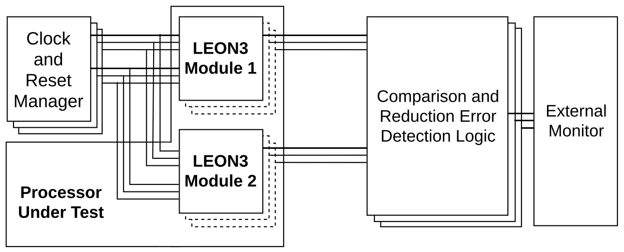


FIGURE 6. SEU testing infrastructure [21].

version employs different SEU mitigation and repair techniques, the resource consumption on the FPGA varies across the versions, where design with TMR only (version #2) and design with TMR and BRAM scrubbing (version #3) occupy 3.9× and 4.8× more slices, as compared to the unmitigated reference design (version #1). Note that versions #4 and #5 occupy same number of slices as version #3 since CRAM scrubbing does not require any additional resources.

Both emulation-based fault injection and neutron radiation testing were used to evaluate each LEON3 system variation for SEU fault tolerance through the same testing infrastructure whose purpose is to detect when the LEON3 processor produces incorrect outputs. Within the mentioned testing infrastructure, two identical processor systems are incorporated onto the same FPGA as shown in Fig. 6, where LEON3 processors run in parallel and bus signals from each processor are compared on a clock-by-clock basis to produce a single failure indicating signal; this signal is attached to the JTAG boundary scan for external sampling. Furthermore, Dhrystone benchmark is continuously executed on both processors to allow errors, if any, to propagate throughout the system so as to cause failures which are then detected and signalled by the comparison circuit available within the testing infrastructure. The foremost outcome of both fault injection and neutron radiation testing is that the sensitivity of the processor system to SEU-induced failures is getting reduced, i.e. improved, incrementally as we traverse from design variant #2 to design variant #5, in comparison to the unmitigated design variant. Finally, processor system design featuring all the three fault mitigation and repair techniques (version #5) offers 51.3× and 28.9× improvement in radiation sensitivity, respectively, as measured in fault injection and neutron radiation experiments.

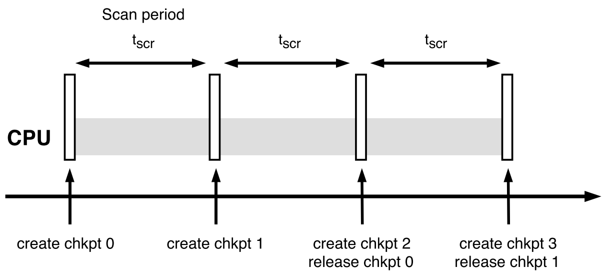


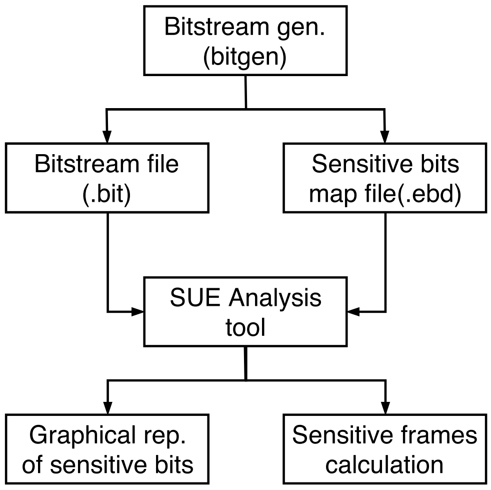
FIGURE 7. Checkpointing with configuration memory scan [23].

B. TEMPORAL REDUNDANCY BASED SOLUTIONS

An SEU mitigation approach was proposed in [23] and [24] for processors embedded in SoPCs where configuration memory scan/scrubbing was used for fault detection and fault repair while checkpointing with rollback recovery technique [25], which is based on temporal redundancy, was employed for fault recovery; notably, checkpointing with rollback recovery cause redundant time delays in process executions – the reason why these approaches are regarded as temporal redundancy-based solutions. In this approach, only the sensitive frames of the configuration memory are scanned periodically to detect SEUs, thus the scan period depends on the number of sensitive frames of the design, while context of the processor is stored at the end of each scan interval during a checkpoint, as illustrated in Fig. 7. Authors has reported that the given approach uses the SEU detection and correction related IP cores built for Xilinx Virtex FPGA devices where each configuration memory frame is protected by Error Correcting Code (ECC), and therefore the fault repair can be done using the error correction capability of the ECC unit whenever faults are detected through scanning.

In the above papers, the configuration memory scrubbing time of a LEON3-based System-on-Programmable Chip (SoPC) design was estimated counting the number of design configuration frames which contain sensitive bits [11] (i.e. counting sensitive configuration frames). Then, a constraint-driven re-placement method was proposed utilizing the idea of column-based placement [26] in order to reduce the number of sensitive configuration frames, and to consequently minimize the scrubbing time. The proposed method's effectiveness was verified on the given SoPC design.

Furthermore, the information from the sensitive bits map file produced by the Xilinx implementation tool-chain was employed along with the bit-stream generated in order to identify how sensitive bits were distributed on the FPGA layout, and to calculate the number of sensitive configuration frames, as illustrated in Fig. 8. This analysis process was applied to the LEON3-based SoPC implemented on a Xilinx Virtex-5 device, whose results have showed that although only 14.16% of the FPGA configuration bits have been found to be sensitive bits, the FPGA slice utilization rate is 46%, and a large portion of the configuration frames are characterized as sensitive frames by a percentage of 84.9%.



Bitstream gen. (bitgen)
Bitstream file (.bit)
Sensitive bits map file(.ebd)
SUE Analysis tool
Graphical rep. of sensitive bits
Sensitive frames calculation

FIGURE 8. Sensitive bits analysis process [23].

As a consequence, a large number of frames contain few sensitive bits causing time to get wasted by the scrubbing process for scanning these underutilized frames. This situation arises due to the problem of inefficient logic placement in the FPGA.

As a remedy to this problem, authors propose a constrained-placement (CP) method to be applied after the initial placement whose target is to maximize the utilization of configuration frames. The steps of the given method are thoroughly explained in section 3 of [23]. With the application of the CP method, it has been reported that the rate of sensitive frames has been reduced from 84.9% to 49.86%, thus an improvement of 41.29% has been achieved in comparison to the initial (i.e. unconstrained) placement, while the clock frequency degradation is negligibly small. The direct consequence of this result is the reduction in the configuration memory scan time by 41%. The CP method was advanced in [24] by introducing a selective scrubbing approach where selected system components not used at a particular time during task execution are not scanned, thus further reducing the scrubbing time by 23%.

Finally, the LEON3-based SoPC system reliability was analytically evaluated using the number of sensitive frames and the configuration memory throughput in order to prove the reliability improvement owing to the reduction in sensitive frame counts achieved by the CP method. The reliability of the SEU mitigation approach proposed was measured in terms of mean-time-to-detect (MTTD) and mean-time-to-repair (MTTR) parameters. It is important to note here that the checkpoint and rollback mechanisms were not actually implemented by the authors, and therefore their impact on the system performance was not considered. Finally, the CP method applied within the proposed SEU mitigation approach reduces the MTTD and the MTTR by 41% in comparison to the case where only unconstrained placement is applied. Furthermore, it was reported that if the selective scrubbing method is incorporated as well in the approach, the reduction in the parameters of MTTD and MTTR would be even better, i.e. 54%. Note that neither fault injection nor irradiation experiments have been conducted to practically measure the reliability of the proposed approach.

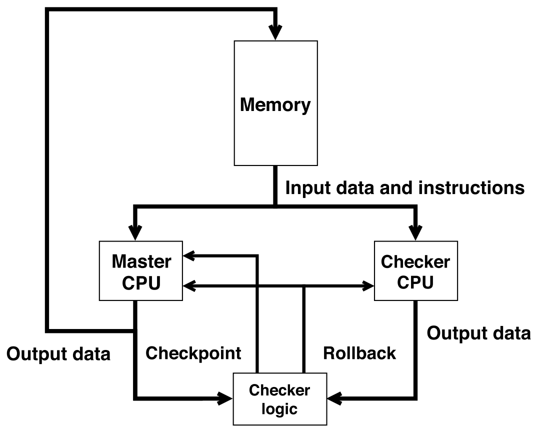


FIGURE 9. Architecture of the lockstep system [27].

C. SPATIAL AND TEMPORAL REDUNDANCY BASED SOLUTIONS

In [27] and [28], a design flow is described for FPGA-based SoPCs which can be adopted to mitigate radiation-induced faults (i.e. SEUs) affecting embedded processor IP cores in harsh environments. The design flow employs three different fault-tolerance methods, i.e. lockstep technique (also known as Duplication with Comparison (DWC) technique) [29] which draws on spatial redundancy, checkpoint with rollback recovery technique (based on temporal redundancy), and on-demand configuration memory scrubbing, where the objective is to provide a balance between resources overhead and reliability. The authors claim that the proposed flow reduces the required hardware resources and makes it possible to easily harden processor cores within new SoPCs which are bound to work in harsh environments. The design flow was furthermore applied in the design of an SoPC-based system with a complex processor IP core, and results of the application were reported.

The proposed processor hardening approach is based on a combination of lockstep for fault detection, checkpointing with rollback for fault recovery, and configuration memory scrubbing for fault repair, which is achieved as follows:

1) The processor in the system is duplicated where two exact copies of the processor (named as the master and checker processors) are synchronized on clock-by-clock basis (i.e. lock-stepping) while executing the same software (see Fig. 9).
2) A monitor module is added to the system in order to detect radiation-induced SEUs occurring either in the master processor or in the checker processor by comparing the bus activities of the two processors, as shown in Fig. 10.
3) If no errors are detected by the monitor module, it periodically issues an interrupt request to the master processor, so that a copy of the system's context

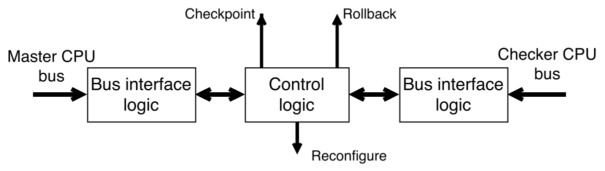


FIGURE 10. **Checker logic architecture [27].**

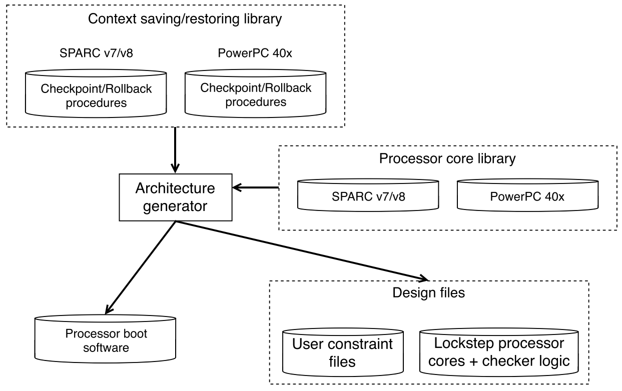


FIGURE 11. **Automated design flow [28].**

(i.e., master processor's register values and contents of the data memory) is stored in a safe memory (i.e. checkpointing).

4) If an error is detected, an interrupt request is issued to both processors, so that the system's context is replaced with the previously saved one, thus rolling back processors to a previous presumably-good state (i.e. rollback recovery).
5) If the monitor module still detects errors after a rollback before reaching a new checkpoint, it is assumed that there occurred persistent SEUs in the FPGA configuration memory (i.e. firm error). To recover from these firm errors, the monitor forces the reconfiguration of the FPGA (i.e. on-demand configuration memory scrubbing), and then it resets both processors forcing them to restart the execution.

Following the above approach, the authors have proposed an automated design flow which can effectively support the safety-critical system designers by streamlining the modification of the processor IP at HDL level, thus applying the proposed hardware hardening solution, and by incorporating interrupt routines for context storage and context recovery operations into the processor boot software. This design flow is illustrated in Fig. 11.

Authors have performed experiments as well to assess the effectiveness of their proposed design flow using an SoPC system based on the LEON3 processor. To evaluate the area overhead, the resource utilization of the TMR version of the LEON3 processor as obtained by the XTMR tool from Xilinx has been compared with that of the duplicated LEON3 processor architecture resulting from the proposed design flow. It has been found that the architecture obtained from the proposed design flow requires fewer logic cells and memory blocks by 43% and 34%, respectively, with respect to the TMR version. Furthermore, it has been found that the critical path of the TMR version is 25% longer than that of the proposed architecture. Consequently, the proposed design flow produces processor architectures with faster clock frequencies with respect to the TMR approach.

Furthermore, four software benchmark applications (i.e. elliptic filter, FIR filter, Kalman filter and matrix multiplication) were considered to evaluate the time overhead of the approach inflicted by the involved context storage periodically performed at checkpoints. In these experiments, where one checkpoint every 300 write cycles was committed, it was found that the time overhead ranges from 17.7% to 53.8% depending on the amount of data to be stored during a checkpoint. No timing overhead has been mentioned for the context recovery operation.

Finally, utilizing emulation-based fault injection tool in [30], ten thousand SEUs were randomly injected into the processor registers, and in particular into the pipeline registers. From these experiments, it was observed that 84% of the injected faults are either effectless or detected and corrected properly, while 15% of the injected faults could not be corrected, therefore causing illegal instruction trap in the processor. The remaining 1% of the injected faults, on the other hand, were also corrected by the system through configuration memory scrubbing, as they caused persistent errors. Note that no irradiation experiments have been conducted for the given design flow.

In [31], an on-board data-handling computer incorporating a LEON3 processor core was considered as implemented on a Xilinx Virtex-II device for a real space mission. The computer system was protected from radiation-induced faults by the design approach mentioned above, and runs a periodic task of acquiring a stream of data, applying the Kalman filter on that data, and then sending the filtered data to the platform computer. The period of the task is reported to be 1.25ms for the operational frequency of 40 MHz. An analysis was performed to assess the effects of heavy ions on the system using two scenarios – that is, the worst-day scenario (related to solar flare events) and the background Galactic Cosmic Radiation (GCR) scenario were considered according to the CREME96 predictions [32], for which the corresponding user-memory and configuration-memory SEU rates were reported.

In this analysis, it was estimated about one upset in the configuration memory of the processor core would occur every 14 seconds and every 1 hour in the worst-day scenario and the background GCR scenario, respectively, thus causing errors in the the program execution. This implies that once every 14s or 1hr depending on the scenario, the monitor module described above would detect a persistent error, and therefore trigger the FPGA reconfiguration and processor reset, which would make the data-handling computer unavailable for 500ms at the given configuration clock frequency every 14s or 1hr. This unavailability would cost the loss of just 3.6% and 0.01% of the data acquired and processed in the worst-day scenario and the background GCR scenario, respectively.

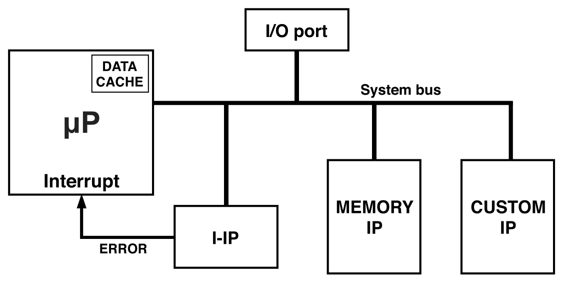


FIGURE 12. Fault-tolerant system with I-IP [33].

Furthermore, due to the SEUs in the user memory, it is estimated that 0.001% of the total amount of data gathered and processed would be lost in the worst-day scenario. Therefore, it is analytically proven that the impact of SEUs in the configuration and user memories is negligible when using the above fault mitigation approach for LEON3 processors in a SoPC.

D. SPATIAL, SOFTWARE AND INFORMATION REDUNDANCY BASED SOLUTIONS

In [33], authors extend their previous technique [34] of fault detection for SoCs in order to incorporate fault correction as well, where SEUs in the data memories and caches are detected and corrected with an approach comprising source code modifications which introduce redundant code and information, i.e. software and information redundancy, respectively. The coherence between redundant information is subsequently verified by a redundantly included, special-purpose Infrastructure IP core (I-IP) [35] during the execution, which is an instance of spatial redundancy. Therefore, this approach proposes high-level application software alterations and a hardware core incorporation with no modification to the rest of the hardware system. The authors claim that the this approach provides the same fault detection/correction capabilities as purely software-based fault-tolerance techniques [8], such as the one reported in [36], with the merits of diminished code and performance overheads.

Fault detection, fault localization and fault correction features are supported in the fault-tolerance technique of [33]. Fault detection is achieved by duplicating variables on which same duplicated instructions are applied. A coherency check between the two replicas of each variable is then performed by the I-IP, connected to the system bus as shown in Fig. 12, which monitors all bus transactions and detects any discrepancies, i.e. faults. Furthermore, in case of fault detection, fault localization is performed by deploying high-level macros in the application code. These macros submit values of a specific set of duplicated variables to the I-IP which subsequently calculates the corresponding execution checksum value [36], and then compares it with the reference checksum value to identify which set of variables is corrupted. The internal structure of the I-IP is shown in Fig. 13.

Moreover, fault correction is provided via a system-level recovery scheme triggered by an interrupt request from the I-IP once a fault is detected. Invoked interrupt service routine

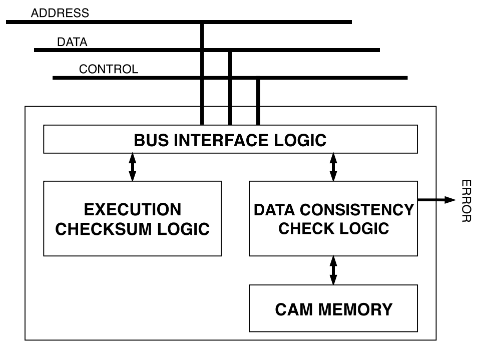


FIGURE 13. I-IP block design [33].

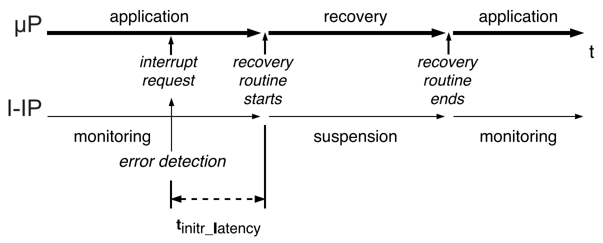


FIGURE 14. Recovery procedure timing diagram [33].

then corrects the corrupted set of variables using the presumably fault-free values of the other set, as depicted by the timing diagram in Fig. 14. Note that the proposed fault-tolerance technique does not provide resilience to control flow errors as the special-purpose registers within the processor, most importantly program counter (PC), and program memory/cache are not protected in any way.

In order to assess the memory and performance overheads, as well the fault tolerance capacity, of the proposed approach, four benchmark programs from the EEMBC AutoBench Performance Benchmark Suite [37] have been chosen to be hardened against soft errors through source code modifications explained above. These benchmarks were respectively related to the Matrix (MTX) multiplication of 10×10 matrices, fifth-Order elliptical (ELPF) filtering over a set of 16 samples and Lempel-Ziv-Welch (LZW) algorithm compressing character strings by replacing them with single codes. All of these programs were cross-compiled to run on a LEON3-based SoC armed with the proposed I-IP. However, the implementation platform for the SoC has not been mentioned in [33].

In the results, it has been shown that the hardware area overhead caused by the I-IP is less than five percent of the total area occupied by the LEON3 processor, whereas the proposed approach exhibits a code memory overhead which is up to 5× smaller than the purely software-based technique in [36] with a bonus of no additional data memory overhead. Furthermore, for all considered benchmark programs, the proposed technique reduces the performance overhead, e.g. by 4.2× for the ELPF case, with respect to [36].

Finally, emulation-based fault injection campaigns targeting data memories and caches were performed for each of the three benchmark programs using the same testing environment in [38], so as to evaluate the fault tolerance performance. In these experiments, although most of the faults have not exhibited themselves, the ones which managed to propagate in the system have been detected and corrected by the utilized technique for all cases, thus avoiding wrong results at the outputs. However, for the LZW case, some exceptions have occurred in the processor due to the injected faults causing corruptions in memory addresses within pointers. Remarkably, no radiation experiments have been conducted to assess the actual fault tolerance performance of the proposed methodology.

Lindoso *et al.* [39] has proposed and implemented a hybrid fault-tolerant technique for a LEON3 soft-core processor implemented in a low-end FPGA (i.e. Xilinx Artix-7), and then evaluated its error detection capabilities. The proposed solution combines the use of error detection and correction codes for memories, a module to detect control-flow errors, software-based techniques to detect data errors and configuration memory scrubbing with repair to avoid error accumulation. Therefore, it fits in the spatial, software and information redundancy based solutions category.

Authors have hardened RAM and caches of the system through the implementation of an unspecified Single Error Correction/Double Error Detection (SEC/DED) scheme in which single errors are corrected without issuing any notice, while double errors are merely detected and reported. Furthermore, the register file of the processor was constructed using duplicated dual-port BRAMs where single error detection was achieved through output comparisons; there is no explanation in the paper as to why correction mechanisms applied to RAM and caches were not employed here as well. On the other hand, to harden the processor control-flow, a Hardware Monitor (HM) was added to the system architecture which observes the instruction flow at the beginning, i.e. at the fetch stage, through the memory bus, and at the end of the pipeline path, i.e. after execution, through the trace interface. The HM module has resulted in an area overhead of 11% and 21% for LUTs and FFs, respectively, with respect to the baseline implementation. Fig. 15 depicts how the HM collects information from different pipeline stages for evaluation, and raises an error signal if they do not match.

Furthermore, the proposal tackles data flow errors with a software-based technique where all variables are duplicated

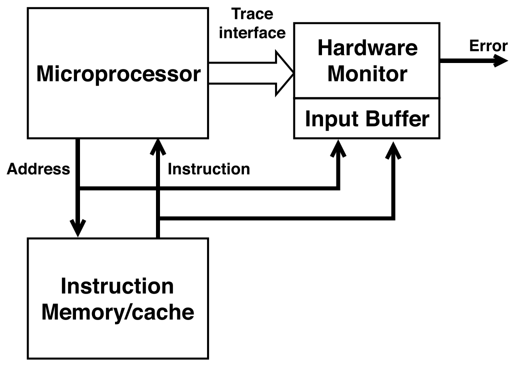


FIGURE 15. Block diagram of the hardware monitor [39].

to enable consistency checks, whenever any variable is modified or a procedure is called, to detect data errors. A control-flow duplication technique, explained in [40], was also deployed at the assembly instruction level to further enhance error detection. Finally, configuration memory scrubbing was applied through the instantiation of the Xilinx Soft Error Mitigation (SEM) IP core [41] in the system for detection and correction of configuration bit upsets. Note that the scrubbing module has caused an overhead of 16% and 30% in terms of LUTs and FFs, respectively.

When comparing each technique in a fault insertion campaign, authors have shown that the HM has the highest single contribution (i.e. 57.1% on average) which is about 1.6 times more effective than the software hardening (i.e. 35.3% on average) alone. Using all the above techniques together achieves an error detection rate of 96%. Moreover, neutron radiation experiments were carried out to compare and evaluate three different versions of the proposed approach, i.e. (i) one with configuration memory scrubbing and SEC in RAMs/caches only; (ii) one with configuration memory scrubbing, SEC/DED in RAMs/caches and duplicated register file only and (iii) one with all four aforementioned techniques. Results show that each version have failed to detect 95, 67 and 23 errors, respectively. Although these experiments have resulted in a high error detection rate, it is hard to put the test results into perspective because authors have included neither exact execution times nor the mean time between failures or undetected errors.

V. CONCLUSION

Various techniques for protecting the configuration and user memories of LEON3 soft processors on SRAM-based FPGAs were thoroughly discussed in this survey through a number of prominent research papers, all of which employ one or more forms of redundancy, such as spatial redundancy, temporal redundancy, software redundancy or information redundancy in order to develop resilience against transient faults induced by energetic particles or electromagnetic waves striking the semiconductor substrate of transistors in radiation environments. Although LEON3 was the focus of this study, fault-tolerant techniques and methods described will equally benefit other soft-core processors implemented in SRAM-based FPGAs. We envisage that choosing the right mix of redundancy to develop a fault tolerance solution for any given soft-core processor has been facilitated for the reader now that most prominent existing solutions have been studied in this survey.

Works presented in this survey have relied either on a single or multiple existent techniques which were statically mapped to solve the given problem. However, there is still an open research challenge on how to dynamically, i.e.at run-time, adjust the system to choose and apply the most appropriate set of techniques at a given point in time. One good starting point which can be exploited and expanded is presented in [19], where multiple and larger reconfigurable areas could be used for replacement units. Furthermore, instead of focusing on just one particular module or technique, we envisage the system can decide by itself which technique/module is the most viable one based on the current error rate, error types, etc.

REFERENCES

[1] *NDA*. Accessed: Oct. 25, 2019. [Online]. Available: https://www.gov.uk/government/publications/nuclear-provision-explaining-the-cost-of-cleaning-up-britains-nuclear-legacy/nuclear-provision-explaining-the-cost-of-cleaning-up-britains-nuclear-legacy

[2] *Fukushima Daiichi Nuclear Power Plant Accident*. Accessed: Oct. 28, 2019. [Online]. Available: https://www.scmp.com/news/asia/east-asia/article/2077394/dying-robots-and-failing-hope-fukushima-clean-falters-six-years

[3] R. Baumann, "Soft errors in advanced computer systems," *IEEE Des. Test. Comput.*, vol. 22, no. 3, pp. 258–266, May 2005.

[4] T. Li, J. A. Ambrose, R. Ragel, and S. Parameswaran, "Processor design for soft errors: Challenges and state of the art," *ACM Comput. Surv.*, vol. 49, no. 3, p. 57, Nov. 2016, doi: 10.1145/2996357.

[5] S. Hauck and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. San Francisco, CA, USA: Morgan Kaufmann, 2007.

[6] B. Johnson, "Fault-tolerant microprocessor-based systems," *IEEE Micro*, vol. 4, no. 6, pp. 6–21, Dec. 1984.

[7] A. Aviziens, "Fault-tolerant systems," *IEEE Trans. Comput.*, vol. C-25, no. 12, pp. 1304–1312, Dec. 1976.

[8] D. K. Pradhan, Ed., *Fault-Tolerant Computer System Design*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[9] T. Nidhin, A. Bhattacharyya, R. Behera, T. Jayanthi, and K. Velusamy, "Understanding radiation effects in SRAM-based field programmable gate arrays for implementing instrumentation and control systems of nuclear power plants," *Nucl. Eng. Technol.*, vol. 49, no. 8, pp. 1589–1599, Dec. 2017.

[10] M. Wirthlin, "High-reliability FPGA-based systems: Space, high-energy physics, and beyond," *Proc. IEEE*, vol. 103, no. 3, pp. 379–389, Mar. 2015.

[11] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of radiation effects in SRAM-based FPGAs for space applications," *ACM Comput. Surv.*, vol. 47, no. 2, p. 37, Jan. 2015, doi: 10.1145/2671181.

[12] J. Gaisler, "A portable and fault-tolerant microprocessor based on the SPARC v8 architecture," in *Proc. Int. Conf. Depend. Syst. Netw.*, Jun. 2003, pp. 409–415.

[13] J. Gaisler and E. Catovic, "Multi-core processor based on LEON3-FT IP core (LEON3-FT-MP)," in *Proc. Conf. Data Syst. Aerosp.*, May 2006, pp. 1–5.

[14] M. Makni, M. Baklouti, S. Niar, M. W. Jmal, and M. Abid, "A comparison and performance evaluation of FPGA soft-cores for embedded multi-core systems," in *Proc. 11th Int. Design Test Symp. (IDT)*, Dec. 2016, pp. 154–159.

[15] *GRLIB IP Core User's Manual*. Accessed: Jun. 7, 2019. [Online]. Available: https://www.gaisler.com/

[16] L. Nabil, B. Jaafar, and S. B. Saoud, "Embedded microprocessor performance evaluation case study of the LEON3 processor," *J. Eng. Sci. Technol.*, vol. 7, no. 5, pp. 574–588, Oct. 2012.

[17] F. Ferlini, F. A. Da Silva, E. A. Bezerra, and D. V. Lettnin, "Non-intrusive fault tolerance in soft processors through circuit duplication," in *Proc. 13th Latin Amer. Test Workshop (LATW)*, Apr. 2012, pp. 1–6.

[18] C. Carmichael, "Triple modular redundancy design techniques for virtex FPGAs," Xilinx, San Jose, CA, USA, Appl. Notes, XAPP197 (v1.0.1), Jan. 2006, pp. 1–37.

[19] S. Di Carlo, A. Miele, P. Prinetto, and A. Trapanese, "Microprocessor fault-tolerance via on-the-fly partial reconfiguration," in *Proc. 15th IEEE Eur.Test Symp.*, May 2010, pp. 201–206.

[20] A. Bouajila, J. Zeppenfeld, W. Stechele, and A. Herkersdorf, "An architecture and an FPGA prototype of a reliable processor pipeline towards multiple soft-and timing errors," in *Proc. 14th IEEE Int. Symp. Design Diagnostics Electron. Circuits Syst.*, Apr. 2011, pp. 225–230.

[21] A. M. Keller and M. J. Wirthlin, "Benefits of complementary SEU mitigation for the LEON3 soft processor on SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 1, pp. 519–528, Jan. 2017.

[22] M. J. Wirthlin, A. M. Keller, C. Mccloskey, P. Ridd, D. Lee, and J. Draper, "SEU mitigation and validation of the LEON3 soft processor using triple modular redundancy for space processing," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2016, pp. 205–214, doi: 10.1145/2847263.2847278.

[23] A. Sari and M. Psarakis, "Scrubbing-based SEU mitigation approach for systems-on-programmable-chips," in *Proc. Int. Conf. Field-Program. Technol.*, Dec. 2011, pp. 1–8.

[24] A. Sari, M. Psarakis, and D. Gizopoulos, "Combining checkpointing and scrubbing in FPGA-based real-time systems," in *Proc. IEEE 31st VLSI Test Symp. (VTS)*, Apr. 2013, pp. 1–6.

[25] N. Bowen and D. Pradham, "Processor- and memory-based checkpoint and rollback recovery," *Computer*, vol. 26, no. 2, pp. 22–31, Feb. 1993.

[26] G.-H. Asadi and M. Tahoori, "Soft error mitigation for SRAM-based FPGAs," in *Proc. 23rd IEEE VLSI Test Symp. (VTS)*, Jul. 2005, pp. 207–212.

[27] M. Reorda, M. Violante, C. Meinhardt, and R. Reis, "A low-cost SEE mitigation solution for soft-processors embedded in systems on pogrammable chips," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Apr. 2009, pp. 352–357.

[28] M. Violante, C. Meinhardt, R. Reis, and M. S. Reorda, "A low-cost solution for deploying processor cores in harsh environments," *IEEE Trans. Ind. Electron.*, vol. 58, no. 7, pp. 2617–2626, Jul. 2011.

[29] D. Pradhan and N. Vaidya, "Roll-forward and rollback recovery: Performance-reliability trade-off," *IEEE Trans. Comput.*, vol. 46, no. 3, pp. 372–378, Mar. 1997.

[30] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Reorda, and M. Violante, "Exploiting circuit emulation for fast hardness evaluation," *IEEE Trans. Nucl. Sci.*, vol. 48, no. 6, pp. 2210–2216, Dec. 2001.

[31] M. S. Reo, M. Violante, C. Meinhardt, and R. Reis, "An on-board data-handling computer for deep-space exploration built using commercial-off-the-shelf SRAM-based FPGAs," in *Proc. 24th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Oct. 2009, pp. 254–262.

[32] *Cosmic Ray Effects on Micro Electronics*. Accessed: Jul. 9, 2019. [Online]. Available: https://creme.isde.vanderbilt.edu

[33] P. Bernardi, L. B. Poehls, M. Grosso, and M. S. Reorda, "A hybrid approach for detection and correction of transient faults in SoCs," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 4, pp. 439–445, Oct. 2010.

[34] P. Bernardi, L. Bolzani, M. Rebaudengo, M. Reorda, F. Vargas, and M. Violante, "A new hybrid fault detection technique for systems-on-a-chip," *IEEE Trans. Comput.*, vol. 55, no. 2, pp. 185–198, Feb. 2006.

[35] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.

[36] M. Rebaudengo, M. S. Reorda, and M. Violante, "A new approach to software-implemented fault tolerance," *J. Electron. Test.*, vol. 20, no. 4, pp. 433–437, Aug. 2004, doi: 10.1023/b:jett.0000039610.30724.b2.

[37] *EEMBC AutoBench Performance Benchmark Suite*. Accessed: Aug. 17, 2019. [Online]. Available: https://www.eembc.org/autobench

[38] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante, "An FPGA-based approach for speeding-up fault injection campaigns on safety-critical circuits," *J. Electron. Test., Theory Appl.*, vol. 18, no. 3, pp. 261–271, Jun. 2002, doi: 10.1023/A:1015079004512.

[39] A. Lindoso, L. Entrena, M. Garcia-Valderas, and L. Parra, "A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 1, pp. 374–381, Jan. 2017.

[40] J. R. Azambuja, S. Pagliarini, M. Altieri, F. L. Kastensmidt, M. Hubner, J. Becker, G. Foucard, and R. Velazco, "A fault tolerant approach to detect transient faults in microprocessors based on a non-intrusive reconfigurable hardware," *IEEE Trans. Nucl. Sci.*, vol. 59, no. 4, pp. 1117–1124, Aug. 2012.

[41] "Soft error mitigation controller v4.1 product guide," Xilinx, San Jose, CA, USA, White Paper PG036, 2014.

SERVER KASAP (Member, IEEE) received the B.Sc. degree (Hons.) in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2006, and the M.Sc. (Hons.) and Ph.D. degrees in electronic engineering from the System Level Integration Research Group, The University of Edinburgh, Edinburgh, U.K., in 2007 and 2010, respectively. He is currently a Senior Research Officer with the University of Essex, Colchester, U.K. His current research interests include reliability and fault tolerance, FPGA hardware design and implementation for digital signal processing applications, and high-performance scientific computing in general.

EDUARDO WEBER WÄCHTER received the B.Eng. degree in computer engineer from the State University of Rio Grande do Sul, in 2009, and the Ph.D. degree in computer science from the Pontifical Catholic University of Rio Grande do Sul (PUCRS), in 2015. He is currently a Senior Research Officer with the University of Essex, U.K. His research interests include many core and NoC systems, and reliability and fault tolerance.

XIAOJUN ZHAI (Member, IEEE) received the Ph.D. degree from the University of Hertfordshire, U.K., in 2013. He is currently a Lecturer with the Embedded Intelligent Systems Laboratory, University of Essex. He has authored/coauthored over 60 scientific article in international journals and conference proceedings. His research interests mainly include the design and implementation of the digital image and signal processing algorithms, custom computing using FPGAs, embedded systems, and hardware/software co-design. He is a member of BCS, and a Fellow of HEA.

SHOAIB EHSAN (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2003, and the Ph.D. degree in computing and electronic systems (with specialization in computer vision) from the University of Essex, Colchester, U.K., in 2012. He has an extensive industrial and academic experience in the areas of embedded systems, embedded software design, computer vision, and image processing. His current research interests are in intrusion detection for embedded systems, local feature detection and description techniques, and image feature matching and performance analysis of vision systems. He was a recipient of the University of Essex Post Graduate Research Scholarship, the Overseas Research Student Scholarship, and the prestigious Sullivan Doctoral Thesis Prize awarded annually by the British Machine Vision Association.

KLAUS MCDONALD-MAIER (Senior Member, IEEE) is currently the Head of the Embedded and Intelligent Systems Laboratory, University of Essex, Colchester, U.K. He is also the Chief Scientist with UltraSoC Technologies Ltd., the CEO of Metrarc Ltd., and a Visiting Professor with the University of Kent. His current research interests include embedded systems and system-on-chip design, security, development support and technology, parallel and energy-efficient architectures, computer vision, data analytics, and the application of soft computing and image processing techniques for real-world problems. He is a member of VDE and a Fellow of the IET.