Microelectronics Reliability 146 (2023) 115010

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

Review paper

Understanding fault-tolerance vulnerabilities in advanced SoC FPGAs for critical applications

Natalia Cherezova [a,*], Konstantin Shibin [a,b], Maksim Jenihhin [a], Artur Jutman [b]

[a] Department of Computer Systems, Tallinn University of Technology, Tallinn, Estonia
[b] Testonica Lab, Tallinn, Estonia

ARTICLE INFO

Keywords:
Soft errors
Fault effects
Fault-tolerance
Cross-layer reliability
SoC FPGA

ABSTRACT

The emergence of heterogeneous FPGA-based SoCs and their growing complexity fueled by the introduction of various accelerators bring the reliability aspect of these systems to the front. The concern is particularly important for critical applications, such as safety-critical autonomous vehicles, real-time systems, space missions, health, and security cyber–physical systems. This paper presents an analysis of the most common types of errors caused by faults in different components and identifies the most critical and vulnerable components based on the literature survey. The study looks into vendor-provided and user-space reliability solutions. The paper highlights the existing fault-tolerance gaps in the modern SoC FPGAs and outlines a vision for future solutions.

1. Introduction

Modern heterogeneous SoC FPGAs, that provide software programmability through scalar and vector processors and hardware programmability through FPGA logic, are getting more complex and advanced, integrating more components on a single chip. However, their dense structure is also increasingly susceptible to various hardware faults. Their reliability assessment has certain challenges: simulation and testing are difficult due to the complexity of the design, and faults in one core can induce errors in other cores [1].

Hardware faults can be caused by various reasons: radiation (soft errors), aging, manufacturing process variations, manufacturing defects, and early-life failures. Nonetheless, soft errors are considered dominant for today's digital systems, hence, the analysis in this paper is focused on them.

The most common type of soft errors is Single Event Upset (SEU), a change of the logic state of the sequential element. Less common but more critical is Single Event Functional Interrupt (SEFI) — a soft error that causes a functional interrupt in circuits, interfaces, or entire chips. Another type of soft error is Single Event Transient (SET), a current or voltage spike in a signal. SET is transitory in nature, however, if captured by a sequential element, it can become an SEU.

The heterogeneous nature of SoC FPGAs is the reason that faults in different components manifest differently and require tailored fault-tolerance solutions, which makes the reliability assurance of the chip a challenging task.

The contributions of this paper are:

- a taxonomy of an advanced SoC FPGA structure;
- an analysis of fault effects caused by faults in the different components of the structure;
- a review of vendor-provided fault-tolerance mechanisms and analysis of fault-tolerance vulnerabilities and gaps;
- a review of complementary user-space fault-tolerance solutions.

The rest of the article is organized in the following way. The typical structure of the modern SoC FPGA and a discussion on the most common types of errors caused by faults in different components are presented in Section 2. The most critical and vulnerable components are identified there as well. Reliability solutions provided by vendors are given in Section 3. Section 4 highlights the fault tolerance gaps. User-space solutions are described in Section 5. Section 6 presents discussions and conclusions, and Section 7 summarizes the paper.

2. Faults and fault effects in SoC FPGAs

This section presents a typical structure of modern FPGA-based SoC along with faults and fault effects characterizing each component.

2.1. Typical structure of modern FPGA-based SoC

The structure is based on the latest series from AMD-Xilinx, Intel, and Microchip: Versal ACAP (Adaptive Compute Acceleration Platform) [2] and Zynq UltraScale+ [3] (AMD-Xilinx); Agilex [4] and

* Corresponding author.
E-mail address: natalia.cherezova@taltech.ee (N. Cherezova).

https://doi.org/10.1016/j.microrel.2023.115010
Received 28 June 2022; Received in revised form 11 April 2023; Accepted 8 May 2023
Available online 18 May 2023

Stratix 10 [5] SoC families (Intel), and PolarFire SoC [6] (Microchip). A modern FPGA-based SoC includes multi-core processors (ARM Cortex-A and Cortex-R cores in AMD-Xilinx and Intel devices, and RISC-V cores in Microchip SoC), FPGA logic, hard peripheral controllers, embedded on-chip memories, and system infrastructure represented by non-functional utility resources, such as chip monitors, ECC controllers, test facilities, etc. The most notable among them is a separate processor that acts as a device manager. Device manager, called Platform Management Unit (PMU) in AMD-Xilinx devices, Secure Device Manager (SDM) in Intel, and monitor core in Microchip, sets the root-of-trust, configures the device after the power-up, and monitors its state during the lifetime. Some platforms have accelerators, like embedded GPU for video rendering or custom engines for AI and ML tasks processing (Zynq UltraScale+ and Versal), and specialized modules, like radio frequency (RF) data converters and video processing units (Zynq UltraScale+). Versal ACAP represents the new generation of programmable SoCs from AMD-Xilinx and has the most complex structure. The processing system, FPGA logic, AI Engines (custom blocks for fast data processing), and peripheral controllers are connected through Programmable Network-on-Chip (PNoC). One can assume that the next FPGA-based SoCs, generally, follow the Versal model.

The typical structure described is presented in Fig. 1. The presented structure can be considered as a super-set. More complex system-on-chips include more elements from the set, and less complex ones include fewer, but they still stay within the given super-set.

2.2. Faults and fault effects in different components

Below is the analysis based on the literature survey of the most common types of errors caused by faults in different components and the most critical and vulnerable elements in each module. Table 1 summarizes fault effects at the module and system levels. System failures are divided into three classes: system crash (SC), application crash (AC), and application silent data corruption (ASDC). The most common system-level failure is ASDC. AC can be caused by control flow errors, and SC might be caused by errors related to the SoC configuration or control circuitry.

Processing system (PS). Structurally faults for a processor were organized into three large groups: faults in registers (including general-purpose and special-purpose ones), caches, and data processing units (ALU/FPU) [7–12]. It can be seen that faults in different components affect processors in two ways: silent data corruption (SDC) and control flow errors that can lead to functional interrupts or crashes. Functional interrupts can be caused by errors in control or special purpose registers, errors in addressing, and errors in instruction (wrong instruction, wrong operand, or wrong immediate) [8]. While functional interrupts require a reset and result in the system being unavailable during the restart, SDC does not disable the system but might be of great concern for a user, as they corrupt results without giving any indications that something went wrong. The most critical elements are registers (particularly special-purpose ones) and caches [7–13] since faults in those components directly affect the data being processed and can introduce control flow errors. Additionally, studies show that processor cores integrated into SoC platforms have higher rates of crashes and functional interrupts compared to standalone processors [14].

FPGA. Structurally, faults in FPGA can be organized into three groups as well: faults in configuration memory, user memory, and control-related logic [15]. Unlike processors, the functionality of FPGA is not hardened and is defined by the values loaded into configuration memory (called CRAM). Therefore, the main fault effect for them is the change in circuit functionality, which might result in silent data corruption as well as application or even system crash.

Configuration memory consists of routing elements and logic resources, such as LUTs (Look Up Tables) and control bits.

Routing resources can be local (inside logic blocks) or global (between logic blocks). Local routing is represented by multiplexers (MUX) controlling inputs and outputs of the logic block and LUTs. Logic blocks are connected through switch boxes and connection boxes by the means of Programmable Interconnect Points (PIPs). PIP may be represented by a simple pass transistor controlled by one configuration bit or a complex structure created using several multiplexers controlled by a group of configuration bits.

For PIP, the following fault models are defined: Open, Conflict, Input Antenna, Output Antenna, and Bridge [16]. Open fault happens when a previously used connection is disabled. Conflict is characterized by a new connection being added between a used input node and a used output node. Input Antenna is a new connection between an unused input node and a used output node. Output Antenna is a new connection between an unused input node and a used output node. Bridge happens when a previously used connection is disabled, and an unrelated input is connected to the output from the disabled connection.

Along with the change in circuit functionality, SEU in routing configuration bits that create an undesired bridge between a signal and an unused wire adds an extra parasitic capacitance (and thus an extra delay) to the signal path. The impact is cumulative and can lead to delays that are significant enough to cause delay faults [17].

Routing faults are the most common and the most critical for FPGAs since the routing structure occupies a large portion of the FPGA area and configuration bits [18]. Moreover, the most failure-sensitive routing elements are local MUXes [19,20].

Faults in logic include LUT values change and control bits changes. LUTs are used to hold the data (when used as distributed RAM or shift register) and implement data processing or Finite State Machine (FSM) to control the workflow. Therefore, faults in LUT values can cause data corruption as well as control flow errors. However, whether it produces an error or not depends on the inputs to the circuit and which part of the logic function is impacted.

On the contrary, changes in control bits generally cause errors for all, or almost all, possible circuit inputs. Control bits define LUT mode (whether it is used as a LUT, distributed RAM, or a shift register), configure storage elements as flip-flops or latches, and define Input/Output Block (IOB) parameters, such as electrical standard, mode (input, output, bi-directional), termination (pull-up or pull-down resistors), signal drive strength, optional storage element, input delay, and a slew rate. Faults in IOB parameters can create unintended outputs [21] and additional delays [22–24].

User memory includes embedded RAM and flip-flops in the logic blocks. It stores the data produced by the application. Faults in user memory are straightforward and cause silent data corruption. While faults in embedded RAM data remain until a new value is stored there or the fault is corrected, original flip-flop values might be restored in the next clock cycle. Faults in user memory are considered less critical than faults in the configuration memory. The former ones do not affect the circuit functionality and can be fixed without device reconfiguration.

Faults in control logic include faults in reset circuitry and configuration ports, such as JTAG TAP (Test Access Port) controller, ICAP (Internal Configuration Access Port), PCAP (Processor Configuration Access Port), and SelectMAP interface. SEU in reset circuitry can clear out the device functionality or all of the flip-flops. Faults in the JTAG TAP controller can damage the FPGA or other system devices as well as cause a loss of control [25]. Faults within the programmable SelectMAP interface configuration pins result in a malfunctioning interface: the corrupted interface can return bad values during a read operation and corrupt the configuration during a write operation [15]. Faults in control logic have serious consequences and require device reconfiguration, though, in practice, their rates are very low [15].

Moreover, in SoC, the FPGA part is proved to be more prone to soft errors [8,26,27], while processors and peripherals have more vanished errors.

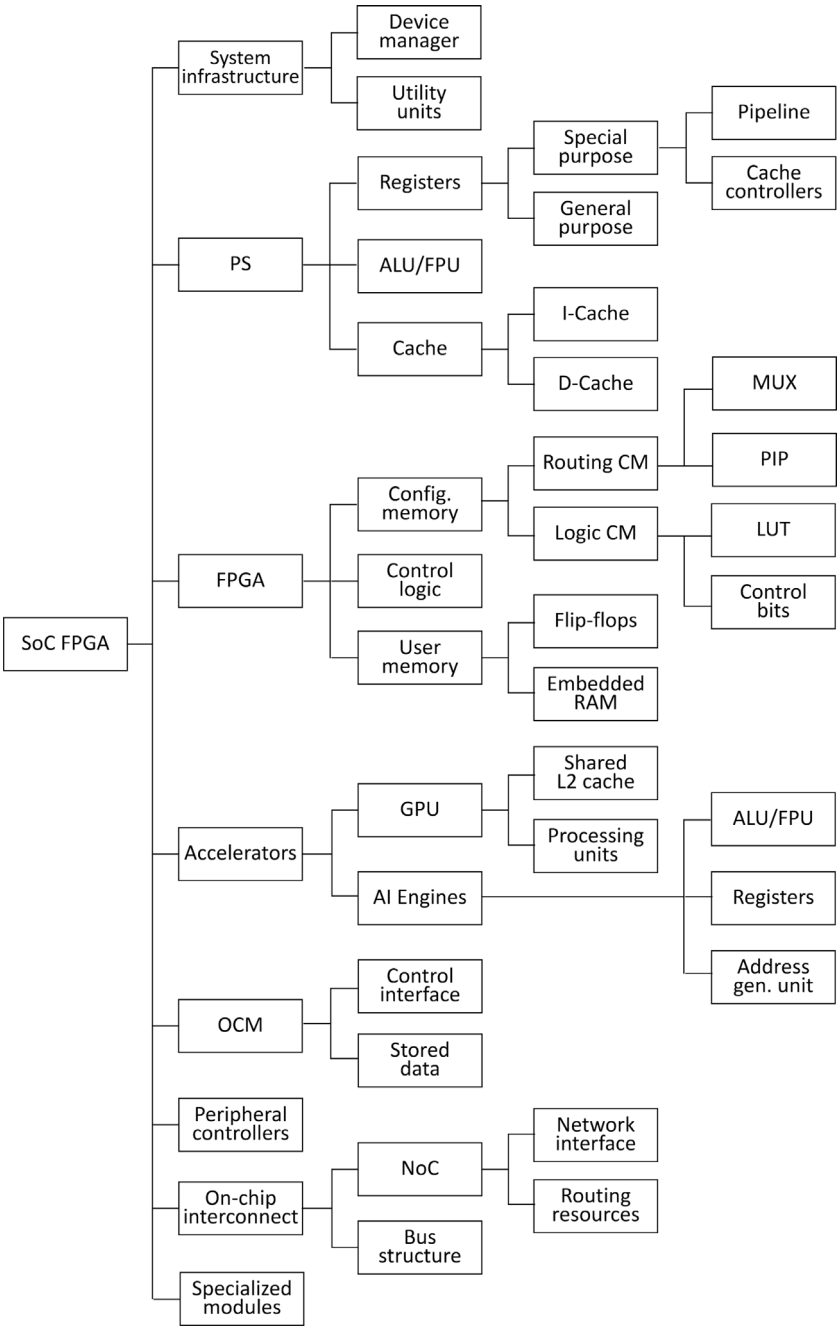

Fig. 1. Typical structure of modern FPGA-based SoC.

Accelerators. Two types of accelerators are considered in this work: embedded GPU and custom AI Engines suggested by AMD-Xilinx. Versal AI Engines [28] consist of two processors, for scalar and vector arithmetic, and thus their structural faults are similar to the processor faults. Faults in instructions can cause data and control flow errors, and faults in registers and in data processing units can cause silent data corruption. AI Engines do not have cache memory, though they do have shared memory for storing the data.

For GPU, faults in global buffers (shared L2 cache) can be considered the most critical, since they may spread in multiple locations and can cause many erroneous outputs [29–31].

Faults in accelerators are not expected to cause crashes but rather produce wrong outputs. While data errors in machine learning applications might have serious consequences, erroneous pixels in a video frame just degrade the user experience.

Peripherals and memory controllers. Experiments show that faults in address and control registers cause many data errors and interrupts [8, 32]. However, there are only a few works that explore faults in so-called "uncore" components, such as peripherals, on-chip memories, etc. [33–35]. However, loss of configuration in peripheral controllers might result in a system crash and require device reconfiguration.

On-chip interconnect. Different components in heterogeneous SoCs are usually connected through a bus-based communication architecture. However, Versal ACAP represents an interesting case that additionally utilizes network-on-chip (NoC). Versal NoC architecture is similar to a typical NoC structure: network interface units connect processing elements to the NoC, and routers perform the transportation of the data through the NoC. Network interface units are represented by NoC Master Units (NMU) and NoC Slave Units (NSU) that denote ingress and egress blocks accordingly. Routers are represented by NoC

Table 1
Faults and fault effects in SoC FPGA.

Fault location		Error within module	System level failures		
			SC	AC	ASDC
Processor					
Register file	General-purpose registers	Silent data corruption, control flow errors	×	•	•
	Special purpose registers	Control flow errors	×	•	∘
Cache	Instruction cache	Control flow errors	×	•	∘
	Data cache	Silent data corruption	×	∘	•
Data processing units (ALU/FPU)		Silent data corruption	×	∘	•
FPGA					
Configuration memory	Routing elements	Functional errors (change in circuit functionality)	•	•	•
	Logic resources				
User memory	Flip-flops	Silent data corruption	×	∘	•
	Embedded RAM				
Control logic	Reset	Configuration loss, functional interrupts	•	•	•
	JTAG TAP controller				
	SelectMAP interface				
Accelerators					
GPU	L2 shared cache	Silent data corruption, control flow errors	×	×	∘
	Processor units				
AI Engine	Register file	Silent data corruption, control flow errors	×	×	•
	Data processing units				
	Address generation unit				
OCM and peripherals					
OCM	Control interface	Silent data corruption, control flow errors	∘	•	•
	Stored data		×	∘	•
Peripheral controllers		Control flow errors	•	∘	∘
On-chip interconnect					
NoC	Processing units	Silent data corruption, configuration loss	•	•	•
	Connections				
Bus structure		Silent data corruption, configuration loss	•	•	•

SC — system crash. AC — application crash. ASDC — application silent data corruption.
×— not expected. ∘ — possible. • — highly likely.

Packet Switch (NPS). It is a crossbar switch that determines the direction of the packet based on the special routing tables. Though Versal NoC is programmable, its configuration is statically defined during the design time, therefore, it includes an additional unit called NoC Peripheral Interconnect (NPI). NPI is an internal register programming interconnect of the NoC that controls and configures NMU, NSU, and NPS. The NPI programs NoC registers that define the aforementioned routing tables [36].

In the classic NoCs, faults in interconnects are considered dominant [37,38]. Additionally, since the configuration of Versal PNoC is programmable, faults in routing components and NPI registers can have serious consequences. They can lead to a loss of configuration and, therefore, to an application or a system crash.

For the AMBA AHB system bus, experiments in [39] show that SDC is the most common error, however, faults in the address bus have a high probability to cause an application or system crash.

3. Vendor-provided fault tolerance mechanisms

The heterogeneous nature of modern SoC FPGAs, combining different components, requires a complex fault tolerance solution that covers the whole chip.

The main fault detection and fault tolerance mechanisms employed by vendors are the following:

- Device manager
- Modular redundancy (usually TMR)
- ECC, CRC, parity checking
- Data scrubbing
- Dual lock-step
- Memory protection units
- Built-in tests
- Watchdog timers

A comparison between fault detection and fault tolerance methods on different platforms is presented in Table 2.

Device manager. As already mentioned, all considered platforms have a dedicated processor or two acting as a device manager: Platform Management Unit (PMU) on AMD-Xilinx platforms, Secure Device Manager (SDM) on Intel platforms, and monitor core on Microchip PolarFire SoC. The device manager holds the root-of-trust, configures the device after the power-up, performs error management, and monitors its state during the lifetime to support the functional safety of the platform.

Triple-modular redundancy (TMR). On AMD-Xilinx and Intel platforms TMR is used to protect the device manager and critical state registers. Additionally, Intel and Microchip offer functionality to generate a triplicated design for FPGA logic. AMD-Xilinx offers similar functionality for Virtex family devices only. For modern platforms, it offers TMR MicroBlaze IP (soft processor).

Error correction code (ECC). ECC is the standard protection mechanism for memories. AMD-Xilinx, Intel, and Microchip have ECC (single-error correction, double error detection) on all processor memories, on-chip memories, FPGA embedded memories, and DDR4 controllers. ECC mechanism supports error injection functionality. On Intel platforms, USB, SD/MMC, EMAC, DMA, and NAND flash controllers have

Table 2
Vendor-provided fault tolerance mechanisms in different platforms.

Method	AMD-Xilinx		Intel		Microchip	
	Realization	What is protected	Realization	What is protected	Realization	What is protected
Device manager	Platform Management Unit (PMU)	System	Secure Device Manager (SDM)	System	RISC-V Monitor core	System
Modular redundancy	Hard TMR	Device manager + critical state registers	Hard TMR	Device manager + critical on-chip state machines	—	
	—		Wizard for TRM generation on FPGA logic	Design on FPGA logic	Wizard for TRM generation on FPGA logic	Design on FPGA logic
ECC	SECDED	Caches, OCM, FPGA embedded memories, DDR4 controller	SECDED/multi-bit correction	Caches, OCM, FPGA embedded memories, DDR4 controller, NAND flash controller	SECDED	Caches, FPGA embedded RAM, DDR3/4 controller
Lock-step	Dual lock-step	Cortex-R5F core	—		—	
Data scrubbing	XilSEM IP	FPGA CRAM	Intel EDC feature	FPGA CRAM	—	
Memory protection units	ARM TrustZone, XMPU, XPPU	Memories, peripherals	ARM TrustZone, SMMU	Memories	MMU, PMP	Memories
Built-in tests	LBIST	XMPU, lock-step, and ECC checkers	*(No info found)*		MBIST	L2 cache
	MBIST	Cortex-R5F core memories				
	STL	Memories, PMU, XMPU, XPPU, clocks, voltages, temperature				
Watchdog timers	Hardware watchdogs	Processing system	Hardware watchdogs	Processing system	Hardware watchdogs	Processing system

integrated ECC-protected memories. While most ECCs provide a single-error correction, double-error detection, for M20K blocks (internal 20 Kbit memories), Intel suggests single-error, double-adjacent-error, and triple-adjacent-error correction in a 32-bit word. However, it is reported that with the ECC feature enabled, M20K runs slower [40]. In order to decrease the possibility of multi-bit errors, all vendors use a memory interleaving technique.

Dual lock-step. AMD-Xilinx platforms include ARM Cortex-R5F dual-core processor that supports the lock-step functionality. In the lock-step mode, two cores perform the same task in parallel.

Data scrubbing. AMD-Xilinx and Intel, both suggest solutions for soft-errors correction in the FPGA configuration memory (CRAM) based on the background scan. Xilinx Soft Error Mitigation (XilSEM) is available as soft IP in the UltraScale+ series [41], however, in the Versal series, it is a part of PMU [42]. Triplicated PMU provides more protection for the module itself. XilSEM uses ECC and CRC for single-error correction and double-error detection. On Versal, XilSEM can additionally detect errors in NPI (NoC Peripheral Interconnect) registers that connect NoC with some modules using SHA techniques. XilSEM supports error injection functionality.

Intel Error Detection Circuit (EDC) with integrated ECC corrects single- and double-bit errors and detects up to 8 bits that fit in a rectangular box (8×1, 4×2, 1×8, 2×4). EDC supports error injection and priority scrubbing. Some portions of the design can be marked as high-priority sectors, those sectors are scanned more frequently. Additionally, there is a sensitivity processing and hierarchy tagging functionality that defines the criticality of the CRAM bit and suggests corrective actions based on the classification [43,44].

Since PolarFire SoC incorporates flash-based FPGA, which is more robust towards soft errors, there is no scrubbing solution suggested.

Memory protection units (MPU). While memory protection units are not explicitly designed for soft error protection, they can help with them as well. One of the main purposes of MPU is to restrict memory access of a process to some defined region. If there is an error in the address due to the SEU so that it no longer belongs to the allowed region, access is blocked, which might prevent SDC or control flow error.

AMD-Xilinx platforms have a Memory Protection Unit (XMPU) and Peripheral Protection Unit (XPPU) to protect on-chip memory blocks and peripheral programming registers [42]. Intel platforms have System Memory Management Unit (SMMU), and one of its functions is memory access permissions. PolarFire has Memory Protection Unit as well. Additionally, since AMD-Xilinx and Intel SoC include ARM processor cores, they are protected by ARM TrustZone.

Built-in tests. AMD-Xilinx provides LBIST for check logic such as XMPU, lock-step, and ECC checkers, and MBIST for Cortex-R5F core memories. Additionally, memories, lock-step functionality, device manager, XMPU, XPPU, clocks, voltages, and temperature can be checked by the dedicated software test library (STL) [42]. Microchip provides MBIST for L2 cache [45].

Watchdog timers. Used for detection of the hanged or stuck in an infinite loop process, watchdog timers are provided by all three considered vendors.

4. Analysis and conclusions about the gaps

4.1. Vendor-provided fault tolerance methods analysis

Fault tolerance methods described in the previous section have certain overheads, such as area, power consumption, or execution time. They also have functional limitations on what kind of errors they can detect or correct and how fast they can do it. Therefore, for this study, it is important to see how vendor-provided methods protect the system from soft errors and what parts of SoC FPGAs are left unprotected.

Device manager. Implementation of the device manager requires one or two additional processors, which results in power consumption overhead. Moreover, the device manager does not protect the system from soft errors.

Triple-modular redundancy. TMR is based on utilizing three copies of the same module working in parallel, which introduces more than 200% overhead in the area and power consumption. This might be critical for energy-restricted applications. Therefore, TMR is only used to protect the device manager which acts as a safety island.

ECC. When enabled, ECC requires additional bits to store the codes, which results in 13 and 22% overhead for 32-bit and 64-bit data accordingly. In most cases, it can only correct single-bit errors and detect double-bit errors. Multi-bit correction and detection versions are possible but might slow down the memory. Since ECC is a memory protection technique, it mostly prevents SDC errors, however, in the case of Instruction Cache, it might prevent control flow errors as well.

Data scrubbing. The scrubbing solution that scans FPGA configuration memory in the background brings power consumption overhead. It has a latency of several milliseconds, during which the device might operate incorrectly. Studies show that scrubbing does not protect the device against immediate failures since it cannot fix errors propagated from faults in configuration memory [46]. Its main functionality is to prevent the accumulation of faults. Besides, it is not intended for use in a high-irradiation environment, such as space. An experiment in [47] shows the significant decrease in neural network accuracy deployed on the AMD-Xilinx UltraScale+ platform under beam radiation testing even with XilSEM enabled.

Dual lock-step. When two cores are running in the dual lock-step mode, they perform the same tasks with one clock cycle shift, which means the system can perform fewer tasks than in the split mode. Furthermore, since lock-step is a fault detection technique, it requires additional time to rerun the task in case of an error. Additionally, lock-step functionality is not supported by every processor.

MPU. As it was mentioned before, the memory protection unit is not designed to handle faults, though it can detect faults in some cases and prevent them from propagating further.

Watchdog timers. As a symptom-based fault detection method, the watchdog timer can neither isolate the fault nor correct it. The watchdog timer can either create an interrupt or restart the processor core.

4.2. Gaps in vendor-provided protection

Based on the information from previous sections, the fault tolerance taxonomy for modern SoC FPGA was created (Fig. 2). Critical and vulnerable components pointed out by research papers are marked by a red square, and components that have some vendor-provided fault detection or fault tolerance solutions are marked using corresponding colors.

As can be seen from the figure, the most protected module in the modern SoC FPGA is the processing system that has watchdog timers, lock-step capabilities, ECC for caches, and memory protection units. However, it should be noted that not every processor supports lock-step functionality, watchdog timers cannot isolate and recover the fault, and memory protection units are not designed to handle faults but can only detect errors in some cases. ECC prevents errors originating from caches, however, the lack of protection in the other components of the processor, including registers that are considered critical as well, means that faults stemming from the other components are not covered. The lock-step mechanism protects the whole processor, however, in the considered SoC families, lock-step functionality is only supported in the ARM R-cores, leaving A-cores vulnerable.

The next protected module is FPGA logic which has scrubbing solutions for configuration memory and ECC for embedded memories. User memory flip-flops, though, are not covered by any means. The least protected modules are peripherals, interconnects, accelerators, and specialized modules, the ones that are less explored in the research articles as well.

The following gaps can be formulated:

1. Most of the vendor-provided fault-tolerance methods protect the processing system and memory components. ECC effectively covers single-bit faults in memories. Since memories usually occupy the major part of the chip, a high percentage of fault tolerance can be achieved by covering only memories using a considerably inexpensive and easy-to-implement mechanism like ECC. However, that leaves computational and control units less protected. They might occupy less area than memories, but faults in those components might lead to a crash.
2. The least protected modules are uncore components that are less considered in the literature as well. However, due to the complexity of the SoC, it is not enough to protect only the processing system. In fact, all components may have an impact on the reliability of the whole system [48].
3. The vendor-provided fault-tolerance methods for cost-efficient COTS devices are not designed for harsh environments or critical applications.
4. Different fault effects and used fault-tolerance methods require a systematic and hierarchical approach for efficient device protection. However, the current approach seems fragmented. The main reason to use the SoC FPGA platform is to use as many of its components as possible, not only the processor or FPGA. In such cases, a communication system, for example, plays an important role. For the application to run correctly, it is important to protect the whole system.

5. Mapping known user-space methods

Clearly, all major FPGA vendors are seeking the most cost-efficient methods to offer basic protection against in-field failures. Advanced or full protection becomes too expensive as the main volume of FPGA devices goes into commercial applications such as industrial, telecom, or video processing, where fault tolerance mechanisms would only introduce additional cost and overhead and, hence, reduce the vendor's competitive advantage. Therefore, gaps in protection are inevitable and have to be addressed carefully by the users targeting commercial off-the-shelf (COTS) devices for mission or safety-critical applications such as aerospace.

The pursuit of improving fault tolerance is not limited to the features that the manufacturer embeds into their devices. There is ongoing research on methods to analyze and improve the fault tolerance [49] using the controls available to the user of an SoC, such as programming of the processor cores or the FPGA logic in a way to make the operation more reliable. Several of those methods are typically based on exploiting some sort of redundancy that is already existing or can be created by the user.

Since the modern SoC FPGAs inevitably contain heterogeneous modules, which are also commonly found separately, it makes sense to look at the methods which are already employed for such modules.

General purpose processing units (PS). The availability of multi-core CPUs in modern SoCs allows for exploiting the inherent redundancy for improving transient fault tolerance in software. Where available in hardware, dual-core or triple-core lock-step redundant core configurations can provide significant protection against transient errors [9,50]. A hybrid lock-step approach with hard CPU cores in combination with redundant soft cores is also possible and demonstrated in [51]. Fault tolerance methods described in [52] allow using temporal redundancy and target non-protected COTS CPUs. They enable the protection by trading higher performance and a larger quantity of these CPUs (compared to fault-tolerant alternatives, if any are present) for fault tolerance by employing redundant code execution with some variable but predictable performance penalty. These methods use structural or temporal *active* redundancy where the reaction to fault detection as well as recovery from an error can be flexible and can depend on the executed application. Another software-centric approach proposed

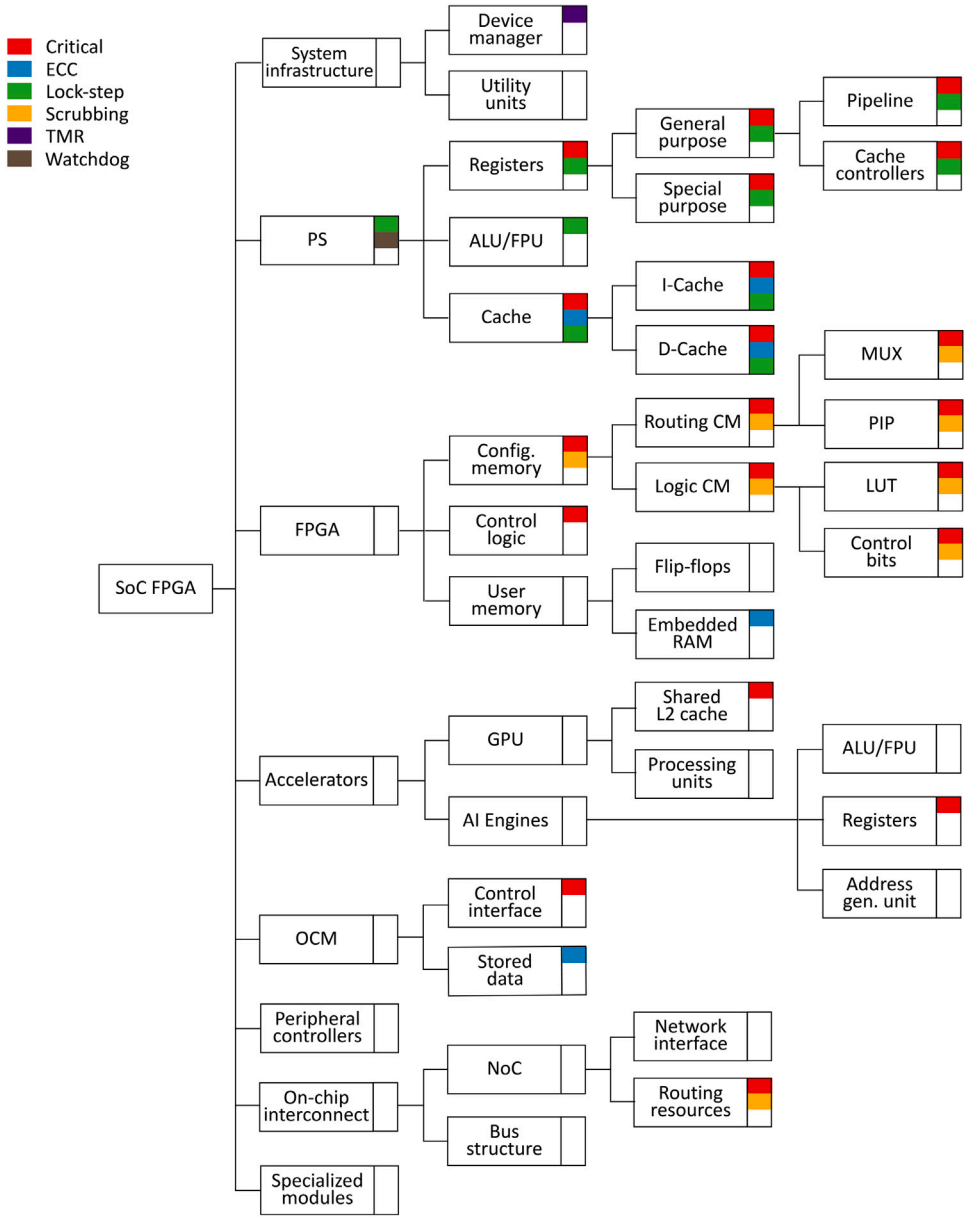

Fig. 2. Fault tolerance taxonomy for SoC FPGA based on vendor-provided methods.

in [53] uses process-level redundancy (PLR), where a set of redundant copies of an application process is created and the results are systematically compared to guarantee correct execution. Redundant multi-threaded processes are proposed for fault tolerance in user-space programs running on multi-core CPUs [54] where the results of two threads are compared for error detection while another thread is used as a watchdog. There are other similar methods from other groups which aim at increasing fault tolerance in software [55,56]. Several methods aim at soft-core CPUs. One of them uses a combination of redundancy in multi-core processors and extra hardware measures for managing redundancy configurations in asymmetric multiprocessing mode (AMP) [57]. In [58], authors implement fault detection in software through thread-replication and coarse-grain lock-step for weakly coupled processing cores.

FPGA fabric is configurable and, from the functional point of view, can work like any other digital hardware and, therefore, can implement hardware methods like TMR, ECC, and others in so-called soft cores. The underlying physical structure of FPGA, however, is very different from usual digital hardware because the function depends on the configuration memory as described earlier. FPGA works properly as long as the configuration memory is intact. In the case of SRAM-based FPGA, the configuration memory is susceptible to SEUs, and these may result in a change of the function. Therefore, usual hardware fault-tolerance methods need to be applied carefully. For example, TMR can be combined with scrubbing [59] in order to detect and correct the errors in the configuration memory, while TMR protects from soft errors in user logic/memory. SEUs in configuration memory can also affect routing, therefore partitioning and placement of the logic in an FPGA can be exploited to increase fault tolerance [60,61]. For example, it can change the probability of errors affecting TMR [62]. Other methods also aim at repairing broken configuration by partial dynamic reconfiguration [63].

Accelerators. Various hardware accelerators are used to increase the performance and/or efficiency of the processing on SoCs, and they, too, can benefit from added fault tolerance. For example, neural network accelerators are susceptible to hardware-induced faults, but there are methods to mitigate the effects of faults [64]. For fixed-hardware neural network accelerators, there are software methods like result

range restriction [65] and algorithm-based fault tolerance [66]. Soft-core accelerators can be augmented with an extra layer for improved fault tolerance [67].

6. Discussion and conclusions

This paper reviewed the state of the art in fault-tolerance mechanisms offered by SoC FPGA vendors with a focus on the most popular COTS devices of the latest SoC families. There has been substantial progress in such protection mechanisms during the last decade as opposed to simple technology-level hardening. As a result, there is a clear industrial trend to build mission- and safety-critical systems around COTS devices [68]. For instance, in the space segment, this trend is further escalated by the need for in-orbit high-performance computing [69].

Still, as the vast majority of SoC FPGAs are employed at non-critical commercial applications, there are gaps in the protection offered off the shelf by vendors. Advanced SoC FPGAs comprise a diversity of tightly integrated components that should all be protected to ensure the system's reliability and availability. However, as was shown, the most protected components are the processing system and memories, leaving the rest of the system vulnerable. Uncore components are not protected and are less considered in the literature. The current approach to fault tolerance is fragmented and not systematic. Finally, vendor-provided fault-tolerance solutions for COTS devices are not intended for harsh environments and critical applications.

Hence, it is necessary to implement complementary user-space fault-tolerance mechanisms tailoring them to target mission and application profiles, from simple custom TMR [70] to advanced fault handling schemes [71]. For better protection, user-space methods should interact with vendor-provided ones resulting in an integrated fault-tolerance solution. For example, the scrubbing procedure can be triggered by a TMR voter in the FPGA logic or lower-cost fault detection checkers or monitors in the FPGA logic. Also, in some scenarios, fault detection may be supported by employing higher-level techniques such as time-domain redundancy for unprotected processors, software-based checking, and external performance/process monitoring. Here, the user-space fault-tolerance mechanisms would both protect the operation and initiate the recovery procedure offered by the vendor, thus improving the overall protection efficiency. More advanced fault handling mechanisms involving cross-layer interaction and data exchange are also possible [72].

However, implementing fault-tolerance solutions on different layers without considering the whole picture might result in an overprotected system or even a degradation in the overall reliability. A prominent solution to cover the aforementioned gaps efficiently is a cross-layer reliability approach.

Cross-layer approach to reliability has been gaining popularity in recent years since it can help to develop a globally optimized fault management system. Several frameworks have been developed by the academic community for the reliability-aware design space exploration to find the best combination of fault-tolerance methods [73,74] and for the cross-layer reliability assessment of the system [75]. Those frameworks primarily target processors at the early design stage, considering HW/SW co-design process. However, as already mentioned, COTS devices are used more and more for critical applications. Thus, a framework targeting heterogeneous SoC FPGAs at the stage of embedded system development would be useful.

7. Summary

This paper presents an analysis of the most common types of errors caused by faults in different components and identifies the most critical and vulnerable components based on the literature survey. The study looks into vendor-provided and user-space reliability solutions. The paper highlights the existing fault-tolerance gaps in the modern SoC FPGAs and outlines a vision for future solutions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported in part by the European Union through European Social Fund in the frames of the "Information and Communication Technologies (ICT) programme" ("ITA-IoIT" topic), by the Estonian Research Council grant PUT PRG1467 "CRASHLES", and by ESA GSTP Contract No. 4000139825/22/NL/AS.

References

[1] C.M. Fuchs, K. Marinis, P. Chou, X. Wen, N.M. Murillo, G. Furano, S. Holst, A. Tavoularis, S.-K. Lu, A. Plaat, A fault-tolerant MPSoC for CubeSats, in: 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT, IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/DFT.2019.8875417.
[2] Versal: The First Adaptive Compute Acceleration Platform (ACAP), Xilinx, Inc., 2020.
[3] Zynq UltraScale+ MPSoC Data Sheet: Overview, Xilinx, Inc., 2021.
[4] Intel Agilex FPGAs and SoCs Device Overview, Intel, 2021.
[5] Intel Stratix 10 GX/SX Device Overview, Intel, 2021.
[6] PolarFire SoC Product Overview, Microchip Technology Inc., 2021.
[7] W. Yang, X. Du, C. He, S. Shi, L. Cai, N. Hui, G. Guo, C. Huang, Microbeam heavy-ion single-event effect on Xilinx 28-nm system on chip, IEEE Trans. Nucl. Sci. 65 (2018) 545–549, http://dx.doi.org/10.1109/TNS.2017.2776244.
[8] X. Du, C. He, S. Liu, Y. Zhang, Y. Li, W. Yang, Measurement of single event effects induced by alpha particles in the Xilinx Zynq-7010 system-on-chip, J. Nucl. Sci. Technol. 54 (2016) 1–6, http://dx.doi.org/10.1080/00223131.2016.1262294.
[9] X. Iturbe, B. Venu, E. Ozer, Soft error vulnerability assessment of the real-time safety-related ARM Cortex-R5 CPU, in: 2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2016, Institute of Electrical and Electronics Engineers Inc., 2016, pp. 91–96, http://dx.doi.org/10.1109/DFT.2016.7684076.
[10] L.A. Tambara, P. Rech, E. Chielle, J. Tonfat, F.L. Kastensmidt, Analyzing the impact of radiation-induced failures in programmable SoCs, IEEE Trans. Nucl. Sci. 63 (2016) 2217–2224, http://dx.doi.org/10.1109/TNS.2016.2522508.
[11] L.A. Tambara, F.L. Kastensmidt, P. Rech, F. Lins, N.H. Medina, N. Added, V.A. Aguiar, M.A. Silveira, Reliability-performance analysis of hardware and software co-designs in SRAM-Based APSoCs, IEEE Trans. Nucl. Sci. 65 (2018) 1935–1942, http://dx.doi.org/10.1109/TNS.2018.2844250.
[12] R. Travessini, P.R. Villa, F.L. Vargas, E.A. Bezerra, Processor core profiling for SEU effect analysis, in: 2018 IEEE 19th Latin-American Test Symposium, Vol. 2018-January, LATS 2018, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 1–6, http://dx.doi.org/10.1109/LATW.2018.8347235.
[13] G. Papadimitriou, D. Gizopoulos, Demystifying the system vulnerability stack: Transient fault effects across the layers, in: Proceedings - International Symposium on Computer Architecture, Vol. 2021-June, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 902–915, http://dx.doi.org/10.1109/ISCA52012.2021.00075.
[14] P. Bodmann, G. Papadimitriou, D. Gizopoulos, P. Rech, The impact of SoC integration and OS deployment on the reliability of ARM processors, in: Proceedings - 2021 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2021, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 223–225, http://dx.doi.org/10.1109/ISPASS51385.2021.00040.
[15] H. Quinn, Radiation effects in reconfigurable FPGAs, Semicond. Sci. Technol. 32 (2017) http://dx.doi.org/10.1088/1361-6641/aa57f6.
[16] C. Bernardeschi, L. Cassano, A. Domenici, L. Sterpone, ASSESS: A simulator of soft errors in the configuration memory of SRAM-based FPGAs, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 33 (2014) 1342–1355, http://dx.doi.org/10.1109/TCAD.2014.2329419.
[17] C. Thibeault, S. Pichette, Y. Audet, Y. Savaria, H. Rufenacht, E. Gloutnay, Y. Blaquiere, F. Moupfouma, N. Batani, On extra combinational delays in SRAM FPGAs due to transient ionizing radiations, IEEE Trans. Nucl. Sci. 59 (2012) 2959–2965, http://dx.doi.org/10.1109/tns.2012.2222668.
[18] T. Nidhin, A. Bhattacharyya, R. Behera, T. Jayanthi, 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. 49 (2017) 1589–1599, http://dx.doi.org/10.1016/j.net.2017.09.002.

[19] N. Jing, J.-Y. Lee, Z. Feng, W. He, Z. Mao, L. He, SEU fault evaluation and characteristics for SRAM-based FPGA architectures and synthesis algorithms, Vol. 18, Association for Computing Machinery (ACM), 2013, pp. 1–18, http://dx.doi.org/10.1145/2390191.2390204.

[20] P.S. Graham, M.P. Caffrey, J.A. Zimmerman, P. Sundararajan, E.J. Johnson, C.D. Patterson, Consequences and categories of SRAM FPGA configuration SEUs, in: Int. Conf. MAPLD, 2003.

[21] N. Rollins, M. Wirthlin, M. Caffrey, P. Graham, Reliability of programmable input/output pins in the presence of configuration upsets, in: MAPLD '02 Laurel MD, 2002.

[22] F.Z. Tazi, C. Thibeault, Y. Savaria, S. Pichette, Y. Audet, On extra delays affecting I/O blocks of an SRAM-based FPGA due to ionizing radiation, IEEE Trans. Nucl. Sci. 61 (2014) 3138–3145, http://dx.doi.org/10.1109/tns.2014.2369417.

[23] F.Z. Tazi, C. Thibeault, Y. Savaria, Detailed analysis of radiation-induced delays on I/O blocks of an SRAM-based FPGA, in: Canadian Conference on Electrical and Computer Engineering, Vol. 2016-October, Institute of Electrical and Electronics Engineers Inc., 2016, http://dx.doi.org/10.1109/CCECE.2016.7726600.

[24] V.M. Placinta, L.N. Cojocariu, C. Ravariu, Proton-induced radiation effects in the I/O blocks of an SRAM-based FPGA, J. Instrum. 14 (2019) T10001, http://dx.doi.org/10.1088/1748-0221/14/10/T10001.

[25] J. Yang, H. Guo, L. Chen, Y. Zhang, X. Li, SEU sensitivity evaluation of JTAG circuit used for SRAM-based FPGA, in: 2017 IEEE 24th International Symposium on the Physical and Failure Analysis of Integrated Circuits, IPFA, IEEE, 2017, pp. 1–5, http://dx.doi.org/10.1109/IPFA.2017.8060186.

[26] J.D. Anderson, J.C. Leavitt, M.J. Wirthlin, Neutron radiation beam results for the Xilinx UltraScale+ MPSoC, in: 2018 IEEE Nuclear & Space Radiation Effects Conference, NSREC 2018, IEEE, 2018, pp. 1–7, http://dx.doi.org/10.1109/NSREC.2018.8584297.

[27] K. Höflinger, S. Müller, T. Peng, M. Ulmer, D. Lüdtke, A. Gerndt, Dynamic fault tree analysis for a distributed onboard computer, in: IEEE Aerospace Conference Proceedings, Vol. 2019-March, IEEE Computer Society, 2019, http://dx.doi.org/10.1109/AERO.2019.8742128.

[28] Versal ACAP AI Engine Architecture Manual, Xilinx, Inc., 2021.

[29] F.F.D. Santos, P.F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, P. Rech, Analyzing and increasing the reliability of convolutional neural networks on GPUs, IEEE Trans. Reliab. 68 (2019) 663–677, http://dx.doi.org/10.1109/TR.2018.2878387.

[30] G. Li, S.K.S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, S.W. Keckler, Understanding error propagation in deep learning neural network (DNN) accelerators and applications, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Vol. 12, SC 2017, Association for Computing Machinery, Inc, 2017, http://dx.doi.org/10.1145/3126908.3126964.

[31] D.A.G.D. Oliveira, L.L. Pilla, T. Santini, P. Rech, Evaluation and mitigation of radiation-induced soft errors in graphics processing units, IEEE Trans. Comput. 65 (2016) 791–804, http://dx.doi.org/10.1109/TC.2015.2444855.

[32] X. Du, C. He, S. Liu, D. Luo, X. Du, W. Yang, Y. Li, Y. Fan, Analysis of sensitive blocks of soft errors in the Xilinx Zynq-7000 system-on-chip, Nucl. Instrum. Methods Phys. Res. A 940 (2019) 125–128, http://dx.doi.org/10.1016/J.NIMA.2019.06.015.

[33] Y. Li, E. Cheng, S. Makar, S. Mitra, Self-repair of uncore components in robust system-on-chips: An OpenSPARC T2 case study, in: Proceedings - International Test Conference, 2013, http://dx.doi.org/10.1109/TEST.2013.6651907.

[34] H. Cho, C.Y. Cher, T. Shepherd, S. Mitra, Understanding soft errors in uncore components, in: Proceedings - Design Automation Conference, Vol. 2015-July, Institute of Electrical and Electronics Engineers Inc., 2015, http://dx.doi.org/10.1145/2744769.2744923.

[35] H. Cho, E. Cheng, T. Shepherd, C.Y. Cher, S. Mitra, System-level effects of soft errors in uncore components, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 36 (2017) 1497–1510, http://dx.doi.org/10.1109/TCAD.2017.2651824.

[36] Versal ACAP Programmable Network on Chip and Integrated Memory Controller, Xilinx, Inc., 2021.

[37] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, C.R. Das, Exploring fault-tolerant network-on-chip architectures, in: Proceedings of the International Conference on Dependable Systems and Networks, Vol. 2006, 2006, pp. 93–104, http://dx.doi.org/10.1109/DSN.2006.35.

[38] A. Prodromou, A. Panteli, C. Nicopoulos, Y. Sazeides, NoCAlert: An on-line and real-time fault detection mechanism for network-on-chip architectures, in: Proceedings - 2012 IEEE/ACM 45th International Symposium on Microarchitecture, MICRO 2012, 2012, pp. 60–71, http://dx.doi.org/10.1109/MICRO.2012.15.

[39] Y. Chen, H. Chen, X. Zhang, P. Lai, Failure localization and mechanism analysis in system-on-chip (SOC) using advanced failure analysis techniques, in: ICEPT-HDP 2012 Proceedings - 2012 13th International Conference on Electronic Packaging Technology and High Density Packaging, 2012, pp. 1348–1351, http://dx.doi.org/10.1109/ICEPT-HDP.2012.6474856.

[40] Intel Agilex Hard Processor System Technical Reference Manual, Intel, 2021.

[41] Zynq UltraScale+ Device Technical Reference Manual, Xilinx, Inc., 2020.

[42] Versal ACAP Technical Reference Manual, Xilinx, Inc., 2021.

[43] Intel Stratix 10 SEU Mitigation User Guide, Intel, 2021.

[44] Intel Agilex SEU Mitigation User Guide, Intel, 2021.

[45] PolarFire SoC MSS Technical Reference Manual, Microchip Technology Inc., 2021.

[46] K.A. Hoque, O.A. Mohamed, Y. Savaria, C. Thibeault, Probabilistic model checking based DAL analysis to optimize a combined TMR-blind-scrubbing mitigation technique for FPGA-based aerospace applications, in: 12th ACM/IEEE International Conference on Methods and Models for System Design, MEMOCODE 2014, Institute of Electrical and Electronics Engineers Inc., 2014, pp. 175–184, http://dx.doi.org/10.1109/MEMCOD.2014.6961856.

[47] J. Vidmar, P. Maillard, T. Jones, M. Sawant, G. Gambardella, N. Fraser, Space DPU: Constructing a radiation-tolerant, FPGA-based platform for deep learning acceleration on space payloads, in: OBDP2021 - 2nd European Workshop on on-Board Data Processing, OBDP2021, 2021, http://dx.doi.org/10.5281/ZENODO.5639613.

[48] A.B.D. Oliveira, L.A. Tambara, F. Benevenuti, L.A. Benites, N. Added, V.A. Aguiar, N.H. Medina, M.A. Silveira, F.L. Kastensmidt, Evaluating soft core RISC-V processor in SRAM-based FPGA under radiation effects, IEEE Trans. Nucl. Sci. 67 (2020) 1503–1510, http://dx.doi.org/10.1109/TNS.2020.2995729.

[49] J. Lojda, J. Podivinsky, Z. Kotasek, Reliability indicators for automatic design and analysis of fault-tolerant FPGA systems, in: LATS 2019 - 20th IEEE Latin American Test Symposium, Institute of Electrical and Electronics Engineers Inc., 2019, http://dx.doi.org/10.1109/LATW.2019.8704593.

[50] A.B. De Oliveira, G.S. Rodrigues, F.L. Kastensmidt, N. Added, E.L. Macchione, V.A. Aguiar, N.H. Medina, M.A. Silveira, Lockstep dual-core ARM A9: implementation and resilience analysis under heavy ion-induced soft errors, IEEE Trans. Nucl. Sci. 65 (8) (2018) 1783–1790, http://dx.doi.org/10.1109/TNS.2018.2852606.

[51] S. Kasap, E.W. Wachter, X. Zhai, S. Ehsan, K.D. McDonald-Maier, Novel lockstep-based approach with roll-back and roll-forward recovery to mitigate radiation-induced soft errors, in: 2020 IEEE Nordic Circuits and Systems Conference, NORCAS 2020 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2020, http://dx.doi.org/10.1109/NORCAS51424.2020.9265137.

[52] M. Pignol, DMT and DT2: Two fault-tolerant architectures developed by CNES for COTS-based spacecraft supercomputers, in: Proceedings - IOLTS 2006: 12th IEEE International on-Line Testing Symposium, Vol. 2006, 2006, pp. 203–212, http://dx.doi.org/10.1109/IOLTS.2006.24.

[53] A. Shye, J. Blomstedt, T. Moseley, V.J. Reddi, D.A. Connors, PLR: A software approach to transient fault tolerance for multicore architectures, IEEE Trans. Dependable Secure Comput. 6 (2) (2009) 135–148, http://dx.doi.org/10.1109/TDSC.2008.62.

[54] H. Mushtaq, Z. Al-Ars, K. Bertels, Fault tolerance on multicore processors using deterministic multithreading, in: 2013 8th IEEE Design and Test Symposium, IEEE, 2013.

[55] H. So, M. Didehban, A. Shrivastava, K. Lee, A software-level redundant multithreading for soft/hard error detection and recovery, in: Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition, DATE 2019, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 1559–1562, http://dx.doi.org/10.23919/DATE.2019.8715089.

[56] J. Liu, Z. Zhu, C. Deng, A novel and adaptive transient fault-tolerant algorithm considering timing constraint on heterogeneous systems, IEEE Access 8 (2020) 103047–103061, http://dx.doi.org/10.1109/ACCESS.2020.2999092.

[57] J. Li, Y. Wang, Transient fault tolerance on multicore processor in AMP mode, in: Proceedings - 2021 8th International Conference on Dependable Systems and their Applications, DSA 2021, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 332–337, http://dx.doi.org/10.1109/DSA52907.2021.00051.

[58] C.M. Fuchs, N.M. Murillo, A. Plaat, E. van der Kouwe, D. Harsono, P. Wang, Software-defined dependable computing for spacecraft, in: 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing, PRDC, IEEE, 2018, http://dx.doi.org/10.1109/prdc.2018.00043.

[59] L.A. Benites, F. Benevenuti, A.B. De Oliveira, F.L. Kastensmidt, N. Added, V.A. Aguiar, N.H. Medina, M.A. Guazzelli, Reliability calculation with respect to functional failures induced by radiation in TMR ARM Cortex-M0 soft-core embedded into SRAM-based FPGA, IEEE Trans. Nucl. Sci. 66 (7) (2019) 1433–1440, http://dx.doi.org/10.1109/TNS.2019.2921796.

[60] R.V. Kshirsagar, S. Sharma, Comparative study of the proposed shifting algorithm for fault tolerance in FPGA, in: 2014 International Conference on Advances in Engineering and Technology Research, ICAETR 2014, Institute of Electrical and Electronics Engineers Inc., 2014, http://dx.doi.org/10.1109/ICAETR.2014.7012944.

[61] S. Fouad, F. Ghaffari, M.E.A. Benkhelifa, B. Granado, Context-aware resources placement for SRAM-based FPGA to minimize checkpoint/recovery overhead, in: 2014 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2014, Institute of Electrical and Electronics Engineers Inc., 2014, http://dx.doi.org/10.1109/RECONFIG.2014.7032506.

[62] F. Lima Kastensmidt, L. Sterpone, L. Carro, M. Sonza Reorda, On the optimal design of triple modular redundancy logic for SRAM-based FPGAs, in: Proceedings – Design, Automation and Test in Europe, Vol. II, DATE '05, 2005, pp. 1290–1295, http://dx.doi.org/10.1109/DATE.2005.229.

[63] F. Lahrach, A. Doumar, E. Châtelet, Fault tolerance of SRAM-based FPGA via configuration frames, in: Proceedings of the 2011 IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2011, 2011, pp. 139–142, http://dx.doi.org/10.1109/DDECS.2011.5783066.

[64] M.A. Hanif, M. Shafique, Dependable Deep Learning: Towards Cost-Efficient Resilience of Deep Neural Network Accelerators against Soft Errors and Permanent Faults, in: Proceedings - 2020 26th IEEE International Symposium on on-Line Testing and Robust System Design, IOLTS 2020, Institute of Electrical and Electronics Engineers Inc., 2020, http://dx.doi.org/10.1109/IOLTS50870.2020.9159734.

[65] L.H. Hoang, M.A. Hanif, M. Shafique, FT-ClipAct: Resilience Analysis of Deep Neural Networks and Improving their Fault Tolerance using Clipped Activation, in: Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition, DATE 2020, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 1241–1246, http://dx.doi.org/10.23919/DATE48585.2020.9116571, arXiv:1912.00941.

[66] K. Zhao, S. Di, S. Li, X. Liang, Y. Zhai, J. Chen, K. Ouyang, F. Cappello, Z. Chen, FT-CNN: Algorithm-based fault tolerance for convolutional neural networks, IEEE Trans. Parallel Distrib. Syst. 32 (7) (2021) 1677–1689, http://dx.doi.org/10.1109/TPDS.2020.3043449, arXiv:2003.12203.

[67] T. Horita, T. Murata, I. Takanami, A multiple-weight-and-neuron-fault tolerant digital multilayer neural network, in: Proceedings - IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2006, pp. 554–562, http://dx.doi.org/10.1109/DFT.2006.8.

[68] J. Budroweit, H. Patscheider, Risk assessment for the use of COTS devices in space systems under consideration of radiation effects, Electronics 10 (9) (2021) http://dx.doi.org/10.3390/electronics10091008.

[69] G. Lentaris, K. Maragos, I. Stratakos, L. Papadopoulos, O. Papanikolaou, D. Soudris, M. Lourakis, X. Zabulis, D. Gonzalez-Arjona, G. Furano, High-performance embedded computing in space: Evaluation of platforms for vision-based navigation, J. Aerosp. Inf. Syst. 15 (4) (2018) 178–192, http://dx.doi.org/10.2514/1.I010555, arXiv:10.2514/1.I010555.

[70] R. Glein, F. Rittner, A. Becher, D. Ziener, J. Frickel, J. Teich, A. Heuberger, Reliability of space-grade vs. COTS SRAM-based FPGA in N-modular redundancy, in: 2015 NASA/ESA Conference on Adaptive Hardware and Systems, AHS, 2015, pp. 1–8, http://dx.doi.org/10.1109/AHS.2015.7231159.

[71] S.T. Fleming, D.B. Thomas, F. Winterstein, A power-aware adaptive FDIR framework using heterogeneous system-on-chip modules, in: F. Kastensmidt, P. Rech (Eds.), FPGAs and Parallel Architectures for Aerospace Applications: Soft Errors and Fault-Tolerant Design, Springer International Publishing, Cham, 2016, pp. 75–90, http://dx.doi.org/10.1007/978-3-319-14352-1_6.

[72] K. Shibin, S. Devadze, A. Jutman, M. Grabmann, R. Pricken, Health management for self-aware SoCs based on IEEE 1687 infrastructure, IEEE Des. Test 34 (6) (2017) 27–35, http://dx.doi.org/10.1109/MDAT.2017.2750902.

[73] E. Cheng, S. Mirkhani, L.G. Szafaryn, C.-Y. Cher, H. Cho, K. Skadron, M.R. Stan, K. Lilja, J.A. Abraham, P. Bose, S. Mitra, CLEAR: Cross-layer exploration for architecting resilience - combining hardware and software techniques to tolerate soft errors in processor cores, in: Proceedings - Design Automation Conference, Vol. 05-09-June-2016, Institute of Electrical and Electronics Engineers Inc., 2016, http://dx.doi.org/10.1145/2897937.2897996, http://arxiv.org/abs/1604.03062.

[74] A. Savino, A. Vallero, S.D. Carlo, ReDO: Cross-layer multi-objective design-exploration framework for efficient soft error resilient systems, IEEE Trans. Comput. 67 (2018) 1462–1477, http://dx.doi.org/10.1109/TC.2018.2818735.

[75] A. Kritikakou, O. Sentieys, G. Hubert, Y. Helen, J.-F. Coulon, P. Deroux-Dauphin, FLODAM: Cross-layer reliability analysis flow for complex hardware designs, in: DATE 2022 - 25th IEEE/ACM Design, Automation and Test in Europe, Antwerp, Belgium, 2022, pp. 1–6.