On the Evaluation of SEU Effects on AXI Interconnect Within AP-SoCs

Corrado De Sio, Sarah Azimi, and Luca Sterpone(✉)

Politecnico di Torino, Turin, Italy
{corrado.desio, sarah.azimi, luca.sterpone}@polito.it

Abstract. G-Programmable System-on-Chips offering the union of a processor system with a programmable hardware gave rise to applications that choose hardware acceleration to offload and parallelize computationally demanding tasks. Due to flexibility and performance they provide at low cost, these devices are also appealing for several applications in avionics, aerospace and automotive sectors, where reliability is the main concern. In particular, the interconnection architecture, and especially the AXI Interconnection for FPGA-accelerated applications, plays a critical role in these systems. This paper presents a reliability analysis of the AXI Interconnect IP Core implemented on Zynq-7000 AP-SoC against SEUs in the configuration memory of the programmable logic. The analysis has been conducted performing a fault injection campaign on the specific section of the configuration memory implementing the IP Core under test, which has been implemented within a benchmark design. The results are analyzed and classified, highlighting the criticality of the AXI Interconnect IP Core as a point of failure, especially for SEU-hardened hardware accelerator relying on mitigation techniques based on fine-grained and coarse-grained replication.

Keywords: AXI · Interconnecting · AP-SoC · FPGAs · SEUs · Fault injection

1 Introduction

In the last years, the advantages provided by the integration of a processor system and other components such as memories and programmable hardware on a single chip have become appealing for a wide range of applications within several domains. Especially, the reduction of cost and developing time along with the increasing of integration and flexibility is very interesting even in fields such as avionics, aerospace and automotive, where reliability is the main concern [1–3]. In particular, All-Programmable-System-on-Chips (AP-SoCs) combine on the same chip both a processor system and a Field Programmable Gate Array (FPGA), commonly referred as programmable logic. This architecture allows the designer to offload the processor system moving and parallelizing on the programmable logic the computationally demanding tasks as well as implementing customized hardware applications interacting with the processor and the other modules of the chips. Moreover, the time to design is shortened by the reuse of optimized IP blocks provided by vendors and third parties or developed through High-Level Synthesis (HLS) tools (even almost transparently to the user) [4, 5].

A. Brinkmann et al. (Eds.): ARCS 2020, LNCS 12155, pp. 215–227, 2020.
https://doi.org/10.1007/978-3-030-52794-5_16

Typically, the IP blocks are connected to each other and/or with the processor through an Advanced Microcontroller Bus Architecture (AMBA), mainly using an Advanced eXtensible Interface (AXI). However, the benefits provided by the on-chip SRAM-based FPGA come along with the reliability issues characterizing these devices. In particular, Single Event Upsets (SEUs) are a dominant source of error for these devices. SEUs can occur in the memory cells of FPGAs when they are exposed to ionizing radiation typical of the space environment, inducing undesired bitflips in the content of the memory cell struck by the ionized particle. However, the effect of SEUs are usually not permanent even if they may produce error states and outputs in the application. Though, when the corrupted memory cell belongs to the configuration memory of the FPGA it can undermine the correctness of the implemented design causing semi-permanent misbehaviors. Indeed, the behavior of the configurable hardware is defined by a bitstream downloaded in the configuration memory of the device where it programs the basic programmable elements of the FPGA (i.e. LUTs, Flip-Flops, DSP, PIP, etc.). Hence if the corrupted memory cell of the configuration layer was configuring a critical resource of the implemented design, the fault will affect the application until the configuration memory is rewritten with the correct content. To mitigate the SEUs induced errors, several techniques have been proposed, such as periodic refresh of the configuration memory to scrub accumulated faults (scrubbing) and replication of the resources to detect and correct misbehavior (e.g. Dual Module Redundancy, Triple Module redundancy).

In this work, we perform a reliability analysis of the AXI Interconnect IP Core by Xilinx for connecting one or more AXI memory-mapped master devices or cores to one or more memory-mapped slave devices or cores, usually used in the AP-SoC as standard mean to interconnect the processor system and the IP cores implemented on the programmable logic. At first, we proposed a benchmark design, subsequently implemented on an AP-SoC Zynq7000, based on the hardware acceleration paradigm. The reliability analysis is carried out through a fault injection campaign addressing the specific section of the configuration memory related to the AXI Interconnect block under test. The experimental results have been aggregated and analyzed to classify produced errors. Finally, the criticality of the AXI Interconnect IP Core as a point of failure even with the hardening of the hardware-accelerated core has been exposed in the conclusion.

The paper is organized into six sections. Section 2 is dedicated to related works, Sect. 3 reports the background of AP-SoC, reliability of programmable devices against SEUs and the AXI Interconnection IP Core by Xilinx. Section 4 describes the evaluation workflow, the design under test and fault injection platform. In Sect. 5, the performed fault injection campaign is described, and the obtained results are reported, classified and analyzed. To conclude, in Sect. 6, we discuss the results and future work.

2 Related Works

Related works focus mainly on the reliability of HLS generated IP Cores and their interfaces. In [6, 7], the reliability evaluation of AXI Interfaces implemented using Vivado HLS is reported. The authors evaluate the reliability of different versions of a custom IP core characterized by different interfaces and hardened solutions against multiple bit upsets. However, the adopted mitigation techniques are restricted to the HLS core, leaving aside the AXI Interconnection IP Core and only one of the proposed configurations implements a hardened AXI Interface (i.e. triplicated AXI-Stream Interface). Even though the interface is triplicated, the AXI Interconnect IP Core is still unhardened and not replicated. Most importantly, it is in common between the instances of the replicated interface. Indeed, AXI Interconnect IP Core is usually used also for mapping in memory of registers of initialization, status, and management of the AXI-DMA IP Core, that manage direct memory access between the memory and devices and cores implementing AXI4-Stream interfaces. The authors perform injections considering the AXI-DMA IP Cores and AXI Interconnect IP Core as a single module, reporting that it is a weak link even after interface replication. In [8], the authors perform analysis on the previously described configuration, reporting the same trend for replicated interfaces connected to a single unhardened AXI Interconnect IP Core.

3 Background

3.1 AP-SoC

The AP-SoC is an electronic device integrating a hard processor system and an FPGA into the same chip. Figure 1 shows the general scheme of the Zynq-7000 AP-SoC architecture. However, the block architecture of different AP-SoCs is very similar and they differ mainly for the I/Os, the characteristics of the processor systems as well as for the size and technology of the programmable logic. In this paper, we carry out our experiments and analyses on the Zynq-7000 AP-SoC which consists of a dual-core ARM Cortex-A9 processors and a 28 nm Series 7 programmable logic [9].

The presence of both processor system and programmable hardware enables the possibility to easily combine software programmability with custom hardware acceleration. An FPGA, and so the integrated programmable logic of the AP-SoC, consists of two layers named Application Layer and Configuration Layer. The Application Layer is composed of the resources available to the user to implement the desired hardware application. The resources are programmable logic elements, such as DSPs, LUTs, flip-flops, and programmable routing elements, such as Programmable-Interconnection-Points (PIPs). The logic functions of these resources and their interconnection are univocally defined by the content of the configuration memory. Indeed, the programmable logic is programmed by downloading and storing a bitstream into the configuration memory. The configuration memory is the main component of the Configuration Layer. Different sections of the configuration memory configure different sections and resources of the Application Layer. The bitstream can be generated by the user using vendor tools starting from the netlist of a target circuit. Additionally, the rise of HLS tools has made the development of core for hardware acceleration easier, providing optimization and interface management through high-level directives [10].

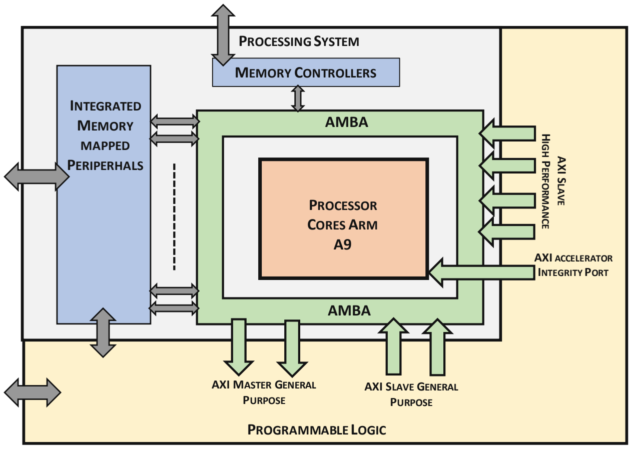
PROCESSING SYSTEM
MEMORY CONTROLLERS
INTEGRATED MEMORY MAPPED PERIPERHALS
AMBA
PROCESSOR CORES ARM A9
AMBA
AXI SLAVE HIGH PERFORMANCE
AXI ACCELERATOR INTEGRITY PORT
AXI MASTER GENERAL PURPOSE
AXI SLAVE GENERAL PURPOSE
PROGRAMMABLE LOGIC

Fig. 1. General overview of Zynq-7000 AP-SoC.

3.2 SEUs in FPGAs Configuration Memory

Single Event Effects (SEEs) are a phenomenon that can occur when the silicon of an integrated circuit is hit by ionizing radiation and particles. The interaction between the silicon of the integrated circuits and the particles can cause several effects in the device, leading to displacement in the lattice of the material, transitory glitches of current, and change the status of bistable elements. In particular, SEUs are one of the most dominant SEEs [11, 12]. They are soft-errors caused by the change of the content of a memory cell when it is struck by a charged particle and so affecting the device functionality for a short period without undermining the device integrity. However, since FPGAs configuration memory data are not usually rewritten during device execution, an SEU corrupting a configuration memory cell will affect the application behavior permanently until the next power cycle or reconfiguration. Figure 2 shows an example of how an interconnection can be disabled by a bitflip in the configuration memory. In the figure, a charged particle striking the memory cell generates an open fault in the interconnection line that can be fixed only by rewriting the correct content in the configuration memory.

In order to mitigate the effects on the application layer induced by the SEUs in the configuration memory, several techniques have been proposed. In particular, SEU-hardening design techniques for FPGAs involve either replication at the gate level (fine-grained) or at the module level (coarse-grained) to detect and eventually correct the errors [13].

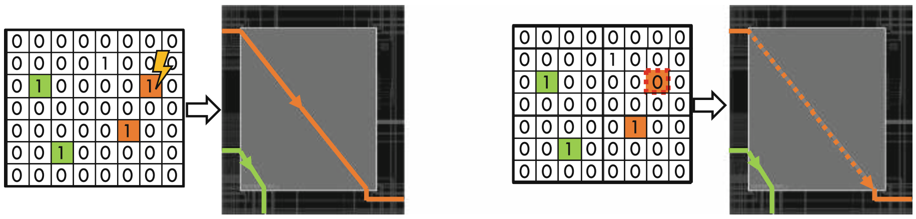

Fig. 2. SEUs affecting the configuration memory section programming a switch matrix.

3.3 AMBA and AXI Interconnect IP Core

AMBA is a standard developed by ARM for the interconnection of blocks in a system-on-chip. AMBA supports high-performance and high-frequency communication and includes the specification for AXI4 interfaces. In particular, AXI4, AXI4-Stream, and AXI4-Lite have been adopted by Xilinx for IP blocks interfacing and on-chip communication [14]. Additionally, as previously illustrated in Fig. 1, AXI ports for general purpose and high-performance communication, as well as the AMBA AXI interconnect, are present on Xilinx AP-SoCs.

The AXI Interconnect IP Core is a logic core provided in the Xilinx IP catalog to be implemented in the programmable logic [15]. It allows connecting AXI masters and AXI slaves modules transparently to the user accordingly with the interface characteristics of any IP block. The AXI Interconnect IP Core can be configured to support various communication models (i.e. 1-to-N, N-to-1, N-to-M).

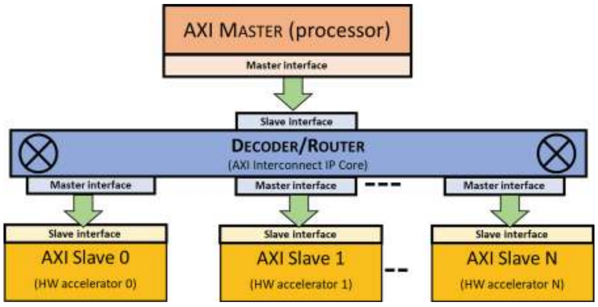

Fig. 3. Single master-multiple slaves AXI architecture.

In this paper, we will focus on the 1-to-N interconnect model, typical of hardware-accelerated systems architecture. In the 1-to-N model, a single master is present and it can access several memory-mapped slaves to use the AXI interconnection module. Figure 3 shows a schema of the 1-to-N communication model. Generally, in the hardware accelerator paradigm, the processor system acts as master, demanding computationally expensive tasks to the hardware modules (slaves). The slave modules can perform different operations if different tasks need to be accelerated on the hardware or perform the same operation on different data vector if a highly parallel computation is desired.

Additionally, the architecture can be used to perform the same function on the same data as mitigation techniques based on the replication, and compare obtained results from the hardware modules to detect and eventually correct errors.

4 Evaluation Platform and Workflow

For analyzing the reliability of the AXI Interconnect IP Core, we developed a benchmark design based on the hardware-accelerator paradigm. The hardware-accelerator module has been replicated emulating a dual with comparison approach for the detection of the misbehaviors in the programmable logic, with the detection check implemented on the processor side. The fault injection campaign has been carried out using a previously developed fault injection platform. The platform can perform automatized fault injection campaigns on the sections of the configuration memory related to the specific modules under test. Moreover, it provides insight into the structure of the configuration memory under test. The platform runs on a host computer connected to the AP-SoC managing of the generation of faulty configuration bitstream as well as of the download in the configuration memory. The test routine executing on the processor system stimulates the hardware accelerators with a randomly generated test vector. The outputs are compared by the processor system with the expected results and sent the execution report to the host computer. Then, the errors are classified accordingly with their characteristics and patterns. In the following subsections, the evaluation platform and the adopted workflow are reported in detail.

4.1 Benchmark Design

The benchmark design consists of a duplicated hardware accelerator connected to the processor system through the AXI Interconnect IP Core. The hardware accelerator has been developed using Vivado HLS. It computes a non-linear signature from four 32 bit fixed-point parameters (22 bits for the integer part and 10 bit for the decimal part). The custom IP core implements also an AXI4-Lite interface that provides access to six memory-mapped registers, four for the inputs, one for the output, and a control register. The architecture of the system under test is illustrated in Fig. 4.

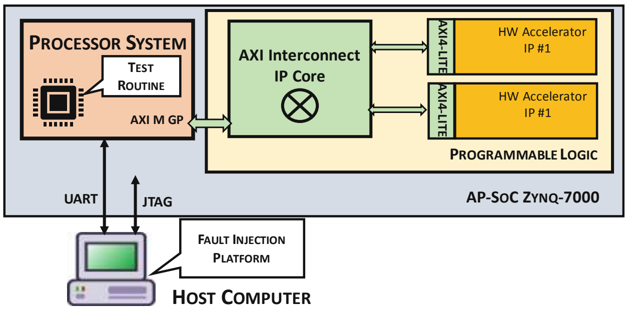

Fig. 4. The overall scheme of the evaluation platform architecture.

The processor system is connected to two instances of the hardware accelerator by the AXI Interconnect IP Core configured as single master-multiple slaves. Both the two hardware accelerators as well as the AXI Interconnect IP Core are implemented in the AP-SoC programmable hardware. Additionally, the AP-SoC is connected with a host computer. The host computer can configure the FPGA using the JTAG interface and can start the test routine running on the processor system through a serial connection with the processor. On the same channel, it can receive the report generated by the test routine.

4.2 Test Routine

The test routine running on the processor system consists of three parts: a preamble, a body, and an epilogue. In the preamble, the routine initializes the software data structures for using the hardware accelerator IPs and verify that they are in a correct state. In the body, the routine stimulates alternately each hardware with 1000 different inputs. For each input, it collects the result and verify its correctness or detects if the IP under test hangs. In the epilogue, the routine reports the status of the IP cores to the fault injection platform and signals the end of the test routine. During all the phases, the processor system reports the status of the test routine and IP cores executions results to the host computer which stores them for future analysis.

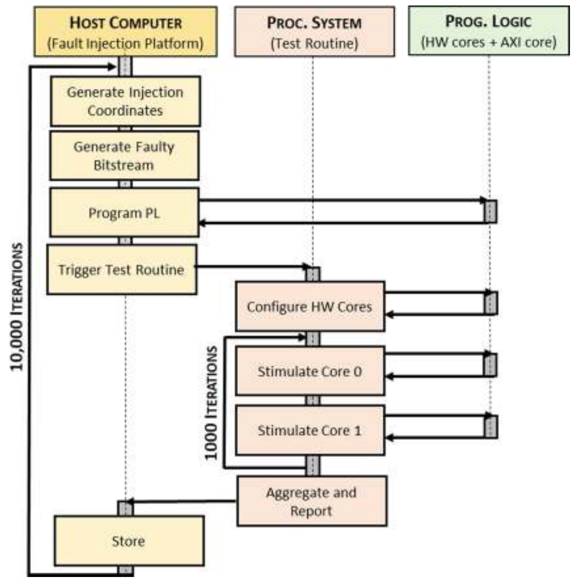

Fig. 5. Fault injection workflow.

4.3 Fault Injection Workflow

An enhanced version of the fault injection platform presented in [16] has been used for the fault injection campaign. PyXEL can interface with Vivado to retrieve the list of the resources used for implementing a specific hierarchical cell (e.g. the AXI Interconnect IP Core) and provide to the user the coordinates in terms of frames and bits where they are programmed. Moreover, it allows generating a visual representation of the content of the configuration memory facilitating the definition of the constraints for the location of the injections. Using these features, it has been possible to limit the fault injection coordinates to the configuration memory section implementing the specific module under test. The platform runs on the host computer and manages the experimental workflow. In detail, it controls the generation of the injection locations and injected bitstreams, the download of the faulty bitstreams in the configuration memory, the trigger of the test routine, and the collection of results. All the injections steps are automated and executed by the platform without user interaction. A representation of the described workflow is reported in Fig. 5

5 Experimental Analysis and Results

The reliability analysis of AXI Interconnect IP Core has been performed through a fault injection campaign. The AXI Interconnect IP Core has been implemented in the programmable logic of a Zynq-7000 AP-SoC within a benchmark design based on the hardware accelerator paradigm. Zynq-7000 AP-SoC integrates on the same chip a Dual-core ARM Cortex-A9 and a 28 nm Xilinx Series 7 programmable logic. The performed fault injection campaign emulates SEUs effect in the configuration memory through bitflips in the bitstream. The amount of injection has been selected accordingly with the target confidence and error margin. The coordinates of injection have been chosen in order to affect only the AXI Interconnect IP core under test. Results are reported in terms of overall error rate computed as the number of faulty bitstreams that generated an error in one or more on-hardware computation out of the amount of faulty bitstream tested. Additionally, incorrect behaviors of the design under test have been classified in different categories accordingly to their effect on the system.

5.1 Fault Injection Campaign

To perform an analysis of the errors produced by emulating radiation-induced SEUs in the configuration memory affecting the AXI Interconnect IP Core, a fault injection campaign has been carried out. We singularly injected 10,000 bitflips in the configuration memory section configuring the AXI Interconnect IP Core under evaluation. The configuration memory section selected for injection consists of 338,446 bits and spans over 196 frames belonging to a single clock region. Figure 6 shows the floorplanning of the implemented benchmark design as shown by the Vivado device view. The resources implementing the AXI Interconnect IP Core are represented in blue and highlighted by a yellow square. In this paper, we will focus on the 1-to-N interconnect model, typical of hardware-accelerated systems architecture. In the 1-to-N model, a single master is present and it can access several memory-mapped slaves to use the AXI interconnection module. Figure 3 shows a schema of the 1-to-N communication model. Generally, in the hardware accelerator paradigm, the processor system acts as master, demanding computationally expensive tasks to the hardware modules (slaves). The slave modules can perform different operations if different tasks need to be accelerated on the hardware or perform the same operation on different data vector if a highly parallel computation is desired. Additionally, the architecture can be used to perform the same function on the same data as mitigation techniques based on the replication, and compare obtained results from the hardware modules to detect and eventually correct errors. In the top-left corner is represented the processor system with the hard interconnections. Similarly, Fig. 7 reports a subsection (from the frame 1800 to the frame 7400) of the configuration memory as shown by the fault injection platform. The part to inject is highlighted by the yellow square.

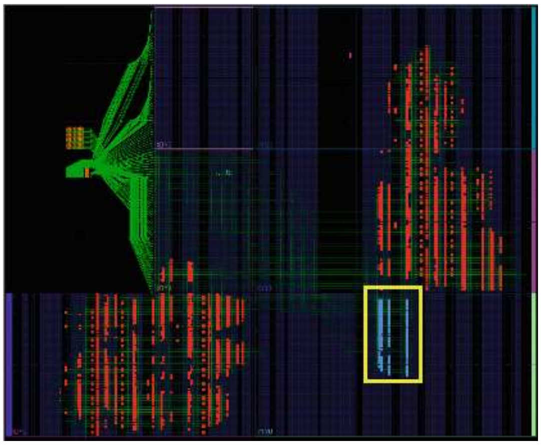

Fig. 6. The Vivado view of the design, and the part selected for the injection (yellow square). (Color figure online)

Please notice that a single frame spans only over a single clock region and frames belonging to the same clock region are sequential in the configuration bitstream and accordingly in the configuration memory view produced by PyXEL. Though, the sequence of clock region in the bitstream is out of order compared to the device view exposed by Vivado. Therefore, blocks in Fig. 7 result displaced compared to their position in Fig. 6.

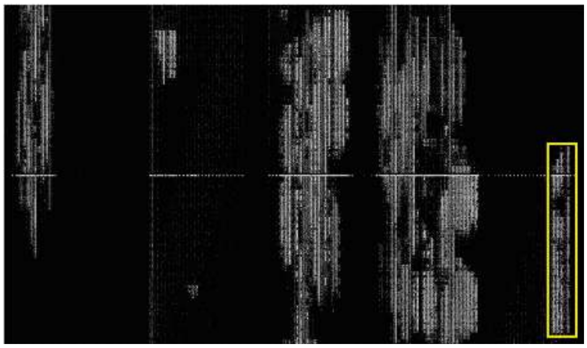

Fig. 7. A Section of the Configuration Memory view of the benchmark generated by PyXEL and the subsection selected for the injection campaign (yellow square). (Color figure online)

Accordingly with (1), 10,000 injections allow to conservatively estimate with 0.01 of margin error (e) and 95% confidence level the probability that a bitflip in the injected section of configuration memory can be a source of error for the system under test [17].

$$n = \frac{N}{1 + e^2 \times \frac{N-1}{t^2 \times p \times (1-p)}} \tag{1}$$

In particular, n is the minimum number of injections needed to meet the target margin of error e. N is the population size (i.e. the 338,446 injectable bits) and p is the estimated probability of a fault to result in an error. We conservatively chose p = 0.5 which maximizes the value of n. The parameter t is the cut-off point corresponding to the desired confidence level computed with respect to the Normal distribution. We choose a 95% confidence level for which the value of t is 1.96.

The injection coordinates, in terms of frames and bit, have been independently and randomly generated subsampling with replacement the section of the configuration memory under test. The fault injection campaign required about 25 h to complete.

5.2 Analysis of Results

As a result 306 out of the 10,000 faulty bitstream have generated errors in the circuit. The fault injection campaign showed an overall error rate of 3.06% with a 1% margin error and a 95% confidence interval. Please notice that only a subset of the resources related to the configuration memory section under test is used and consequently the number of bits with value 1 is significantly lower than the number of bits with value 0. Nonetheless, they are more likely to generate errors in the application when corrupted [18]. In detail, only 10% of the injected faults were 1 to 0 bitflips but 18% generated errors. Therefore about 50% of errors were generated by injections from 1 to 0 despite the asymmetry in the subsampling. Table 1 reports a summary of the injections and results they generated.

Table 1. Summary of SEUs injections and results

Injected fault	Amount	Generating errors
0-to-1	9155 (91.55%)	154 (1.6%)
1-to-0	845 (8.45%)	152 (17.9%)
Total	10 000 (100%)	306 (3.06%)

5.3 Classification of Faulty Results

The results have been classified accordingly with the issue they produced in the system. As reported in Table 2, more than 88% of errors provoked a failure of both the hardware accelerators. Please notice that even if for clarity faults are classified accordingly with the effect they generate on the communication with the hardware accelerator cores, given the injection process previously reported, all the faults are caused by a malfunction of the AXI Interconnect IP Core. In detail, in 49.35% of the detected misbehaviors, both the hardware accelerator cores cannot be initialized correctly or stopped to work after few successful communications. In 32.68% of the cases, both the hardware cores can be reached by the processor system but their computations are faulty. In 4.90% of cases, only one core out of two can be reached by the test routine but the results it returns are wrong. Only in the 11.76% of the faulty cases, we have one of the two core behaving correctly, while the other does not respond to the stimuli. The results are summarized in Table 2.

Table 2. Classification of system faults

Class	Amount
Both HW cores hang	151 (49.35%)
Both HW cores fail	100 (32.68%)
Single HW core fails	36 (11.76%)
HW core fails + HW core hang	15 (4.90%)
Others	4 (1.31%)
Total	306 (100%)

Additionally, it has been observed that when both the cores produce erroneous computation, the returned values are the same. Therefore, it is impossible to detect the fault without comparing them with the golden result even with the replication of the hardware accelerator.

6 Conclusions and Future Works

In this paper, we evaluated the reliability of AXI Interconnect IP Core against SEUs. The study has been carried out through a fault injection campaign in the configuration memory of Zynq-7000 programmable logic. Errors have been classified accordingly with their effect on the system and type. The analysis has shown as AXI Interconnection IP Core can be a source of errors for architecture exploiting hardware acceleration. Additionally, it has been shown as errors generated in the interconnection core can thwart mitigation techniques based on the replication of hardware modules. As future work, the design of a hardened the AXI Interconnect IP core needs to be performed to perform a comparative analysis

References

1. Flesch, G., Keymeulen, D., Dolman, D., Holyoake, C., McKee, D.: A system-on-chip platform for earth and planetary laser spectrometers. In: 2017 IEEE Aerospace Conference, Big Sky, MT, pp. 1–12 (2017)
2. Sabogal, S., George, A., Crum, G.: ReCoN: a reconfigurable CNN acceleration framework for hybrid semantic segmentation on hybrid SoCs for space applications. In: 2019 IEEE Space Computing Conference, SCC, Pasadena, CA, USA, pp. 41–52 (2019)
3. Shea, E., George, A.: OPIR video preprocessing and compression for on-board aerospace computing. In: 2017 IEEE National Aerospace and Electronics Conference, NAECON, Dayton, OH, pp. 142–148 (2017)
4. Vaidya, B., Surti, M., Vaghasiya, P., Bordiya, J., Jain, J.: Hardware acceleration of image processing algorithms using Vivado high level synthesis tool. In: 2017 International Conference on Intelligent Computing and Control Systems, ICICCS, Madurai, pp. 29–34 (2017)
5. Toft. J.K., Nannarelli, A.: Implementation of hardware accelerators on Zynq, Kgs. Lyngby: Technical University of Denmark. DTU Compute-Technical Report-2016, No. 7 (2016)
6. Benevenuti, F., Kastensmidt, F.L.: Reliability evaluation on interfacing with AXI and AXI-S on Xilinx Zynq-7000 AP-SoC. In: 2018 IEEE 19th Latin-American Test Symposium, LATS, Sao Paulo, pp. 1–6 (2018)
7. dos Santos, A.F., Tambara, L.A., Benevenuti, F., Tonfat, J., Kastensmidt, F.L.: Applying TMR in Hardware Accelerators Generated by High-Level Synthesis Design Flow for Mitigating Multiple Bit Upsets in SRAM-Based FPGAs. In: Wong, S., Beck, A.C., Bertels, K., Carro, L. (eds.) ARC 2017. LNCS, vol. 10216, pp. 202–213. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56258-2_18
8. Benevenuti, F., Kastensmidt, F.L.: Analyzing AXI Streaming Interface for Hardware Acceleration in AP-SoC Under Soft Errors. In: Voros, N., Huebner, M., Keramidas, G., Goehringer, D., Antonopoulos, C., Diniz, Pedro C. (eds.) ARC 2018. LNCS, vol. 10824, pp. 243–254. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78890-6_20
9. Xilinx, Inc.: Zynq-7000 All Programmable SoC: Technical reference manual, San Jose, CA, USA, User Guide, UG585, July 2018
10. Xilinx, Inc.: Vivado design suite user guide: high level synthesis, San Jose, CA, USA, User Guide, UG902, December 2018
11. Quinn, H.: Radiation effects in reconfigurable FPGAs. Semicond. Sci. Technol. **32**(4), 044001 (2017)
12. Du, B., et al.: Ultrahigh energy heavy ion test beam on Xilinx Kintex-7 SRAM-based FPGA. IEEE Trans. Nucl. Sci. **66**(7), 1813–1819 (2019)

13. Siegle, F., Vladimirova, T., Ilstad, J., Emam, O.: Mitigation of radiation effects in SRAM-Based FPGAs for space applications. ACM Comput. Surv. **47**(2), 34 (2015). Article 37
14. Xilinx, Inc.: Vivado Design Suite: AXI Reference Guide, San Jose, CA, USA, User Guide, UG1037, July 2017
15. Xilinx, Inc.: AXI Interconnect v2.1: LogiCORE IP Product Guide, San Jose, CA, USA, Product Guide, PG059, December 2017
16. Bozzoli, L., De Sio, C., Sterpone, L., Bernardeschi, C.: PyXEL: an integrated environment for the analysis of fault effects in SRAM-based FPGA routing. In: 2018 International Symposium on Rapid System Prototyping, RSP, Torino, Italy (2018)
17. Leveugle, R., Calvez, A., Maistri, P., Vanhauwaert, P.: Statistical fault injection: quantified error and confidence. In: 2009 Design, Automation & Test in Europe Conference & Exhibition, Nice, pp. 502–506 (2009)
18. De Sio, C., Azimi, S., Bozzoli, L., Du, B., Sterpone, L.: Radiation-induced single event transient effects during the reconfiguration process of sram-based FPGAs. Microelectro. Reliab. **100**, 113342 (2019). ISSN 0026-2714