A New Domains-based Isolation Design Flow for Reconfigurable SoCs

Andrea Portaluri, Corrado De Sio, Sarah Azimi, Luca Sterpone
Dipartimento di Automatica e Informatica (DAUIN)
Politecnico di Torino
Turin, Italy
andrea.portaluri@studenti.polito.it
{corrado.desio, sarah.azimi, luca.sterpone}@polito.it

Abstract— **Reconfigurable SoCs are widely adopted in mission-critical tasks in aerospace and automotive. Though, one of their main drawbacks is the susceptibility to high-energy particles both in space and at sea level. Isolation Design Flow is a promising implementation approach to improve the reliability of circuits. However, considering the high number of modules in a complex circuit, especially when redundant techniques are applied, IDF requires a complex floorplanning stage. In this paper, the benefits of using IDF are evaluated, both for plain and hardened-by-redundancy designs. We propose an implementation methodology to tackle the complexity of applying IDF to TMR-based circuits that usually make the implementation approach unfeasible. The impact of different design policies on the reliability of the system is evaluated through fault injection campaigns. The proposed method is applied to the TMR-hardened CORDIC core implemented on Zynq AP-SoC and compared with other possible solutions. The results report a significant improvement in the TMR effectiveness when the proposed domains-based IDF is applied.**

Keywords—**Fault injection, Isolation design flow, Reliability, Reconfigurable, SoC, SEU, SRAM-based FPGA.**

I. Introduction

Lately, Reconfigurable Systems on a Chip (SoCs) and Field-Programmable Gate Arrays (FPGAs) have become attractive solutions for many safety-critical applications in the automotive and aerospace industries. Thanks to their hardware programmability, the designers can meet high performance and strict requirements with relatively low costs, power consumption, and time-to-market. One of the main limitations to the usage of SRAM-based programmable hardware platforms in safety-critical applications is their Single-Event Upsets (SEUs) susceptibility [1][2][3]. An SEU is a harmful modification of the content of a memory cell (i.e., a bit) caused by the interaction between a single ionizing particle and the device matter. The electronic circuit (i.e., the netlist) implemented by the programmable hardware is defined by the content of a configuration memory (CM). Therefore, a modification of the content of the memory may result in a modification in the circuit that compromises the nominal behaviour of the application, possibly leading to critical scenarios.

Due to technology and voltage scaling, SEUs are no longer concerning only applications operating in outer space but also at sea level due to the secondary particles [4]. Therefore, the mitigation of the SEU effects is becoming the main challenge for adopting electronic devices used in any safety-critical applications, both at space and ground level. Several SEU mitigating techniques have been proposed for reconfigurable-oriented SoCs, especially hardening-by-design techniques such as hardware redundancy [5][6][7][7].

The Isolation Design Flow (IDF) is proposed as a promising approach to improve system reliability, acting directly on the design placement especially when applied to the reconfigurable logic of SRAM-based FPGAs. The underlying idea behind IDF is to isolate modules during the floorplanning phase to build more reliable systems. Indeed, module isolation helps at detecting faults in the single modules and limits the propagation of faults between them. The isolation is obtained by adopting a complex set of Design Rule Constraints (DRCs) that guarantees isolation but can easily lead to routing congestion.

Thus, IDF increases the complexity during floorplanning, especially with numerous modules involved, for example when modular redundancy is applied. Therefore, the application of IDF represents a challenging task.

Only a few research works analyzed the benefits introduced by the IDF in terms of the reliability of the application. However, the impact of state-of-the-art IDF and the adoption of different isolation policies on modular redundancy mitigation techniques are not investigated in any research work.

This work provides two main scientific contributions. The first consists on the investigation of applying IDF on an unmitigated circuit in terms of reliability improvement. The reliability evaluation is performed through fault injection, considering both the isolated and non-isolated versions of the same benchmark design. Moreover, the reliability improvement of a hardened-by-replication version of the same circuit is evaluated as well. The topological constraints make the use of the state-of-the-art IDF for the Triple Modular Redundant (TMR) version of the circuit infeasible. Therefore, in this work, we propose a general methodology to reduce the floorplanning complexity for IDF when applied to modular redundant designs. Moreover, the proposed approach for reducing floorplanning complexity is compared with other solutions based on different aggregation policies through fault injection. The experimental results show how the use of the proposed domains-based IDF not only allows to successfully apply the isolation flow but also produces remarkable results in terms of reliability of the system compared to the non-isolated TMR or other module aggregation policy.

The paper is organized as follows: Section II reviews previous works related to isolation and mitigation techniques. Section III gives an overview of the SEUs origin and elaborates on the Isolation Design Flow. The domains-based

Isolation Design Flow is described in Section IV. The experiment environment is presented in Section V. The obtained results are reported in Section VI. Finally, Section VII contains conclusions and discussions on further works.

II. Related Works

So far, few research works have focused on IDF or IDF-based architectures [9][10][11]. The authors in [9] were the first to propose a technique to use IDF with a partial reconfiguration for Xilinx SRAM-based FPGAs, supporting online module relocation. The approach proposed in [10] suggests a novel method to ease the bitstream relocation in presence of IDF constraints. Eventually, the authors in [11] implement off-chip trusted communication with the partial reconfigured section. However, even though the mentioned methods are all highly effective, the FPGA commercial design tool (e.g., Xilinx Vivado) currently does not support any partial reconfiguration integrated with IDF. Therefore, the main challenges remain the need to interface with external tools and the elevated time needed for implementing the design. As far as the design is concerned, the author in [12] lists the most common design-for-reliability solutions such as hardware redundancy, error-correction coding, and configuration scrubbing. Among these, TMR represents one of the most used, effective, and well-known approaches for SEU mitigation [13]. However, to the best of our knowledge, no research works have evaluated the efficiency of applying IDF-based techniques integrated with TMR on increasing the reliability of the design, which is the focus of this work.

III. Background

A. *Single Event Upsets and TMR mitigation*

Due to the interaction of high-energy particles (e.g., ionizing radiation) with the silicon structure of SRAM-based FPGAs, undesired electric reactions can arise within the devices. In particular, when a particle strikes atoms in the silicon lattice, the released energy can change the electric state of a node producing an SEU. One of the most common effects of SEUs is the single bit upset, where the content of a memory cell is flipped. Concerning SRAM-based hardware programmable devices, a bit upset involving a programmed memory cell used in the configuration of the netlist can modify the circuit programmed in the programmable logic. Therefore, these changes will affect the application functionality similar to a hardware fault until the next reconfiguration. A detailed classification of the faults that SEUs can cause in SRAM-based FPGAs is given in [14]. To mitigate the effects of SEUs, replication of the modules in the design is a widely adopted technique (e.g., Duplication With Comparison and TMR). The basic concept of TMR exploits the triplication of the hardware modules, that perform the same operations on the same data. The results of the three copies are voted, allowing to detect and correct single misbehavior out of three. However, replication leads to overhead in terms of power consumption, area, and timing which is not always affordable. Moreover, the increasing of modules composing the design can severely vitiate placement and routing feasibility when the Isolation Design Flow is pursued.

B. *Isolation Design Flow*

In the case of an occurring SEU, modules may encounter chain failures, compromising the correctness of the application. The Isolation Design Flow (IDF) is a design technique adopted to assure the non-interference of functions within the same chip through physical isolation of the resources, thus preventing fault propagation between modules. In order to guarantee the isolation between modules, each module to isolate must have its own hierarchical instance in the hardware description of the netlist (HDL). During the design placement phase, a fence must be used to isolate each module in the design from the others. Fences are a set of contiguous rows/columns of unused resources separating two isolated regions. The requirements on fence width in terms of unused resources is depending on the specific device. The on-chip communication must be implemented using trusted routes. Routes (i.e., nets connecting isolated modules) must fulfill strict requirements to be marked as trusted. In detail, the routes have to connect one source and one destination only (point-to-point connection) and cross only tiles in the fence separating the two isolated regions that the route is connecting.

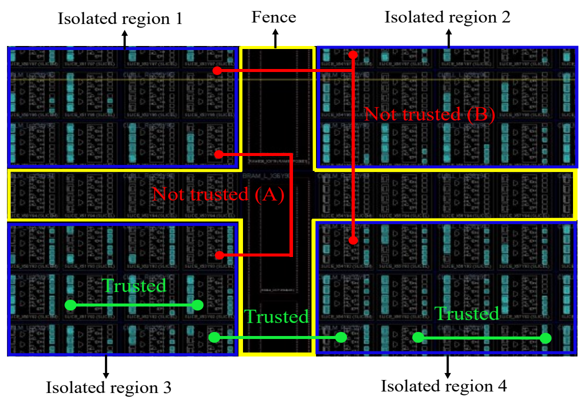

Fig. 1. Available routes: *A* and *B* are representing non-trusted routes.

Fig. 1 shows a conceptual scheme of isolated regions, fences, and routes. *Route A* is not a trusted route since it passes out the fence region comprised between the two isolated regions that it is connecting. *Route B* is not trusted since it does not realize a point-to-point connection.

Due to the need to respect constraints on fences and trusted routes, IDF requires an elaborated floorplanning phase that is only partially supported by the vendor tools. During the floorplanning, the communication between isolated modules can be implemented only between adjacently-placed modules. Moreover, the width of the fence follows some constraints (usually between 3 and 8 tiles). Thus, the manual floorplanning process requested for implementing isolation becomes highly challenging or, in the worst cases, impossible when the number of modules increases. To cope with the increase of complexity, it is possible to group modules under a higher hierarchical level. Therefore, a trade-off between the isolation modules and the complexity and feasibility of the design placement and routing should be considered.

IV. The Domains-Based Isolation Design Flow

We are proposing a set of design practices for reducing floorplanning complexity when a replication-based mitigation approach is used. These rules are intended to simplify the floorplanning phase during the isolation design flow, usually delegated to the designers, that quickly explodes in complexity when the state-of-the-art IDF is applied to replicated modules. Indeed, the very high complexity fails to meet the constraints required by IDF for topological reasons, forcing the designer to give up on isolation. Our proposed solution reduces the number of blocks to be isolated, coupling together different modules within the same isolated region. However, unaware relaxation of the isolation constraints and module aggregation can lead to nullifying the advantages introduced by IDF. For instance, when coarse-grained TMR is applied, the modules to be hardened are replicated. The redundant modules are used for performing the same computation independently. Then, the results are compared to detect and correct possible errors through a voter circuit. Since errors that will not affect more than a single replicated computational unit are filtered by the voter, the underlying idea is to prioritize the isolation between modules that contribute to two different data domains of the voter. Differently, isolation between modules in the same voter domains (i.e., contributing to the same voter input) can be relaxed. The reduction of the number of the isolated regions will consequently reduce the floorplanning constraints and complexity. The steps for integrating domains-based IDF in the traditional FPGA design flow are illustrated in Fig. 2. It consists of the four tasks listed below:

- *Pre-Synthesis*: the isolated regions are defined. Different from the state-of-the-art IDF, the modules to include together in the same isolated region must be regrouped in the design hierarchy. In this phase, it is important to avoid grouping together modules belonging to different domains of the same voter, as explained above. Clock signals or other inter-region signals must be declared in the placement constraint file, thus allowing them to cross isolated regions.
- *Post-Synthesis*: the modules previously identified to form an isolated region must be declared as isolated, in order to let the CAD tool know which modules are intended to be isolated. This property will constraint communication only through the trusted routes.
- *Floorplanning*: the floorplanning phase is executed manually by instantiating placement blocks (pblocks). A pblock is a collection of physical resources (e.g., LUTs, PIPs) of the programmable hardware. The routing and the logic cells of an isolated module will be placed only in the associated pblock. At this stage, the fencing rules must be accurately followed in order to achieve correct isolation. An estimated value of resources needed for the function within the pblock will be reported by the FPGA design tool in order to not run into resource overflow.
- *Post-implementation*: after the implementation, the Vivado Isolation Verifier (VIV) [15] built-in tool is used to verify the correct implementation of the IDF rules between the isolated blocks. This tool generates a report on possible misplacements of blocks or fences and physical overlay of modules.

V. The Fault Injection environment

For evaluating the benefits introduced by the plain and domain-based IDF, a fault injection environment has been developed. The environment automatizes both the faults generation and faults evaluation tasks, as well as results collection and analysis. For this work, the PyXEL framework has been extended to support Essential Bits, allowing to focus of the analysis only on the sensitive bits of the configuration memory [16]. Fig. 2 shows the scheme of the experimental analysis flow. The schema illustrates how the domain-based IDF phases are integrated with the traditional FPGA design flow, as well as the steps and modules involved in the fault injection process.

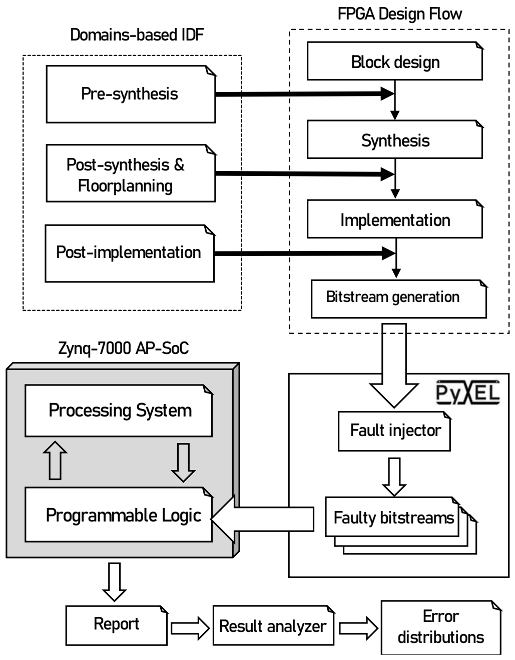

Fig. 2. The scheme of the developed experimental flow.

A. Fault injection platform and methodology

In order to evaluate and compare the reliability of the study cases, we developed the environment for performing fault injection campaigns. The environment runs on a host computer and communicates with the platform implementing the circuit under test through serial communication. The fault injection mechanism relies on the PyXEL framework [16]. PyXEL is a Python library for the analysis of faults in FPGA, which allows modifying single or multiple bits in the bitstream to emulate faults. For this work, PyXEL has been extended to focus on a subset of the configuration memory, named Essential Bits (EB). The EBs are a subset of the programmable bits of the specific circuits that are reported by the vendor tool (i.e., Vivado) as bits that if corrupted may lead to errors in the circuit [17]. The fault injection campaigns consist of a collection of single independent trials. For each trial, an essential bit of the circuit under test is corrupted and the effect introduced by the fault is evaluated. The generated faulty bitstream is used for programming the programmable hardware. Then, a software test routine is loaded in the processing system of the AP-SoC. The software test routine stimulates the computing modules on the PL. Then, it sends the results to the fault injection platform where they are collected and analyzed. All the steps are fully automatized.

VI. Experimental Analyses and Results

For evaluating the benefits introduced by traditional and domain-based IDFs, we carried out fault injection analyses. Fault injection campaigns have been executed using the fault platform reported in section V. The Zynq-7000 AP-SoC has been used as the hardware platform. The evaluated benchmark application is the CORDIC IP provided by the

Vivado IP Library. The analyzed designs include both plain and TMR-hardened versions implemented with and without traditional and domains-based IDFs. SEU in configuration memory is the fault model emulated during fault injection tasks. The reliability analyses have been compared to quantitively measure benefits introduced by the traditional and domains-based isolation flows.

A. Benchmark designs

As an application under test, we developed a hardware-accelerated system to be executed on the Zynq-7000 platform. The Xilinx CORDIC IP Core is the hardware-accelerated core implemented on the programmable hardware. The CORDIC IP Core is a widely adopted module in aerospace applications, where it is used for implementing transcendental functions and digital signal processing. The CORDIC IP Core is controlled by a software routine running on the Cortex-A9 processing system. Communication between the software routine and hardware modules is implemented through AXI4 Interconnection Cores. Data transfers are implemented using AXI DMA [14]. When the software routine is triggered by the fault injection platform running on the host computer, it stimulates the cores on the PL and evaluates if they are working correctly. In detail, the software routine provides a test vector to the CORDIC IP Core, compares the results of the hardware computation with the expected ones, and sends the experiment report to the results collector module running on the host computer. The design to be implemented on the programmable hardware consists of three AXI cores for communication purposes and a computational core (i.e., CORDIC). The IP Core is connected to the processor system of the SoC through the AXI Interface. A hardened version of the benchmark circuit has been designed using TMR. The CORDIC Core and its communication interfaces have been replicated three times. Each replication can be accessed by the PS through the AXI Interconnect module. The software routine votes the final results based on the output obtained by the three replicas.

B. Errors classification

The software routine running on the processor system stimulates the cores on the programmable logic. The obtained results are compared with the golden results to detect misbehaviors. If a mitigation approach based on replication is applied, the software running on the processor system runs the computation on each replicated module. The results of the cores are voted and compared to each other to correct or detect errors. The misbehaviors resulting from the fault injection campaigns have been classified into four categories:

1. *Data Unavailability (DU)*: we defined data unavailability when it is not possible to receive any results from the PL, usually due to faults affecting the communication modules.
2. *Silent Data Corruption (SDC)*: silent data corruption occurs when the results obtained by the PL have errors, but they are detectable only through comparison with the expected results (i.e., there is no cores replication or the voting process elected the wrong result).
3. *Recoverable Data Corruption (RDC)*: it occurs when different results are returned by the cores, but the correct results are recovered through voting.
4. *Detectable Data Corruption (DDC)*: it occurs when different results are returned by the cores and it is not possible to vote a result (e.g., in a TMR design, two modules return two different results and the third one is unavailable).

C. Evaluation of IDF benefits on plain benchmark design

Two versions of the plain benchmark design (i.e., without TMR) have been implemented using standard design flow and the state-of-the-art IDF. The reliability of the two designs has been evaluated through a fault injection campaign, emulating SEUs in the configuration memory. In the design implemented using state-of-the-art IDF, the block in the higher level of hierarchy (i.e., Zynq PS, AXI DMA, AXI Interconnect, and CORDIC) have been selected to be placed, isolated as required by IDF application notes.

TABLE I. ESSENTIAL BITS OF STANDARD AND IDF CONFIGURATIONS

Design	CM bits [#]	EB [#]	EB in CM [%]
Standard	32,345,856	717,873	2.22
IDF	32,345,856	810,181	2.50

The SEU fault model has been evaluated for each design through two different fault injection campaigns. Each campaign consists of 10,000 fault injections affecting the EB of the benchmark design implemented with and without state-of-the-art IDF. Table I describes the number of EB in each configuration and their percentage over the CM bits.

TABLE II. DISTRIBUTION OF ERRORS FOR STANDARD AND IDF DESIGNS CONSIDERING 10,000 INJECTIONS

Error Type	Implementation Flow	
	Standard	IDF
DU [#]	103	97
SDC [#]	194	148
Total [#]	297 (2.97%)	245 (2.45%)

The campaign resulted in an error rate of 2.97% for the design without IDF and 2.45% for the design implemented using IDF. The distribution of the errors is reported in Table II.

TABLE III. RESOURCES UTILIZATION FOR STANDARD AND IDF DESIGNS

Resources	Implementation Flow			
	Standard		IDF	
	Used [#]	Utiliz. [%]	Used [#]	Utiliz. [%]
LUTs	3,257	6.16	3,539	6.65
Flip-Flops	4,196	3.94	4,555	4.28
Memories	5	3.57	5	3.57

Using IDF, we reduced the error rate by 17.51%, acting only on the design placement constraints. The overhead in terms of utilization is reported in Table III. As can be observed, the amount of additional resources is negligible with respect to the available ones. However, IDF design requires about 10% more flip-flops and LUTs compared to the standard one, which can result problematic for more complex circuits.

D. Isolation policies for redundant design

In order to evaluate the benefits introduced by IDF for replicated designs, we carried out additional fault injection analyses on the TMR version of the benchmark design. In particular, we evaluated the reliability of the circuit resulting from the standard design flow, proposed domains-based IDF, and non-domains-based IDF. Please note that even if the benchmark design under test is very small (i.e., less than 7%

of resource utilization), it has not been possible to implement the state-of-the-art IDF. Indeed, the number of modules to isolate when TMR is applied makes it unfeasible to satisfy the isolation constraints.
The analyzed benchmark implementing TMR is described as follows:

- *Standard (unconstrained) configuration*: this benchmark has been implemented without IDF constraints, thus the modules are not isolated in this version. The block scheme of the modules at the higher level of the hierarchy is presented in Fig. 3.

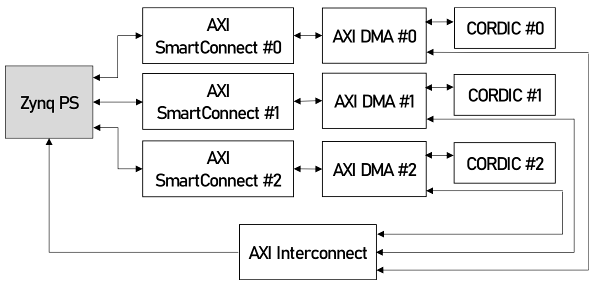

Fig. 3. Block scheme of the standard design.

- *Domains-based IDF configuration*: this design implements the domains-based isolation flow. The modules of a domain are grouped together in a block. The blocks are isolated using IDF. A single isolated block is composed of an AXI SmartConnect block, AXI DMA, and the CORDIC IP, as represented in Fig. 4.

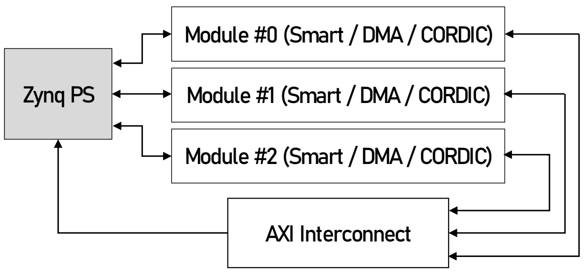
Module #0 (Smart / DMA / CORDIC)
Zynq PS
Module #1 (Smart / DMA / CORDIC)
Module #2 (Smart / DMA / CORDIC)
AXI Interconnect

Fig. 4. Block scheme of the domains-based isolation.

- *Non-domains-based IDF configuration*: this last isolation pattern, represented in Figure 5, couples together modules by task. AXI SmartConnect, DMA, and CORDIC blocks of the different domains are grouped between them. IDF is applied to these groups. This configuration has been proposed to evaluate the benefits of using the proposed domain-based aggregation policy with respect to an aggregation policy aiming only to minimize the number of modules to be placed, without taking into account the concept of domains.

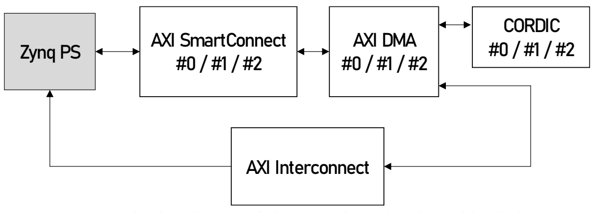

Fig. 5. Block scheme of the non-domains-based isolation.

Both of the IDF configurations isolate singularly the AXI Interconnect Module as it is recognized as a weak point of the design [18][19].

E. Evaluation of domains-based IDF benefits on TMR

The fault injection campaigns consist of 10,000 injections emulating SEUs in configuration memory, affecting the essential bits of different versions of the TMR-hardened benchmark circuit implemented using different design flows. The total number of CM bits and the EB for each design has been reported in Table IV.

TABLE IV. EB OF STANDARDS, DOMAINS-BASED, AND NON-DOMAINS-BASED CONFIGURATIONS

Design	CM bits [#]	Essential Bits [#]	EB in CM [%]
Standard TMR	32,345,856	2,811,321	8.69
Domains-based	32,345,856	2,930,999	9.06
Non-domains-based	32,345,856	3,002,114	9.28

Concerning the resource utilization, we observed a slightly higher requirement of logic resources when adopting IDF, similarly to what was obtained with the plain benchmark. In particular, IDF needs almost 1.5% of LUTs and 2% more FFs than the standard configuration when using IDF, as is reported in Table V.

TABLE V. RESOURCES UTILIZATION FOR STANDARD AND ISOLATED DESIGNS

Resources	Implementation Flow					
	Standard TMR		Domains-based		Non-domains-based	
	Used [#]	Utiliz. [%]	Used [#]	Utiliz. [%]	Used [#]	Utiliz. [%]
LUTs	12,878	24.21	13,064	24.56	13,204	24.82
Flip-Flops	17,706	16.64	17,713	16.65	18,037	16.95
Memories	15	10.71	15	10.71	15	10.71

We carried out the fault injections randomly targeting the EB of the designs. Due to the definition of EB, not all the injections will affect bits programming the used resources of the design. As matter of fact, only some of them will cause an error in the output of the application. This can happen as a result of a fault injected in the used resources or the activation of an unused resource that leads to a conflict. Considering the three possible configurations we observed the following results:

- *Standard configuration*: in this configuration, we detected a percentage of 5.37% faulty behaviors. Of these, 50.47% were RDCs, 29.61% DDCs, 15.27% SDCs, and 4.65% of DUs.
- *Domains-based IDF configuration*: in this case, the fault injection campaign produced 2.89% of faulty outputs: in particular, 65.74% are RDCs, 30.10% DDCs and 4.16% consists in the DUs. No SDCs have been detected.
- *Non-domains-based IDF configuration:* the experiment resulted in a 3.13% of error rate. In detail, we observed 45.37% of RDCs, 32.59% of DDCs, 2.87% of SDCs, and 19.17% of DUs.

The collected data are reported in detail in Table VI.
Comparing both of the configurations with the unconstrained design, it can be observed that due to the IDF implementation, the total error rate is slightly dropped with focus on the silent errors (no SDC and 0.09% of the total with the domains-based and non-domains-based configurations, respectively).

TABLE VI. DISTRIBUTION OF ERRORS FOR THE STANDARD TMR, DOMAINS-BASED AND NON-DOMAINS-BASED CONFIGURATIONS FOR 10,000 SEUs

Error type	Implementation Flow		
	Standard TMR	**Domains-based**	**Non-domains-based**
RDC [#]	271	190	142
DDC [#]	159	87	102
SDC [#]	82	0	9
DU [#]	25	12	60
Total [#]	537	289	313

The domains-based design produced the lowest error rate as well as the highest RDC ratio among the analyzed implementations. Such achievements are also supported by the absence of SDCs, which represent the worst possible behavior due to their undetectability. The non-domains-based implementation also brought the decrease of SDC with respect to the standard design. However, this configuration appeared to be very sensitive to the DU, which occurred more than three times with respect to the domains-based case. This is likely due to its by-task aggregation, where a fault in one of the communication modules propagates to the communication infrastructure of all the domains. Figure 6 compares the error classification resulting from the three implementation methodologies.

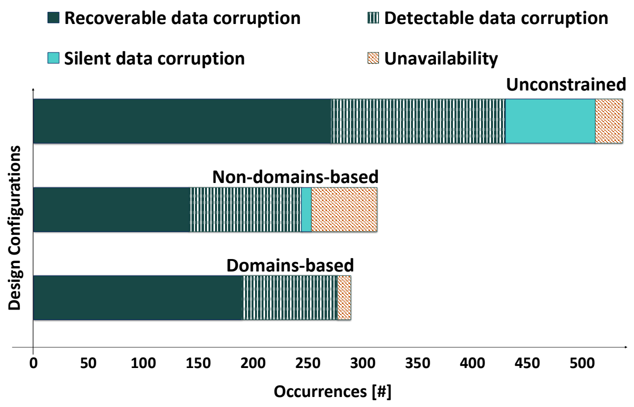

Fig. 6. Distribution of errors of 10,000 fault injection campaigns on standard, intra-domain, and inter-domain configurations.

We have performed further analysis on the causes of the Data Unavailability errors affecting the three designs. In particular, we investigated the contribution of the AXI Interconnect module to these errors compared to the other isolated regions of the designs. Using the PyXEL framework, we identified the physical resources affected by the faults resulting in DU errors. Then, retrieving which module is associated with the physical resource, the DU errors have been grouped in two categories: *AXI Interconnect-fault* (AI-F) and *domain-fault* (D-F). The AI-Fs occurs when the fault injection resulting in DU error targets a resource of the AXI Interconnect Module. D-F happens when the bit-flip corrupts a memory cell programming a resource not used by the AXI Interconnect Module. Table VII reports the results of DU categorization.

TABLE VII. DATA UNAVAILABILITY ANALYSIS

Category	Implementation Flow		
	Standard TMR	**Domains-based**	**Non-domains-based**
AI-F [#]	13	11	54
D-F [#]	12	1	6
Total [#]	25	12	60

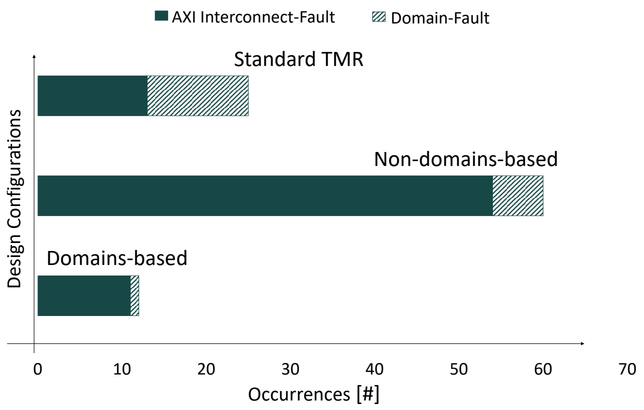

Fig. 7. The distribution of errors due to the unavailability of data.

The analysis we performed has brought out that the source of data unavailability when IDF is applied is the AXI Interconnect in about 90% of the experiment. Since the normalized values of AI-F for non-domain- and domain-based are comparable, it is likely the high number of DUs observed in non-domain-based IDF design are due to random fault injection produced a higher number of faults in the AXI Interconnect module. However, it is interesting to notice how in standard TMR, where AXI Interconnect is not isolated with respect to the other modules, the contribution of the faults injected in the AXI Interconnection to DU is much lower. It is reasonable to suppose that errors not affecting AXI Interconnect when isolation is not applied, more easily propagate to AXI Interconnect producing DUs. This effect is probably prevented when IDF is applied, making faults in the AXI Interconnect module the main cause of DUs. The obtained results are summarized in Fig. 7, where we identified the number of errors due to AI-F and D-F in the three configurations.

VII. CONCLUSIONS

This paper focuses on the effectiveness of IDF for redundant designs implemented on programable systems on a chip, proposing a set of design guidelines in the case of complex systems unable to implement state-of-the-art IDF, due to topological issues. The proposed domains-based IDF has been proved capable of mitigating radiation-induced faults for mid-to-high complex designs through different fault injection campaigns. Moreover, it has shown an increase in the effectiveness of TMR. In particular, Domains-based IDF prevented all Silent Data Corruption errors and increased the recoverable errors by about 33%. Future works perspectives include the development of a placement algorithm to automatize the floorplanning process following the domains-based IDF design rules and the application of IDF to SoPC-based computational clusters.

REFERENCES

[1] H. Asadi, et al., "Soft Error Susceptibility Analysis of SRAM-Based FPGAs in High-Performance Information Systems," in *IEEE Transactions on Nuclear Science*, vol. 54, no. 6, pp. 2714-2726, Dec. 2007.

[2] B. Du et al., "Radiation-induced Single Event Transient effects during the reconfiguration process of SRAM-based FPGAs," in *Microelectronics Reliability*, vol. 100-101, Sept. 2019.

[3] B. Du et al., "Ultrahigh energy heavy ion test beam on Xilinx Kintex-7 SRAM-based FPGA," *in IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1813-1819, July 2019.

[4] R. C. Baumann, Landmarks in terrestrial single-event effects," *Nuclear and Space Radiation Effects Conference*, San Francisco, USA, 2013.

[5] W. Wang, et al., "The research of FPGA reliability based on redundancy methods", in *Internation Conference on Computer Science and Network Technology*, Harbin, China, 2011, pp. 1608-1611.

[6] Z. Wang, et al., "The reliability and availability analysis of SEU mitigation techniques in SRAM-based FPGAs" in *European Conference on Radiation and Its Effects on Components and Systems*, Brugge, Belgium, 2009, pp. 497-503.

[7] W. Lie and W. Feng-yan, "Dynamic Partial Reconfiguration in FPGAs," in *Third International Symposium on Intelligent Information Technology Application*, Nanchang, China, 2009, pp. 445-448.

[8] S. Azimi and L. Sterpone, "Digital Design Techniques for Dependable High Performance Computing," *2020 IEEE International Test Conference (ITC)*, Washington, DC, USA, 2020, pp. 1-10.

[9] L. Gantel, et al., "Module relocation in Heterogeneous Reconfigurable Systems-on-Chip using the Xilinx Isolation Design Flow," in *International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico, 2012, pp. 1-6.

[10] J. Rettkowski, et al., "RePaBit: Automated generation of relocatable partial bitstreams for Xilinx Zynq FPGAs," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, Mexico, 2016, pp. 1-8.

[11] K. Pham, et al., "IPRDF: An Isolated Partial Reconfiguration Design Flow for Xilinx FPGAs," in *IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, Hanoi, Vietnam, 2018, pp. 36-43.

[12] M. Wirthlin, "High-reliability FPGA-based systems: space, high-energy physics, and beyond," in *Proceedings of the IEEE*, 2015.

[13] A. Sanchez, et al., "Evaluation of TMR effectiveness for soft error mitigation in SHyLoC compression IP core implemented on Zynq SoC under heavy ion radiation," in *IEEE Internation Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, Noordwijk, Netherlands, 2019.

[14] M. Ceschia et al., "Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs," *in IEEE Transactions on Nuclear Science*, vol. 50, no. 6, pp. 2088-2094, Dec. 2003.

[15] Xilinx, "Vivado Isolation Verifier," Xilinx, 2020.

[16] L. Bozzoli, et al., "PyXEL: An Integrated Environment for the Analysis of Fault Effects in SRAM-Based FPGA Routing," in *International Symposium on Rapid System Prototyping (RSP),* Turin, Italy, 2018, pp. 70-75.

[17] Xilinx, "Soft Error Mitigation Controller v4.1 Product Guide," Xilinx, 2018.

[18] Xilinx, "AXI DMA v7.1 LogiCORE IP Product Guide," Xilinx Product Specification, 2019.

[19] C. De Sio, et al., " On the analysis of radiation-induced failures in the AXI interconnect module," in *Microelectronics Reliability*, pp. 243-254, 2020.

[20] C. De Sio et al., “On the Evaluation of SEU Effects on AXI Interconnect Within AP-SoCs,” in *2020 Architecture of Computing Systems – ARCS*, 2020.