SEU Mitigation on SRAM-based FPGAs through Domains-based Isolation Design Flow

SEU Mitigation on SRAM-based FPGAs through Domains-based Isolation Design Flow

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

***Abstract*—We developed a domain-based isolation design flow for the mitigation of SEU effects on SRAM-based FPGAs. Fault injection experimental analysis on TMR circuits mapped on AP-SoC demonstrates an improvement of 44% versus traditional mitigation techniques.**

***Keywords*—Isolation Design Flow, SEU, SRAM-based FPGA.**

I. Introduction

Nowadays, All-Programmable System-on-Chips (AP-SoCs) and Field-Programmable Gate Arrays (FPGAs) are an attractive solution for safety critical applications such as avionics, aerospace and automotive. Thanks to their hardware programmability, the designers can meet high performance and strict requirements with relatively low costs, power consumption, and time-to-market. The main limitations of the usage of Commercial-of-the-Shelf (COTS) SRAM-based programmable hardware platforms in mission-critical applications is the susceptibility of these devices to Single-Event Upsets (SEUs). Therefore, the mitigation of the SEU effects is the main challenge for adopting SRAM-based FPGAs in safety-critical applications. Several SEU mitigating techniques have been proposed for SRAM-based FPGAs and AP-SoCs. The traditional techniques applied on such devices are based on hardware redundancy [1][2].

The Isolation Design Flow (IDF) is proposed as a promising approach to improve fault containment and analysis, acting directly on the design placement. The key idea behind IDF is to isolate modules during the floorplanning to build more reliable systems. Indeed, module isolation eases fault detection in the single modules and limits the propagation of faults between them. The isolation is obtained by adopting an extensive set of Design Rule Constraints (DRCs) that can easily lead to routing congestion. As a result, it increases the complexity in 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 analyses the benefits introduced by the IDF in terms of the application reliability. The authors in [3] 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 [4] suggests a novel method to ease the bitstream relocation in presence of IDF constraints. Eventually, the authors in [5] implement also off-chip trusted communication with the partial reconfigured section. Even though the mentioned methods are all highly effective, currently, the FPGA commercial design tools (e.g., Xilinx Vivado) do not support any partial reconfiguration integrated with IDF. Therefore, the main challenge is to interface with external tools and the related high time required.

In this paper, we have introduced the benefits of applying IDF in increasing the reliability of the system. Moreover, to overcome the complexity of applying IDF to Triple Modular Redundant (TMR) of the circuits, we proposed a methodology with the goal to reduce the complexity of the floorplanning stage. This methodology is applied to the TMR version of the CORDIC benchmark, and the results obtained on the reliability improvement of TMR CORDIC are reported in experimental results.

This paper is organized as follows. Section II gives an overview of the Isolation Design Flow. Section III is describing the domains-based isolation policy while the fault injection environment is explained in Section IV. The obtained results are described in Section V. Finally, Section VI contains conclusions.

II. Background

The Isolation Design Flow is a design technique that acts on the physical isolation placement of the resources on the FPGA layout to assure the non-interface of functions within the same chip, thus preventing the fault propagation between modules.

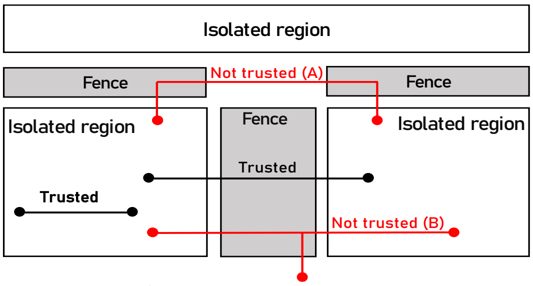

Fig. 1. Examples of routes: *A* and *B* are representing non-trusted routes.

To guarantee the isolation between modules, each module must have its own hierarchical instance in the hardware description of the netlist (e.g., HDL), imported in the implementation tools. The isolation mechanism relies on the concept of fences. A fence is a set of contiguous rows/columns of unused resources separating two isolated regions. The requirements on fences width in terms of unused resources is depending on the specific device. The on-chip communication must be implemented using trusted routes. The routing 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 represents an example of the fences and routes between isolated regions. In order to respect the constraints regarding fences and trusted routes, IDF requires an elaborated floorplanning phase that is supported partially by the vendor tools. Therefore, a manual floorplanning process is required. However, when the number of modules to be placed increases, the manual floorplanning process requested for acheving isolation becomes highly challenging to implement or, in the worst cases, impossible. To overcome this challenge, grouping a module under a higher hierarchical level could present a possible solution. Therefore, a trade-off between the isolation between modules and the complexity and feasibility of the design placement and routing is required.

III. Domains-Based Isolation Design Flow

The application of the IDF to the replicated modules of a design is a high complexity task that typically leads difficult to reach all the design constraints required. Therefore, we are proposing a set of design practical rule to reduce the floorplanning complexity during the isolation design flow, when replication-based approaches such as TMR are used.

The proposed approach is based on coupling together different modules within the same isolated region, thus reducing the number of blocks to be isolated. However, an 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 do not affect more than a single replicated computational unit are filtered by the voter, the underlying idea is to prioritize the isolation between modules 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.

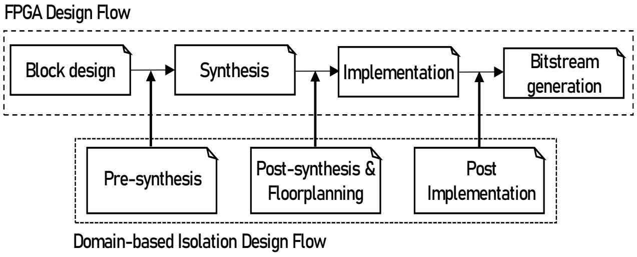

Fig. 2. Domains-based IDF integrated with the FPGA design flow.

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 illustrated in Fig. 2. It consists of the four tasks listed below:

- *Pre-Synthesis*: in this phase, 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. It is important to avoid grouping together modules belonging to different domains of the same voter. Clock signals or other inter-region signals must be declared in the placement constraint file to be, allowed 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*). 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*: the implementation is verified by manufacturer tool such as the Vivado Isolation Verifier (VIV) [5] built-in tool is used to verify the correct implementation of IDF between the isolated blocks. It generates a report on possible misplacements of blocks or fences and physical overlay of modules.

IV. The SEU Injection environment

In order to evaluate the benefits introduced by domains-based IDF and compare the reliability of study cases while domains-based IDF is applied, an SEU-oriented fault injection environment is developed. The environment automatizes both the faults generation and faults evaluation tasks, as well as results collection and analysis. Fig. 3 represents the developed fault injection environment.

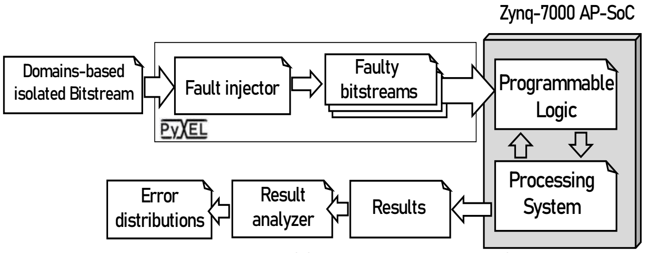

Fig. 3. The scheme of fault injection analysis flow.

The fault injection mechanism starts with the usage of our in-house developed PyXEL framework [7] which is a Python library for the analysis of faults in FPGA, that 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 [8].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.

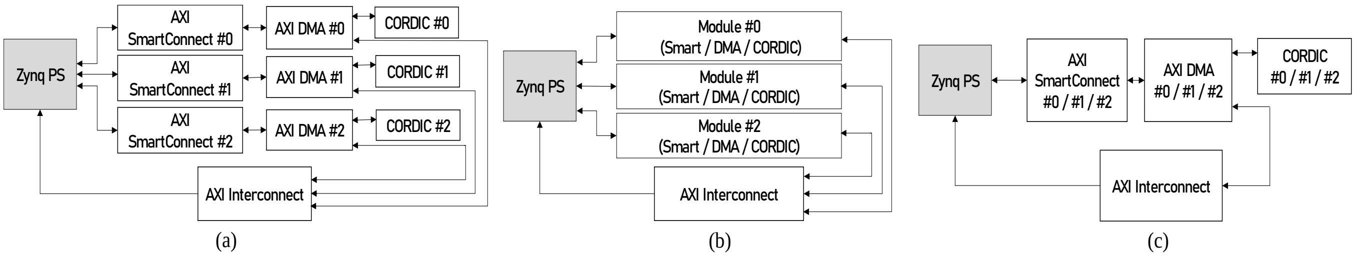

Fig. 4. The Block Scheme of: (a) unconstrained design (b) Intra-domain IDF (c) Inter-domain IDF

V. Experimental Analysis and Results

For evaluating the benefits introduced by domain based IDF, we carried out fault injection campaignes, executed using the fault platform reported in section IV. 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 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. When 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.

The analyzed designs include both TMR-hardened versions implemented with and without 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 domains-based isolation flows.

A. Domain-Based IDF Reliability Evaluation

In order to find the best isolation policy, an analysis of the optimal isolated set of blocks has been performed. To elaborate more, the authors in [9] highlight the AXI Interconnect module as a vulnerable area of the designs. Therefore, the AXI modules have been chosen for isolation in order to prevent the propagation of the faults within the isolated module. Eventually, aiming to ease the placement of the modules, we maximized the number of IPs in a single isolated region in order to reduce the number of pblocks to be placed. Two possible configurations met the requirements discussed above and therefore were selected for further analysis: intra-domain and inter-domain. The first one focuses the isolation on the single domain comprising CORDIC, AXI SmartConnect, and DMA blocks, represented in Fig. 4.b while the inter-domain configuration, represented in Fig. 4.c, couples together AXI SmartConnect, DMA, and CORDIC blocks of the three different domains. Both the intra-domain and inter-domain configurations isolate singularly the AXI Interconnect module and the Zynq PS. For comparison purposes, we also implemented an unconstrained configuration, letting Vivado automatically manage the placement and routing of the design, imposing no isolation constraints, represented in Fig. 4.a. Eventually, we generated the golden bitstreams of the three designs in order to perform SEU fault injections and compare the output data.

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. The misbehaviours due to the fault injection campaigns have been classified into four categories: *Data Unavailability (DU)* happening when it is not possible to receive any results from the PL, usually due to faults affecting the communication modules, *Silent Data Corruption (SDC)* occurs when the results obtained by the PL have errors but they are detectable only through comparison with the expected results, *Recoverable Data Corruption (RDC)* arises when different results are returned by the cores, but the correct results are recovered through voting, *and Detectable Data Corruption (DDC)* occurs when different results are returned by the cores and it is not possible to vote a result.

C. Fault Injection Experimental Results

The golden bitstreams obtained in the previous section have been used to identify the EB. We identified such bits by analyzing the mask files provided by Vivado after the bitstream generation and mapping them to the CM bits. Regarding the original unconstrained design and the two chosen configurations for IDF, the total number of CM bits and the EB for each design have been reported in Table I.

Table I. Essential Bits of Unconstrained, Intra and Inter-Domain Configurations

Design	CM bits [#]	EB [#]	EB in CM [%]
Unconstrained	32,345,856	2,811,321	8.69
Intra-domain	32,345,856	2,930,999	9.06
Inter-domain	32,345,856	3,002,114	9.28

Table II reports a comparison regarding resource utilization between the unconstrained, intra-domain and inter-domain-based designs. As it can be observed, IDF need almost 1.5% of LITs and 2.5% more than the standard configuration for domain-based and non-domain based IDF.

Table II. Resources Utilization for Standard and Isolated Designs

Designs	Unconstrained TMR		Intra-domain IDF		Inter-domain IDF	
Resources	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 performed 10,000 fault injections on EB, randomly. Therefore, not all the chosen bits are the property of the used resources of the design. This condition mimics the typical radiation environment where bit-flips can be generated in any location of the FPGA's configuration memory. As matter of fact, some of them will trigger or modify unused resources of the device. If the bit-flip targeted a used cell or connection, then a faulty output will be collected and classified. Table III report the overall error rate due to 10,000 SEU injections while the error distribution is represented in Fig. 5.

TABLE IIII. SEU ERROR RATE REPORT

Designs	SEU Injection [#]	SEU Error Rate [%]
Unconstrained	10,000	5.37
Intra-domain	10,000	3.13
Inter-domain	10,000	2.89

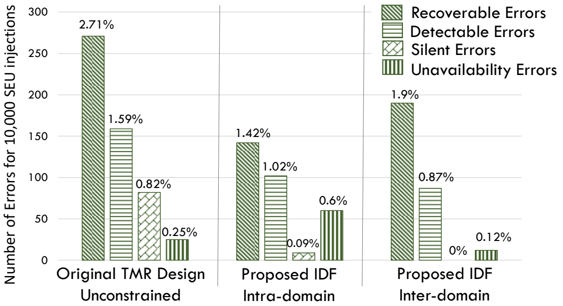

Fig. 5 The distribution of SEU-induced errors of TMR designs with unconstrained, intra and inter-domain IDF configurations.

The proposed approach is reducing the number of SEU-induced errors affecting the design by about 44% with respect to the original unconstrained design. In particular, the proposed IDF approach adopting the inter-domain configuration is nullyfing the number of silent error and increasing the number of recoverable errors. This result enables the applicability of self-repairing techniques. Comparing intra-domain and inter-domain configurations with the unconstrained design, it can be observed that due to the IDF implementation, the total error rate is slightly dropped with a focus on the silent errors (no silent data corruption and 0.09% of the total with the intra-domain and inter-domain configurations, respectively). Although the inter-domain design has witnessed the lowest recoverable data corruption among the three, it has not proved robust to the unavailability, which happened more than three times with respect to the intra-domain case. This is most likely due to its serial configuration, where a fault in any of the three modules will cause a cascade breakdown in the data flow.

TABLE IV. SEU ERROR RATE REPORT ON AXI MODULE

Designs	SEU Injection [#]	SEU Error Rate [%]
Unconstrained	10,000	0.45
Intra-domain	10,000	0.14
Inter-domain	10,000	0.5

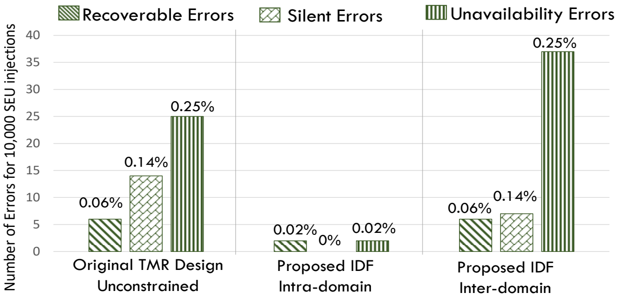

Fig. 6. The distribution of errors due to 10,000 SEU injections on the AXI module in the different design configurations.

We further investigated which is the occurrence of cross-domain faults against the AXI Interconnect module faults. In detail, we implemented a cycle counter within the PS environment, which can recognize the unavailability due to a faulty behaviour of one of the three isolated domains. We identified two possible kinds of behaviour: *AXI Interconnect-fault* (AI-F) occurs when the three outputs are not detected by the voter due to corruption of the communication module, and *domain-fault* (D-F) happens when the injected SEU corrupts the data flow within the isolated domain, causing the unavailability of a single output.

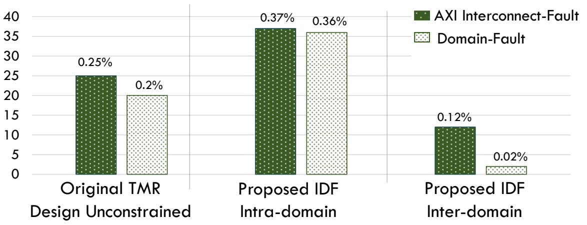

Fig. 7. The SEU-induced error distribution for domain and AXI interconnect Faults for 10,000 SEU injections.

The unavailability of data scenario is always caused by AI-F, where all three outputs are not collected by the system. It represents the only case where AI-F happens while D-F can occur in recoverable, silent, or unavailability and data corruption errors. Table IV represents the total error rate on AXI module while Fig. 6 reports error distribution due to the unavailability of domain or AXI Interconnect. Fig. 7 represents the identified number of errors due to AI-F and D-F in the three configurations. The analysis shows that 85.71% of the unavailable scenarios are due to AI-F in the intra-domain design, highlighting strong reliability of the isolated domain against unavailability compared with the other configurations.

VI. CONCLUSIONS

This work proposed a domains-based IDF design flow to easily mitigate radiation-induced SEUs in presence of mid-to-high complexity systems in SRAM-based devices. The optimal set to be isolated has been found between two different configurations by means of fault injection campaigns. Intra-domain isolation has shown to decrease the errors due to the unavailability of the single domain, thus focusing the failures mainly on the AXI Interconnect module.

REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

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

[7] C. De Sio, 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)*, 2018, pp. 70-75.

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

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