POLITECNICO DI TORINO
Repository ISTITUZIONALE

Layout-oriented Radiation Effects Mitigation in RISC-V Soft Processor

Original
Layout-oriented Radiation Effects Mitigation in RISC-V Soft Processor / Vacca, E., De Sio, C., Azimi, S.. - ELETTRONICO. - (2022), pp. 215-220. (19th ACM International Conference on Computing Frontiers 2022 Torino May 2022) [10.1145/3528416.3530984].

Availability:
This version is available at: 11583/2966363 since: 2022-06-09T12:21:14Z

Publisher:
ACM

Published
DOI:10.1145/3528416.3530984

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript

(Article begins on next page)

22 June 2026

Layout-oriented Radiation Effects Mitigation in RISC-V Soft Processor

Invited Paper

Eleonora Vacca
Department of Control and Computer Engineering
Politecnico di Torino
Turin, Italy
eleonora.vacca@polito.it

Corrado De Sio
Department of Control and Computer Engineering
Politecnico di Torino
Turin, Italy
corrado.desio@polito.it

Sarah Azimi
Department of Control and Computer Engineering
Politecnico di Torino
Turin, Italy
sarah.azimi@polito.it

ABSTRACT

Last decade, the RISC-V soft processor has become popular due to the benefits such as transparency and availability of hardware implementations on reconfigurable devices such as SRAM-based FPGAs. However, these devices are highly sensitive to high-energy particles. In this work, we propose an implementation methodology that ranges from the hardening method applied at the Register Transfer Level (RTL) to the optimization of layout techniques acting on the place & route of the design. As a case study, the TMR-hardened Arithmetic Logic Unit (ALU) of the RISC-V soft processor implementation was taken and its reliability under different design layouts has been analyzed using fault injection campaigns. Experimental results show that the design reliability can be improved by applying ad hoc layout customization..

CCS CONCEPTS

Hardware~Robustness~Safety critical systems

KEYWORDS

Floorplanning, RISC-V, Reliability, SRAM-based FPGA, Fault Injection, Routing, Reconfigurable devices.

ACM Reference format:

Eleonora Vacca, Corrado De Sio and Sarah Azimi. 2022. Layout-oriented Radiation Effects Mitigation in RISC-V Soft Processor: Invited Paper. In *Proceedings of ACM Computing Frontiers (CF'22). ACM, May 17-19, Turin, Italy, 6 pages.* https://doi.org/10.1145/3528416.3530984

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, republish, post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CF'22, May 17–19, 2022, Torino, Italy
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9338-6/22/05…$15.00
https://doi.org/10.1145/3528416.3530984

1 Introduction

Last decade, the RISC-V soft processor has become interesting. Its open-source nature, accompanied by a modular and highly intuitive design, made it attractive for various fields, including aerospace. Considering space applications, the reliability of electronic components is jeopardized by the high presence of radiation. Thus, consideration must be given to the effect of high-energy particles on functional failures of the system and suitable mitigate techniques [1]. Reliability becomes even more critical when designs are implemented on SRAM-based programmable logic devices [2][3][4] since configuration memory is an extremely radiation-sensitive element [5][6] as it stores the binary information to configure the logic to implement the circuit. Hence, a radiation-induced alteration of such memory can cause a structural change to the implemented design [7][8][9].

One of the most common approaches to ensuring a working programmable logic device even in case of a failure is the hardening-by-design method, typically consisting of the adoption of redundancy [10][11][12] that can be applied at different levels. Nonetheless, hardening techniques are often not sufficient to achieve the required level of reliability which in turn can be increased or decreased by the layout solution adopted. The P&R algorithms typically supported by commercial FPGA design tools aim to realize feasible, performant but not necessarily reliable solutions. The level of reliability is certainly dependent on the application context of the design, which is known a priori by the designer and not the tool. Hence, the designer may decide to adopt a certain implementation strategy to increase the reliability of the design, but this is not reflected in the layout solution arranged by the commercial tool. In this context, it is certainly helpful to have an intuitive way to make any changes the designer deems necessary on the placement and/or routing of the design. Thus, having the ability to generate a custom layout that reflects the design specifications imposed by the user.

In this paper, we present two layout-oriented approaches, one for placement and the other one for routing with the goal to increase the reliability of the design. The developed tools receive customization directives from the user and generate design constraints to be integrated into the commercial design flow. They

have been applied to the hardened design of the RISC-V Arithmetic Logic Unit (ALU) extracted from the RI5CY core part of the PULPissimo [15] project. Firstly, the ALU has been hardened by applying the Triple Modular Redundancy (TMR) technique. Secondly, its reliability has been evaluated under different layout conditions, realized with the developed tools, by performing fault injection on the various configurations of design.

The experimental results have shown that layout is affecting the reliability of a design, especially by compromising the applied error mitigation technique. Considering TMR, this must be accompanied by customization of both placement and routing, applying isolation based on TMR domains.

2 Related Works

The P&R algorithms are one of the most computational challenges in the field of Computer-Aided Design (CAD) tools [14]. The placement customization approach is today implemented in several commercial tools, and it relies on manual placement techniques supported by the graphic interface, as used in [16]. Other proposed techniques provide full floorplanning algorithm solutions as in [19][20] that are complex or rely on partial reconfiguration [17][18][19].

As much as these approaches are valid, they are tough to implement for those who want to change the layout in a few steps and have little knowledge of what is behind it. On contrary, the support of a graphical interface in which the user can choose the physical position of modules could be the preferred approach. Therefore, commercial tools offer an instrument for floorplanning [16], that consists in defining the so-called placement block.

The routing process is another critical step in the FPGA design flow. Interconnect resources occupy predetermined locations in the FPGA area, thus, drastically reducing flexibility. In addition, the associated delay contributions are much greater than those of the logic, making routing one of the main performance-dependent issues in FPGAs. Some of the most renowned routing algorithms include the Maze routing algorithm [22] which aims to find the shortest path between two points while avoiding already used routing resources. However, it performs routing by looking for the local optimum, not considering that the path found may block the routing of the following nets. Therefore, the efficiency of the algorithm depends on the order in which the nets are processed. The algorithm proposed at [23] is based on the Maze one but with the possibility of also using resources already used.

Moreover, Vivado Design provides a tool for manual routing that allows you to customize one net routing path at a time, choosing each time the next segment of the path [21].

In this paper, we present methodologies to automate the routing and placement customization process compared to the commercial tool approaches, limiting user intervention to entering customization directives

3 Background

FPGAs are typically organized in a bi-dimensional array of Configurable Logic Blocks (CLBs) that are connected using programmable interconnects grouped within Switch Boxes [13].

3.1 Logic Resources Organization

Although in most FPGAs the basic hardware resource is the CLB, in Xilinx devices the SLICE plays a key role in the placement algorithm. The physical resources are uniquely identified through the SLICEs they belong to; the SLICEs are organized in a matrix system of dimension M x N, where each of them is labeled through a pair of coordinates (i, j). Each SLICE contains four Look-Up Tables (LUT), some logic gates, eight flip-flops, and one carry logic block. The content of each SLICE defines a set of primitive types. A primitive type is a resource that is directly recognized by the Xilinx implementation software and that corresponds to a physical resource in the target FPGA.

This means that, given a design, any logic cell in the netlist will be mapped to one of these primitive types, depending on the functionality it implements. Once the primitive type has been defined for each netlist element, the Xilinx Vivado Design tool proceeds with the Implementation phase which is the process of assigning physical resources to the logical ones.

3.2 Interconnection Resources Organization

In Xilinx FPGAs, the CLBs are organized in a two-dimensional array structure, and between the rows and columns of this array, there are horizontal and vertical routing channels that allow exchanging data among CLBs, Input blocks, and Output blocks. Switch Boxes (SB) enclose programmable interconnections that allow switching between vertical and horizontal wires, enabling access to CLBs.

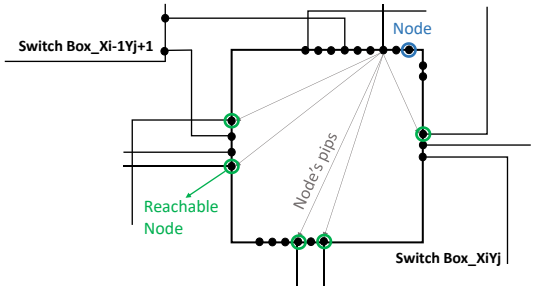


Figure 1: Switch Box view: node selection with all its reachable nodes.

In the nomenclature adopted by Xilinx, each Switch Box is associated with a pair of (x, y) coordinates, that follow the same trend as the coordinates related to CLBs. These coordinates uniquely associate each SB with a precise point in the space of interconnections, so the latter can be modeled as a matrix where each cell is a programmable interconnection resource.

The SB can be modeled as a black box with a set of nodes that are placed along its perimeter, as shown in Fig. 1. Each node has associated with some Programmable Interconnect Point (PIPs). A PIP is a switch that connects two wires, controlled by SRAM configuration cells. Given a node, its PIPs correspond to a set of reachable nodes belonging either to the same matrix (internal connection) and/or to other nearby matrices. In addition, each node is characterized by a direction and a displacement. Unlike PIPs, easily extractable from the device features, the other information

that outlines the characteristics of each node is not provided by the manufacturers. However, this is essential information to build a router. Thus, in this work, node characteristics were extracted by taking a generic net as an example and routing it using the manual routing wizard provided by the FPGA commercial design tool. For each segment of the example net, most of the possible nodes and connections have been tried and the information about the destination node, direction, and displacement have been collected.

4 The Proposed Place & Route Approach

The proposed approach aims to automatize both the relocation of modules and the customization of routing, compared to the methods proposed by commercial tools, keeping a user-friendly orientation. In the case of placement customization, the platform allows applying any type of constraint with a granularity of the single logic cell, while the routing tool allows the identification of a feasible routing path that reflects a user-defined constraint on multiple nets during the same run, drastically reducing the time effort of the manual routing approach.

4.1 The Placement Flow

The workflow of the developed placement platform is shown in Fig. 2. The starting point is the post-implementation design produced by the commercial tool. The physical description of the design and its properties, along with the user's directive on the hierarchy level of interest are used as input to the platform. The platform will proceed with extracting the modules consisting of the design and providing the modules list to the user. The user will indicate the placement customization guidelines and the modules to which they should be applied. The platform will continue with the relocation of the modules and, at the end of the process, will return a physical design constraints file containing the customizations defined by the user.

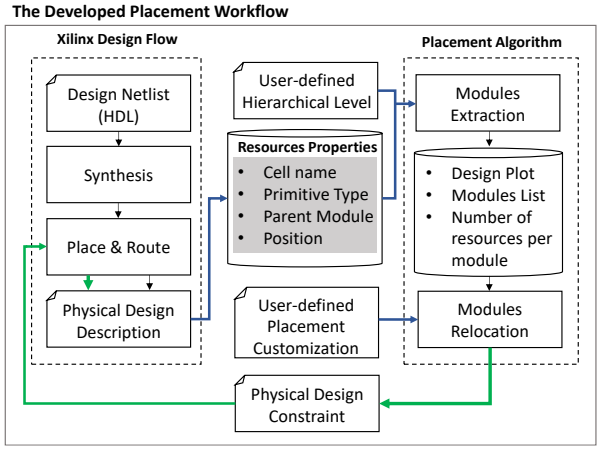


Figure 2: The scheme of developed placement platform workflow.

Going into further detail, the fundamental key to the functionality of the platform is the creation of an environment that emulates the architectural organization of the FPGA family under evaluation through a two-dimensional array of objects that reproduce the functionality of the SLICEs. Taking into account that any design can be seen as a list of primitive types, each implementing a cell of the netlist, the resource netlist is extracted in terms of a list of properties that characterize each resource used by the current design. Among all, those of interest is the physical location of the resource in the SLICE space, its primitive type (LUT, FF, CARRY, etc.), the logic cell it implements, and the module to which belongs.

The parent module name, which also appears in the cell name, is provided as a hierarchical path. Starting from the design name, going through the nested sequence of top entities and macro modules, up to the final module that has logic cells associated. This gives the possibility to choose the hierarchy level at which to perform the module extraction. By truncating the hierarchical path at a certain level, all cells that have the same sequence of entities in their name path up to the given level, regardless of what comes next, will be grouped under the same module. Therefore, the number of modules depends on the chosen hierarchical level. In particular, the higher the level, the more modules of the design will be interpreted as independent and separate, and it will be possible to apply different design constraints to each of them.

Once the extraction has been completed, this information is used to remap the design in the platform, without applying any change. The tool will provide visual feedback of the mapping, where the module visualization is performed according to the hierarchical level inserted. As an example, Fig. 3 shows the same design mapped in the platform with different hierarchical level selections.

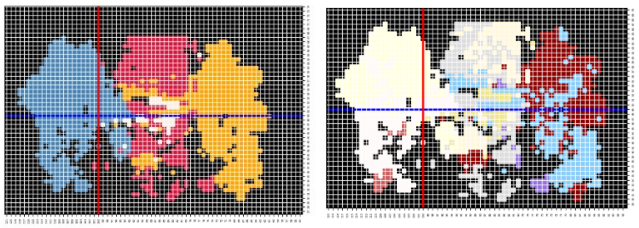


Figure 3: Design view on the placement platform with different hierarchical level selection: (a) hierarchy level equal to 4, (b) hierarchy level equal to 5.

In addition, for each extracted module, the tool indicates the number of resources (LUTs) used. After the mapping is performed, the user can impose placement constraints to customize the layout, indicating the module name and the perimeter coordinates in which their placements have to be confined. The placement will proceed to take all the primitive resources of the current design that match the module name in the associated cell name; the number of resources, previously counted, will be used to define how many SLICEs the module requires to be mapped, considering that each logic cell is mapped in a LUT and that a SLICE contains four LUTs.

Once the placement customization is performed for all the modules selected, the new placement is translated into a Tcl file which indicates a new position for each customized cell. To perform effective customization of the original unconstrained design, this file must be integrated into the commercial tool design flow. This is done by using it as a source file in the commercial tool

command line after the Synthesis phase, to force a fixed location for each cell, which then will be translated into a constrained location property used during the next implementation phase.

4.2 The Router

The matrix model of the interconnections space allows simplifying the routing problem by reducing it to a point-to-point connection in Euclidean space. The routing path is built by selecting one segment at a time, where the destination Switch Box, with the current chosen node, becomes the source for the next segment. Based on this model, the router interfaces with a matrix environment where each cell of the matrix is an object reflecting the properties of the Switch Box. Using the developed router to customize the commercial tools approach, the constraints should be set at the beginning, by linking them with the list of nets to which they should be applied. The constraints must be specified as coordinates (x_{min}, x_{max}, y_{min}, y_{max}) within which the path should be contained. In the range [x_{min}, x_{max}] the user must indicate a direction for the path which can be either vertical or horizontal.

```
1. read source_node, end_node
2. read axis, Xmin, Xmax, Ymin, Ymax

3. horiz_move = (source_node.x > end_node.x) ? westward : eastward
4. vert_move = (source_node.y > end_node.y) ? downward : upward
4. dir = (axis = horizontal) ? horiz_move : vert_move

5. blacklist = empty
6. used_nodes = empty
7. current_node = source_node

8. while (
9.      (axis = horizontal) and ( Xmin < current_node.x < Xmax ))
10.    or
11.    (axis = vertical) and (Xmin < current_node.x < Xmax))
12.    {
13.     nodes_list = nodes compatible with current_node to move in direction dir
14.     nodes_list = nodes_list - blacklist - used_nodes
15.     if nodes_list is empty
16.     {
17.        #NO NODE AVAILABLE FOR axis MOVEMENT
18.        nodes_list = nodes compatible with current_node to move on the
unconstrained axis
19.     }
20.     if nodes_list not empty
21.     {
22.        chosen_node = random node from nodes_list
23.        append chosen_node to used_nodes
24.        append current_node to path_list
25.        current_node = chosen_node
26.     }else{
27.        append source_node to blacklist
28.     }
29.     }
30. return path_list
```

Figure 4: The pseudocode of the developed Router algorithm in case of constrained movement.

The source node, destination node, starting and ending path coordinates, and the net name are extracted from the original routing of the post-implementation design. The router will move in the matrix space either at east, west, north, or south. Each movement has a set of nodes associated with it, common to all the Switch Boxes.

The working principle of the developed router consists in identifying a set of nodes compatible with the current source node that can satisfy the user-defined constraint and reduce the distance to the destination node. Since many nodes can satisfy these prerequisites, the choice of the specific node among the suitable ones is, at the moment, random. Once a node is selected, the router checks its availability in the corresponding Switch Box. If the check is positive, the router continues from the selected node. In case the node is not available, the router will choose another one from the list of proposed nodes. If the search for an available node ends with a negative result (either because all nodes in the list are used by other nets, or because the list is empty) then the router violates the direction constraint for that single stretch of the path.

However, it may happen that even in the non-preferred direction, the router is unable to find an available node. This indicates that the current node is a dead end, so the router proceeds to remove it from the path and restores the conditions before picking another possible node. Fig. 4 shows the steps that the router carries out when the user has defined a constrained direction for the range of values between x_{min} and x_{max}.

Since not all nodes have been classified in their behavior, it happens that the router is not able to find the path such that the distance between source and destination is zero yet keeping close. However, the routing can be left incomplete as Vivado's routing process will complete the routing path starting from the last constrained segment.

Once the router has completed the route assignment for all the nets, these are translated into a Tcl file. The file indicates the new assigned path for each customized net. To apply the customization, the file needs to be integrated into the commercial tool design flow at the implementation phase.

5 Application of the Place and Route Approach

The proposed place&route tools were used to evaluate how layout affects the reliability of a fault-tolerant design. For this purpose, we applied the TMR technique to the ALU of the RISCY core. Keeping the original interface of the ALU, a new RTL entity was created inside which three independent replicas of the original ALU core were instantiated, all fed with the same inputs, plus a majority voter. The voter evaluates if two of the three inputs are equal. If so, the common value will be provided as the final output of the macro-entity at the end of the processing.

5.1 The Test Environment

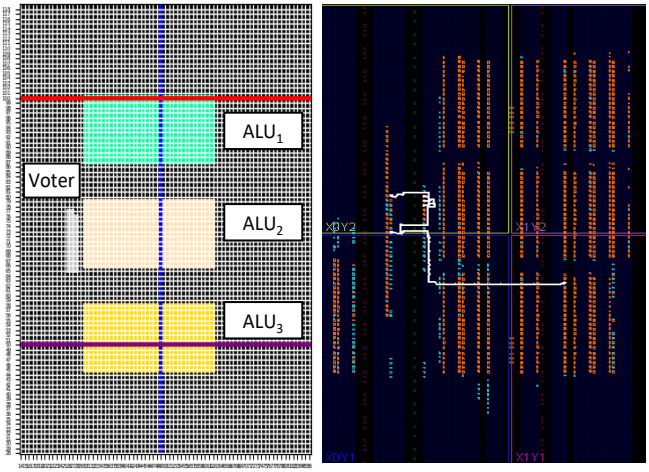
The RI5CY ALU has been hardened by replication and provided to the FPGA design flow as a testbench. In order to compare the implementations by fault injection campaigns, it is necessary to include in the design, in addition to the module to be tested, a processor core with the purpose to stimulate the DUT and collect its results. Therefore, a processing system is added to the design. The block design will continue unmodified through elaboration, synthesis, and implementation steps of the design flow as used by the Xilinx design flow.

5.2 Customized Layout Solutions

An accurate analysis of the placement proposed by Vivado revealed resource sharing between the TMR domains, i.e. logic cells belonging to different redundant modules were implemented on the same SLICEs and connected through the same Switch Boxes.

These shared resources are likely the most vulnerable points for a TMR design on FPGA because if a single SEU occurs in the memory cell that configures that resource, we will have a fault that affects two modules instead of one, potentially leading to an inefficiency in the voting system and to cross-domain failures.

As suggested in [16], benefits can be derived from applying an isolation policy based on the TMR domain. The isolation has been imposed through the developed placement tool that detects redundant modules from the design hierarchy and automatically organizes the resources assigned to each redundant core in such a way that between neighborhood areas, there are at least four rows of unused resources. The voter instead has been placed in front of the central replica, according to the user's constraint inserted through the tool command line. Moreover, the tool offers a preview of the customized solution, as shown in Fig. 5. a, so that users can find the optimal solution that reflects their intent, without having to go through Vivado's entire design flow.



(a) (b)

Figure 5: an example of (a) the isolated placement layout proposed by the developed placement tool (b) domain-based isolation at net level achieved through the developed router

In addition, for evaluation purposes only, we have created a placement layout that emphasizes the vulnerability induced by resource sharing, imposing that for each SLICE of the design there is the simultaneous presence of logic cells belonging to the three.

5.3 Customized Routing Solutions

The developed router was used to extend the concept of isolation at the net level, according to the TMR domain principle. As the voter is a critical point, it must be treated individually. The presence of the voter, to which the outputs produced by the three replicas converge, pushed the routing algorithm adopted by the commercial tool to arrange the routing path of an output produced by a replica in the area where another replica is defined, to satisfy the performance fitness. This means that if the replicas do not share the same logical resources, they share the interconnection resources, thus compromising the functionality of the TMR technique. To overcome this problem, as a constraint, we assign a horizontal routing path for the entire length of the rectangle of the parent redundant core and then a straight vertical movement up to the voter module. Fig 5.b represents as example of the outcome of the developed routing tool.

6 Experimental Analysis

Fault injection is a widely used technique to emulate radiation effects. In the world of programmable logic, injection is done by forcing a flip in one or more bits of the device configuration memory. Specifically, in this research work, the injection was done statically, that is, by injecting a fault into the bitstream before it is loaded into memory.

6.1 Experimental Setup

The hardware platform used to implement and test the design is the PYNQ-Z2 development board, containing the Zynq-7000 AP-SoC. The bitstream was corrupted by employing the PyXEL [24] framework, which is a Python tool for the emulation of faults effects affecting the FPGA configuration memory. This platform allows injecting faults by changing one or more bits in the bitstream. As the bitstream is organized into frames made of several bits and since not all the bits are used to develop the target design, PyXEL allows choosing in which region of the bitstream to inject the fault, which is useful to avoid wasting time and resources by injecting into regions of the bitstream not used by the current design. Each injection campaign consists of generating 10k corrupted bitstreams, each with a single bit-flip located at a random point in a specific region of the bitstream. The region was chosen such that the three placement solutions being compared had resources defined there.

6.2 Experimental Results

Fig. 6 represents the results of the injection campaigns conducted on the three layouts. The bare-metal testing routine executed by the ZYNQ7 arise an error every time there's a mismatch between the output provided by the DUT and the one produced by itself on the same input data. Thus, an error is a failure of the whole TMR system.

The low error rate is mostly due to the fact that the design uses a small number of device resources and therefore while limiting the injection bitstream area, a single bitflip in configuration memory does not have a substantial impact. Nevertheless, it is possible to appreciate differences in failure trends between the layout solutions. The isolated design shows an improvement in vulnerability to SEUs, compared to the original design. On the other hand, the results show that a design with a high sharing of resources and net congestion is much more sensitive, up to the point of having about three times as many occurrences of wrong output in at least one replica and twice as many TMR failures than the isolated one.

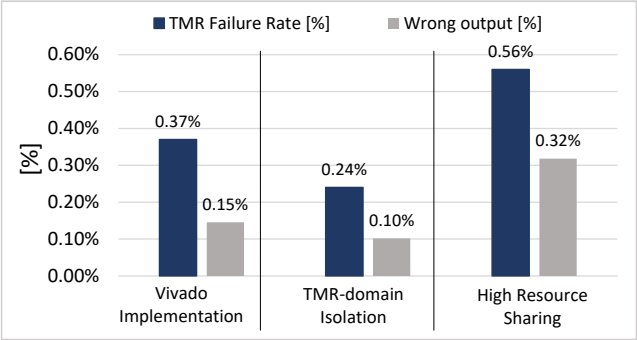


Figure 6: Detected TMR-failures and wrong results on 10,000 fault injections on original, isolated, and high resource sharing layout configurations.

Table I. Design Layouts Comparison

TMR Designs	Performance [MHz]	Power Consumption [W]	Resource Utilization [%]
Original	50	1.397	13.18
Domain isolated	20	1.395	15.25
High resource sharing	15	1.394	23.10

Table 1 shows the comparison among the three layouts. Due to the presence of unused resources among the replicas and the voter, the critical path delay increases in the isolated layouts leading to a degradation of performance, which becomes even worst in the case of high resource sharing layouts where the nets are heavily congested. The power consumption undergoes a slight variation between designs, depending on the dynamic power contribution that decreases as the clock frequency decreases. Finally, the number of resources used increases progressively between the three designs, since in both the isolation case and especially in the high sharing case, more interconnection resources and more buffers are needed. The use of the developed platforms on one hand accelerated the realization of the isolated layout as 64 nets were customized in less than five minutes, compared to the commercial tools approach that requires almost 64 times more. On the other hand, it allowed the realization of the high sharing layout, which is not feasible with the traditional placement approach, unless a user decides to manually place cell by cell, taking days/weeks.

7 Conclusions and future works

This paper presents a method to automatize the customizing process of the layout of designs mapped on FPGAs to obtain a more reliable solution. We presented a modulable placement and routing approach that enforces commercial tools to automatically achieve a robust implementation. The proposed solution has been applied to a Xilinx development board, and it has been used to evaluate the effect of placement on the reliability of a fault-tolerant design by taking the TMR version of an ALU extracted from a RISCV-core implementation as a case study. The results revealed that applying TMR domain-level isolation involving both logical cells and nets leads to a more robust TMR technique.

REFERENCES

[1] S. Azimi and L. Sterpone, "Digital Design Techniques for Dependable High-Performance Computing," IEEE International Test Conference (ITC), pp. 1-10, 2020, doi: 10.1109/ITC44778.2020.9325281.

[2] C. De Sio, et al., "FireNN: Neural Networks Reliability Evaluation on Hybrid Platforms," in IEEE Transactions on Emerging Topics in Computing, doi: 10.1109/TETC.2022.3152668.

[3] B. Du, et al., "On the Reliability of Convolutional Neural Network Implementation on SRAM-based FPGA," IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 1-6, 2019, doi: 10.1109/DFT.2019.8875362.

[4] L. Sterpone et al., "A Novel Error Rate Estimation Approach forUltraScale+ SRAM-based FPGAs," NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 120-126, 2018, doi: 10.1109/AHS.2018.8541474.

[5] S. Azimi, et al., "On the analysis of radiation-induced Single Event Transients on SRAM-based FPGAs", Microelectronics Reliability, Vol.88–90, pp. 936-940, 2018, doi: 10.1016/j.microrel.2018.07.135

[6] C. De Sio, et al., "Analyzing Radiation-Induced Transient Errors on SRAM-Based FPGAs by Propagation of Broadening Effect," in IEEE Access, vol. 7, pp. 140182-140189, 2019, doi: 10.1109/ACCESS.2019.2915136.

[7] 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, 2019, doi: 10.1109/TNS.2019.2915207.

[8] L. Sterpone, et al., "A 3-D Simulation-Based Approach to Analyze Heavy Ions-Induced SET on Digital Circuits," in IEEE Transactions on Nuclear Science, vol. 67, no. 9, pp. 2034-2041, 2020, doi: 10.1109/TNS.2020.3006997.

[9] C. De Sio, et al., "On the analysis of radiation-induced failures in the AXI interconnect module", Elsevier Microelectronics Reliability, vol 114, 2020, doi: 10.1016/j.microrel.2020.113733.

[10] De Sio, et al., "SEU Evaluation of Hardened-by-Replication Software in RISC-V Soft Processor," IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 1-6, 2021, doi: 10.1109/DFT52944.2021.9568342.

[11] S. Azimi, et al., "A Radiation-Hardened CMOS Full-Adder Based on Layout Selective Transistor Duplication," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 8, pp. 1596-1600, 2021, doi: 10.1109/TVLSI.2021.3086897.

[12] S. Azimi, et al, "A New Single Event Transient Hardened Floating Gate Configurable Logic Circuit," 18th IEEE International New Circuits and Systems Conference (NEWCAS), pp. 311-314, 2020, doi: 10.1109/NEWCAS49341.2020.9159844.

[13] Xilinx, "7 Series FPGAs Data Sheet: Overview", Xilinx Product Specification, September 8, 2020.

[14] S. Azimi, et al., "A new CAD tool for Single Event Transient Analysis and mitigation on Flash-based FPGAs", Integration, the VLSI journal, vol 67, pp. 73-81, 2019, doi: 10.1016/j.vlsi.2019.02.001.

[15] P. D. Schiavone, et al., "The RI5CY: User Manual Revision 4.0", 2019.

[16] A. Portaluri, et al., "A New Domains-based Isolation Design Flow for Reconfigurable SoCs," *IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS),* pp.1-7, 2021, doi: 10.1109/IOLTS52814.2021.9486687.

[17] P. Banerjee, et al., "Floorplanning for partially reconfigurable FPGAs", IEEE Transaction Computer-Aided Design of Integrated Circuits Systems, vol. 30, pp. 8-17, 2011.

[18] M. Rabozzi, et al., "Floorplanning Automation for Partial-Reconfigurable FPGAs via Feasible Placements Generation," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 151-164, 2017, doi: 10.1109/TVLSI.2016.2562361.

[19] L. Cheng and M. D. F. Wong, "Floorplan design for multimillion gate FPGAs", IEEE Transaction Computer-Aided Design of Integrated Circuits Systems, vol. 25, no. 12, pp. 2795-2805, 2006, doi: 10.1109/ICCAD.2004.1382589.

[20] Y. Feng and D. P. Mehta, "Heterogeneous floorplanning for FPGAs", International Conference on VLSI Design, pp. 257-262, 2006, doi: 10.1109/VLSID.2006.96.

[21] Xilinx, "Vivado Design Suite Tutorial: Design Analysis and Techniques", UG986 (v2018.2) June 6, 2018.

[22] Mo, A. Tabbara and R. Brayton, "A Force-Directed Maze Router," IEEE/ACM International Conference on Computer-Aided Design, pp-404-407, 2001, doi: 10.1109/ICCAD.2001.968658.

[23] L. McMurchie and C. Ebeling, PathFinder,"A negotiation-based Performance-Driven Router for FPGAs, " ACM FPGA Symposium., pp. 111-117, 1997, doi: 10.1109/FPGA.1995.242049.

[24] L. Bozzoli, et al., "PyXEL: An Integrated Environment for the Analysis of Fault Effects in SRAM-Based FPGA Routing," International Symposium on Rapid System Prototyping (RSP), pp. 70-75, 2018, doi: 10.1109/RSP.2018.8632000.