An integrated flow for the design of hardened circuits on SRAM-based FPGAs

Cristiana Bolchini, Antonio Miele, Chiara Sandionigi
Dip. Elettronica e Informazione
Politecnico di Milano
Milano - Italy
{bolchini,miele,sandionigi}@elet.polimi.it

Niccolò Battezzati, Luca Sterpone, Massimo Violante
Dip. di Automatica e Informatica
Politecnico di Torino
Torino - Italy
{battezzati,sterpone,violante}@polito.it

***Abstract*—This paper presents an enhanced design flow for the implementation of hardened systems on SRAM-based FPGAs, able to cope with the occurrence of Single Event Upsets (SEUs). The framework integrates three strategies independently designed to tackle the problem of SEUs; first a systematic methodology is used to harden the circuit exploiting an enhanced TMR-based technique, coupled with partial dynamic reconfiguration. Then, a robustness analysis is performed to identify possible TMR failures, eventually solved by a specific local re-design of the critical portions of the implementation. We present the overall flow and the benefits of the solution, experimentally evaluated on a realistic circuit.**

I. Introduction

As SRAM-based Field Programmable Gate Arrays (FPGAs) are gaining popularity in the development of safety- and mission-critical applications to be deployed in radioactive environments, several SEU mitigation techniques have been proposed in the past years, due to their susceptibility to Single Event Upsets (SEUs) [1]. Faults may occur in the device configuration memory, the bitstream, thus modifying the implemented functionality with a *persistent* effect, until the correct configuration is re-written. On the other hand, faults may occur in the circuit memory elements, causing a *transient* effect, which can be dealt with by means of fault masking approaches. To mitigate such effects, a lot of approaches have been devised in the past years [2], [3], [4], [5], [6], [7], [8], which combine reconfiguration-based techniques and redundancy-based techniques to achieve a reliable system.

While the first families of FPGAs were characterized by a homogeneous reconfiguration fabric, most of the first approaches in both categories of techniques tackled the SEU problem, by acting on the entire system. Bitstream recovery took into account the entire configuration memory [5], whereas redundancy-based techniques, such a Triple Modular Redundancy (TMR), were applied to the entire system, with a fine granularity (e.g., the approach by Xilinx's TMRTool, or X-TMR, [8]). However, as the latest and future families of SRAM-based FPGAs offer a finer grain control in the device reconfiguration, supporting partial dynamic reconfiguration, more refined approaches have been proposed, to reduce costs and improve error latency and power consumption [7], [9].

This work has been partially funded by MIUR-PRIN project #2008K4P7X9_002.

Common to all these hardening approaches is the fact that the final implementation onto the reconfigurable fabric might not achieve a complete fault tolerance to single SEUs, due to faults in the configuration memory that might cause more than a single failing replica in the circuit, thus invalidating the voting strategy. This limitation is due to the particular nature of the reconfiguration fabric, and to the standard synthesis process that does not manage any additional constraint. This issue has been addressed in the recent years, leading to the development of tools able to perform robustness analysis of the TMR-based hardened implementations, to identify such single elements causing a TMR failure [10]. Furthermore, also specific placement and routing strategies [11] have been developed to locally re-design those parts of the design where faults would cause the above mentioned problems.

However, all these approaches have been defined as stand-alone solutions, each one tackling only a part of the problem. We claim that an integrated flow, offering both i) an accurate design exploration in the application of the most convenient form of TMR-based hardening technique, and ii) the possibility to directly fix the identified local implementation points of TMR failure, is an important step forward in the direction of an up-to-date tool chain for the design of reliable systems on SRAM-based FPGAs. This paper presents the first results of this integrated flow, based on the seamless interaction of four different tools to design fault tolerant systems onto the selected platform. More precisely, the proposed integrated design flow joins the efforts of a hardening approach devised to identify better solutions than the ones proposed by applying the standard TMR technique, and strategies to guarantee the achievement of a complete fault tolerance. By exploiting the peculiarities of the constituting tools, the resulting framework can identify hardened implementations – based on the TMR technique coupled with partial dynamic reconfiguration – that constitute interesting solutions, especially with respect to fault recovery characteristics. Such solutions have been compared also to the solution achieved by using X-TMR and the results of an experimental campaign are here reported. At present, the flow is mainly based on the exploitation of the direct interaction and information sharing, although in the future a more tightly-coupled interaction is foreseen.

The rest of the paper is organized as follows. Section II presents the relevant work of the most recent years on the hardening of designs implemented on SRAM-based FPGAs. Section III and IV introduce the proposed architectural solution and the integrated design flow, which is the main contribution of our presentation. The subsequent section reports the experimental results of the integrated flow, applied to a case study, discussing its benefits also with respect to the solution achieved with the TMRTool approach. Section VI closes the paper, also highlighting future work.

II. Related work

A lot of effort has been devoted to the definition of techniques to mitigate the effects of SEUs in SRAM-based FPGAs. These techniques can be organized in two categories: reconfiguration- and redundancy-based techniques. The former are used to remove faults from the FPGA-based system, while the latter to mask fault effects. It is important to remark here that these techniques are complementary, and therefore a reliable system is obtained by combining both of them.

A. Reconfiguration-based techniques

SEUs affecting the configuration memory of SRAM-based FPGAs can be considered as persistent faults. Indeed, whenever an SEU is latched in the configuration memory it persists until a fresh image of the configuration memory is reloaded. The simplest approach used to remove SEUs in the FPGAs' configuration memory is known as *Scrubbing*, which consists in periodically rewriting the configuration memory [5]. It introduces a limited overhead; the circuit needed to control the bitstream loading process, and the memory for storing an error-free bitstream. The system also needs a mechanism to control the period, usually referred to as *scrub rate*. This rate is determined on the basis of the expected SEU rate, i.e., on the basis of a figure predicting how often SEUs may appear in the FPGA configuration memory. An improvement to the Scrubbing method exploits the partial reconfiguration capability of the latest generation of SRAM-based FPGAs, which allows reconfiguring only a user-selected portion of the configuration memory (known as frame) [2]. This technique uses a *Readback* process to read one frame at a time. Either the frame is compared with an error-free one, stored in a *safe* off-chip memory, or a CRC check code is used to verify its correctness. When an SEU is detected, the faulty frame is rewritten, usually allowing the rest of the circuit to operate normally. Readback operation, although not impacting the normal system behavior, entails power consumption, and therefore its occurrence rate should be minimized. However, a low Readback occurrence rate corresponds to a long SEU latency. As a result, in harsh environments, where the SEU rate is high, a long SEU latency increases the probability of SEU accumulation, thus increasing the probability that accumulated SEUs corrupt the fault masking techniques the system embeds (that are normally devised under the single-fault assumption).

B. Redundancy-based techniques

The techniques belonging to this category exploit additional hardware components, or computation time, for detecting the presence of SEUs modifying the expected circuit operations, and/or masking SEUs propagation to the circuit's outputs.

Fault detection can be achieved by duplicating the circuit the FPGA implements: the outputs the two replicas produce are continuously compared, and an alarm signal is raised as soon as a mismatch is found [5]. This solution is fairly simple, and cost-effective: however, it is not able to mask SEU's effects.

When fault masking is mandatory, designers may resort to the TMR approach, using three copies of the same circuit and building a majority voter on the outputs of the replicated circuits. When the configuration memory of FPGAs is considered, the TMR implementation should be reconsidered, since a modification in the configuration memory may affect every FPGA's resource: routing resources implementing interconnections, combinational resources, sequential resources, I/O logic. This means that three copies of the whole circuit, including I/O logic, have to be implemented to harden it against SEUs [5]. For these, as well as for the other special FPGA's resources (e.g., block RAM, Shift-registers), there are particular recommendations to guarantee an accurate TMR architecture (more details can be found in [2], [5]).

Other methodologies to implement redundant architectures on SRAM-based FPGAs are available, too. One of these techniques is oriented to perform all mitigations using the description language to provide a functional TMR methodology [12]. According to this methodology, interconnections and registers are triplicated and internal voters are used before and after each register in the design. The advantage of this methodology is that it can be applied to any type of FPGAs.

Another approach is based on the concept that a circuit can be hardened against SEUs by applying TMR selectively (STMR) [3]. This approach extends the basic TMR technique by identifying SEU-sensitive gates in a given circuit and then by introducing TMR selectively on these gates, only. Although this approach optimizes TMR by replicating only the most sensitive portions of a circuit (thus saving area), it needs a high number of majority voters, since one voter is needed for each SEU-sensitive circuit portion.

To reduce both the pin count and the number of voters used to implement the TMR approach, Lima et al. proposed a technique based on time and hardware redundancy to harden combinational logic [4], [13]. This technique combines duplication with comparison (DWC) with a concurrent error detection (CED) machine based on time redundancy that works as a self-checking block. DWC detects faults in the system, and CED detects which blocks are fault free.

The techniques discussed so far rely on the designer's experience for system partitioning and TMR (or DWC) application, with the partial support of automated tools like TMRTool for design replication and voter insertion. The level of granularity to be used when replicating the design (component level, module level, or system level) is a choice let to the designer and is not supported whatsoever [14], [6]. When the designer must proceed manually, it is difficult to evaluate many alternatives, due to the effort in techniques application and synthesis. At present, there is only a methodology ([7]) supporting a systematic analysis of the possible alternative solutions deriving from the application of the hardening techniques at different levels of granularity. The approach, called R4R, exploits redundancy-based techniques to mask the occurrence of a fault *and* to localize the portion of the FPGA hit by the fault, to trigger an *on-demand* partial re-configuration. The methodology offers an automated design space exploration that compares different solutions with respect to selected cost/benefit parameters.

However, independently of the adopted technique, place and route is a fundamental step of the design implementation, and should be performed by enforcing dependability-oriented placement and routing rules that guarantee to obtain a circuit which is free from single-point of failures originated by SEUs in the FPGA configuration memory [11]. Therefore, to complete any of the above mentioned hardening approaches, which work at a functional level, it is necessary to use other structural strategies to enforce an independence between the replicas to guarantee the effectiveness of the voting mechanism.

The proposed flow aims at integrating tools for hardening the system both at functional and structural level, exploiting both redundancy- and reconfiguration-based techniques, to achieve the desired fault tolerance against SEUs. The architecture of the resulting hardened system is described in the next section, to provide a general idea, whereas the integrated flow is introduce in Section IV.

III. The system architecture

The architecture of the fault tolerant system (Fig. 1) consists of three modules. The core of the system is the Payload FPGA, an SRAM-based device implementing the user application. Each time the system is powered on, the SRAM-based FPGA is programmed using the bitstream stored in the Configuration memory backup (a Flash memory device).

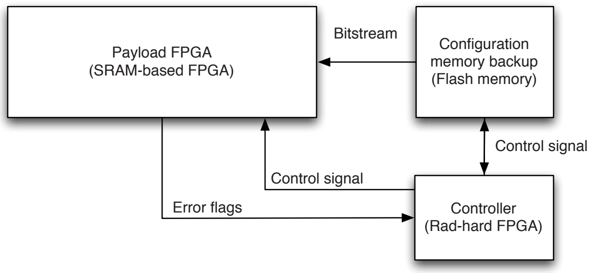

Fig. 1. The system architecture.

The Controller module, implemented using a rad-hard FPGA, supervises the operation of the whole system. It implements two functionalities: the generation of the control signals for reading the content of the Configuration memory backup, and for managing the configuration of the Payload FPGA, and a watchdog timer functionality. It periodically challenges the Payload FPGA (for example reading its status register) to verify whether it is correctly operating, or whether it is stalled due to the occurrence of a Single Event Functional Interrupt (SEFI). In case an SEFI is detected, the controller resets the Payload FPGA, and performs a complete reconfiguration. If the operation does not succeed, the controller performs a power cycle, and reprogram the Payload FPGA. In our architecture, configuration memory scrubbing is performed *on-demand*. This means that the configuration memory of the Payload FPGA is refreshed, through partial reconfiguration, only when Error flags indicate that an error is detected. Error flags are generated by ad-hoc designed majority voters employed in our architecture. Beside the typical functions of a voter (like that of voters used by X-TMR, the solution produced with the TMRTool), the voters we are using provide additional outputs to identify the TMR domain where the SEU is located. With this information, the Controller is capable of activating configuration memory scrubbing only when actually needed, and only the faulty portion of the configuration memory is rewritten, thus removing error detection latency.

IV. The integrated design flow

To design effective hardened TMR-based implementations for SRAM-based FPGAs, offering the mitigation efficacy of the TMR, coupled with the benefits of partial dynamic reconfiguration, to recover from the occurrence of the SEUs, an integrated flow is proposed. More precisely, the proposed flow is the result of the seamless integration of the data produced by the different tools, whose intrinsic flexibility allows for an easy adaptation and exploitation of the provided information.

A. The proposed flow

The starting point of the proposed flow is the initial VHDL specification of the nominal circuit to be hardened by means of the TMR technique. The ultimate goal is to identify a few interesting solutions among which the designer can then select the one s/he likes best. The output of the design flow is the few bitstreams corresponding to such solutions, ready to be implemented on the FPGA.

The flow, depicted in Fig. 2, can be seen as constituted by three main phases, each one carried out by the developed tools: i) *system hardening* by means of the TMR technique, producing a set of optimal solutions, ii) preliminary solutions *robustness analysis* to identify possible TMR failures due to the design implementation on the reconfigurable fabric, and, for the most promising solutions, iii) *local re-design* of the critical areas to achieve a complete fault tolerance.

B. System hardening

Reconfiguration for Reliability (*R4R*) is a methodology and a companion tool for the design of reliable systems on SRAM-based FPGAs featuring an on-demand partial scrubbing of the faulty part of the system implemented on the architecture proposed in Fig. 1. The methodology was preliminary presented in [7], while the complete framework is described in [15].

The R4R tool takes as input the graph-based representation of the circuit obtained by the VHDL description and annotated with its costs and requirements. Additional information comes from the repositories for the set of available hardening techniques, for the FPGAs' resources models, and for the set of

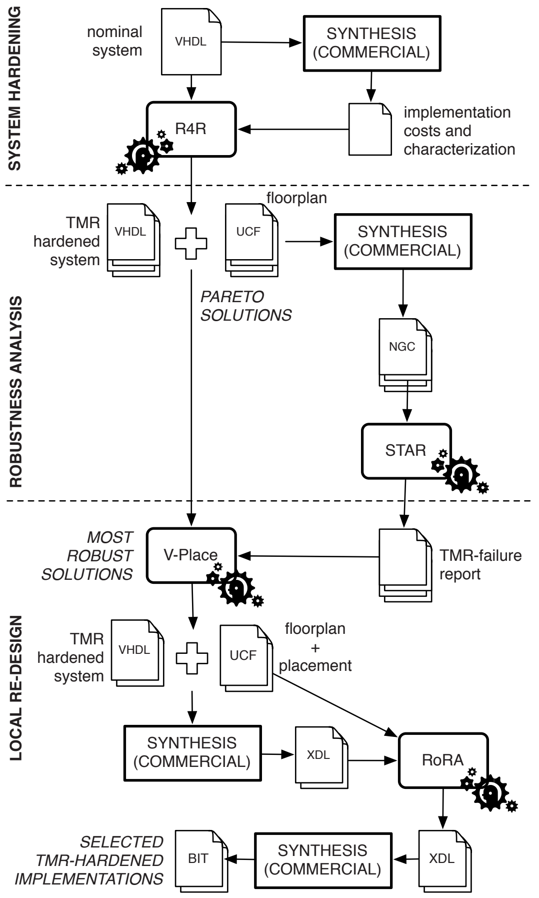

Fig. 2. The integrated design flow.

metrics driving the evaluation of the hardened solutions. The R4R flow consists of two phases: i) a *techniques' application* and ii) a *floorplanning* one. The former performs a fast automated design space exploration powered by a multi-objective genetic algorithm that analyzes the different solutions deriving from the application of the available hardening techniques. The goal is to optimize the solution according to a set of figures of merit estimating the quality of the identified circuit and the impact of the techniques before its implementation.

The explorative approach of the methodology is motivated by the fact that there are several degrees of freedom that generate a huge space of different hardened implementations. The tool supports the use of heterogeneous reliability requirements for the different components (e.g., fault detection or tolerance) and the use of various techniques. However, here, a homogeneous fault tolerance requirement is set for the entire circuit and only the TMR technique is adopted. Yet, even in this simplified scenario, the possible solutions are numerous, since each hardening technique can be applied at different levels of granularity, on the entire system, on each component, or on groups of components. By partitioning the system into groups of components, it is possible to localize the portion of the FPGA affected by the fault, since the voting circuits not only perform a majority vote, but also identify the corrupted replica (or voter), by means of a special ad-hoc design. Furthermore, the replicas and the voter itself can be mapped onto Independently Reconfigurable/Recoverable Areas (IR Areas), portions of the FPGA that can be partially scrubbed to recover, when the fault is precisely located. The replicas and the voter can be hosted on different IR Areas, or on the same one, or on two or three IR Areas. As a result, the grouping and mapping generate a relevant number of possible solutions, and the design exploration phase identifies the most promising ones. In this way, R4R actually couples a variation of the TMR hardening technique with partial dynamic reconfiguration.

The second phase, called floorplanning, analyzes for each identified solution the suitable placements of the obtained IR Areas on the reconfigurable fabric to guarantee that the required resources (number and type) and the reconfiguration constraints are satisfied for each IR Area.

The output of the this first step is a set of Pareto-optimal solutions, with respect to the metrics the designer has deemed as important. In the present scenario, the proposed flow aims at offering a more interesting alternative to the X-TMR solution generated by the TMRTool, pursuing solutions that, with a comparable cost in terms of area overhead, can recover from faults in a reduced amount of time. As a consequence, the adopted metrics for driving the exploration and weighting the granularity of the application of the TMR technique take into account: 1) the average number of frames to be reconfigured to recover each IR area, which indicates the average scrubbing time, 2) the standard deviation of the number of frames to be reconfigured, which is used for balancing the IR area dimension, and 3) the the number of voted lines that provides an early estimation of the possible TMR failure points. The set of solutions usually accounts for up to ten different designs, offering equivalent trade-offs with respect to the adopted cost functions. The output of this first step is, for each one of these optimal solutions, the VHDL description of the hardened system, with all the replicas and the necessary voters, as well as the floorplan information on where to position the various replicas onto the reconfigurable fabric, in a User Constraint File (UCF). At this point, there is *only* an area constraint derived from the floorplanning phase, to guarantee the possibility to independently reconfigure a portion of the device when a fault occurs and is localized by the adopted enhanced TMR.

At this point, commercial tools (such as Xilinx's ISE Tool Suite) can be used synthesize the obtained hardened solutions, using the floorplan constraints.

C. Hardened implementations robustness analysis

The second step of the integrated flow consists of the analysis of these implemented solutions, to verify how each implementation meets the functional requirements of a robust TMR technique application, that is if there are single faults that cause errors in more than a single replica, thus invalidating the voting strategy. STAR [16] performs a low level analysis of the TMR-hardened circuit netlist, by building an internal representation which derives the various replicas' domains and the voters. By emulating the injection of single bit flips in each one of the used bits of the bitstream, the algorithm identifies those faults that have an effect in more than a single replica domain, thus invalidating the voting strategy. These faults are reported as “TMR failures” or “single point of failures”.

D. Local re-design

The last phase of the flow performs a local re-design of the most promising hardened solutions, by solving the TMR failure problems, by re-distributing the logic onto the reconfigurable fabric, within the area constraints set by R4R.

This phase is based on two tools elaborating the circuit’s netlist at the layout level; Versatile Placement algorithm (V-Place) [17] and Reliability-oriented Routing Algorithm (RoRA) [11]. The V-Place algorithm starts by reading the description of the circuit, which consists of logic functions and the connections between them. The goal of V-Place is to find a valid placement for each configuration logic block while minimizing single point of failures of the circuit. The modification of the placement positions is made accordingly to the area constraints defined by the R4R approach. The V-Place algorithm reads the XDL circuit description which includes CLBs, I/O pads and their interconnections PIPs, and generates a new UCF file to bind the placement position of each CLBs within the FPGA’s array, maintaining each CLB position in the area defined by R4R.

The algorithm is based on a min-cut optimization technique mixed with a quadratic placement, where the min-cut technique is applied to the general CLBs and the quadratic placement is executed for each voter’s CLB.

The overall V-Place placement strategy is a three-phases approach. After an execution of the preliminary phase which initializes the execution environment, the V-Place algorithm works on two parts: the former aims at reducing the overall routing congestion of the connections between CLBs of the selected area, while the latter is a local optimization to improve the specific SEU’s critical position related to the voter resources. The circuit is loaded and stored into an undirected graph, where vertices represent the logic resources such as LUT, MUXs of FFs and the undirected edges represent the interconnections between the logic resources. The vertices of the graph are identified by a location which corresponds to the physical location of the logic element into the FPGA’s array. Each logic resource vertex is associated with the type of resources and with a domain identified, that identifies the number of the TMR domain the logic element belongs, and is moved according to the placement rules defined in [17].

RoRA re-routes all the nets identified as critical by STAR, and creates a new routing path using different PIPs in order to remove possible multiple-open or short events that may corrupt the TMR-based hardening structure. RoRA creates a final XDL file, where a robust design according to the single fault assumption is stored.

The output of the entire integrated flow, is a few hardened implementations of the nominal system, by means of a TMR-based SEU mitigation approach, which couples the replication and voting strategy applied to groups of components with partial dynamic reconfiguration, to quickly recover from soft errors. Characteristic of the final solutions is the possibility to trigger *on-demand* the reconfiguration of the portion of the device affected by the detected SEU.

V. Experimental results

The proposed integrated design flow has been used to carry out some experimental sessions on a realistic circuit, to evaluate the benefits of the joint analyses. The circuit is an H.264 standard that provides a way to (de)compress video images. The device selected for the circuit implementations is a Xilinx FPGA xc4vsx35 [18].

The first part of the design flow, performed by the R4R framework, exploits components grouping to reduce the number of voters, the area and the average reconfiguration time. The design space exploration has been driven by multi-objective functions taking into account, for this experimental session, the average reconfiguration time (average number of frames per IR Area) and the frames standard deviation. Also a second execution has been performed, using the number of voted lines, as the second metric. The Pareto solutions (P1-P7) are reported in Tab. I, together with the trivial solution corresponding to TMR at system level on a single area (S1) and X-TMR from TMRTool. For each solution, the following values are reported: the total used area (number of slices), the average reconfiguration time (number of frames) and the resource optimization index (ro-index, the ratio between the number of required resources and the number of available resources). Finally, the last column reports the number of IR Areas for each solution; in particular, for solution P4 the identified floorplan is shown in Fig. 3. Results show that, on average, it is possible to reduce the reconfiguration time requested when a fault is detected, of at lease an 87%; both S1 and X-TMR require the reconfiguration of the entire device.

TABLE I
R4R Pareto solutions’ characteristics.

Solution	# Slices	Avg. # Frames	ro-index	# IR Areas
P1	13881	1012	0.341852	7
P2	11661	1426	0.387758	5
P3	13957	1013	0.162202	7
P4	13815	1070	0.383484	7
P5	11166	1746	0.377309	4
P6	13672	1469	0.385801	5
P7	13496	1900	0.347295	4
S1	13396	7600	0.353776	1
XTMR	14354	7600	0.369368	1

All solutions have been analyzed using the STAR algorithm, to verify the robustness of the implementations and their sensitivity to TMR failures. The results of the analysis are reported in Tab. II, where the second column shows the number of SEUs causing a multiple failure such that the TMR technique is not effective, and the third one shows the size of the final circuit, in terms of the number of used bits in the

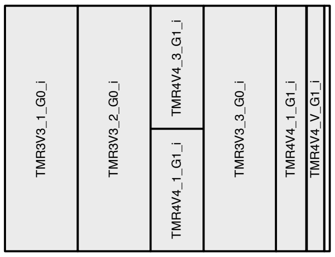

Fig. 3. R4R floorplan for solution P4.

bitstream. Columns 4 and 5 show the same information, after running V-Place and RoRA, respectively.

TABLE II
STAR ANALYSIS, AND V-PLACE & RORA FINAL ROBUST SOLUTIONS.

Solution	TMR failures	Used bits	TMR failures	Used bits
P1	650	597688	231	601225
P2	1175	501521	252	506783
P3	1155	612833	243	614918
P4	538	585386	103	587459
P5	374	466115	83	482193
P6	377	603644	84	610385
P7	46	528551	0	531782
S1	469	544137	64	561005
X-TMR	7893	709652	12	764788

It is interesting to note that, although the solutions are similar with respect to the used metrics, that find a convenient trade-off between average reconfiguration time, balanced areas and voted lines, their sensitivity to TMR failures is quite different. In general, the solutions identified by R4R show a lower number of sensitive bits with respect to X-TMR solution, but the X-TMR solution is also the one that most benefits from the adoption of V-Place and RoRa. The benefits of combining R4R with V-Place and RoRa become evident with P7, where the number of sensitive bits is reduced to 0, while in the X-TMR case, even after the application of V-Place and RoRa 12 bits remain sensitive. Future work is devoted to the investigation of the nature of such failures and in the mechanisms adopted by the local re-design tools to improve their effectiveness also for the solutions identified by R4R.

VI. CONCLUSIONS AND FUTURE WORK

The paper proposes an integrated design flow for the implementation of TMR-based hardened circuits on SRAM-based FPGA platforms. The flow is the result of a seamless integration of independently developed tools, to define a framework able to offer interesting solutions with respect to the commercially available TMRTool. More precisely, these solutions are characterized by similar area overheads but a significantly reduced recovery time: the portion of the device hit by a SEU is specifically reconfigured when the fault is detected and localized. The result is possible thanks to the integration of tools able to functionally harden designs by applying flexible and novel TMR-based hardening technique coupled with partial dynamic reconfiguration (R4R) and tools for the analysis of the robustness of the implemented designs on the FPGA (STAR). A final phase of local re-design is then carried out to solve the limited number of TMR-related failure points of the implementation (carried out by V-Place and RoRA) thus achieving complete fault tolerance. The experimental results show interesting figures in terms of both the achieved solutions and the reduced effort of the last phase with respect to X-TMR solutions.

Future work is devoted to the investigation of a more tightly coupled integration of the hardening and analysis tools, to anticipate the robustness analysis – possibly – to the design exploration phase, for a more efficient solution identification.

REFERENCES

[1] E. Normand. Single Event Upset at Ground Level. *IEEE Trans. on Nuclear Science*, 43(6/1):2742–2750, 1996.
[2] C. Carmichael, M. Caffrey, and A. Salazar. *Correcting Single-Event Upsets Through Virtex Partial Configuration*. Xilinx App. Notes 216, 2000.
[3] P. K. Samudrala, J. Ramos, and S. Katkoori. Selective triple modular redundancy (stmr) based single-event upset (seu) tolerant synthesis for fpgas. *IEEE Trans. on Nuclear Science*, 51(5), October 2004.
[4] F. Lima Kastensmidt, G. Neuberger, R. Hentschke, L. Carro, and R. Reis. Designing Fault-Tolerant Techniques for SRAM-Based FPGAs. *IEEE Design and Test of Computers*, 21(6):552–562, Nov./Dec. 2004.
[5] C. Carmichael. *Triple Module Redundancy Design Techniques for Virtex FPGAs*. Xilinx Application Notes 197, 2006.
[6] C. Pilotto, J. R. Azambuja, and F. Lima Kastensmidt. Synchronizing triple modular redundant designs in dynamic partial reconfiguration applications. In *Proc. Symp. Integrated Circuits and System Design*, pages 199–204, 2008.
[7] C. Bolchini and A. Miele. Design Space Exploration for the Design of Reliable SRAM-based FPGA Systems. In *Proc. 23rd IEEE Defect and Fault Tolerance in VLSI Systems Symposium, DFT*, pages 332–340. IEEE Computer Society, 2008.
[8] Xilinx Inc. *Xilinx TMRTool*, 2006.
[9] C. Bolchini, D. Quarta, and M. Santambrogio. SEU Mitigation for SRAM-Based FPGAs through Dynamic Partial Reconfiguration. In *Proc. ACM/IEEE Great Lake Symposium on VLSI*, pages 55–60, 2007.
[10] L. Sterpone and M. Violante. A New Analytical Approach to Estimate the Effects of SEUs in TMR Architectures Implemented Through SRAM-based FPGAs. *IEEE Trans. on Nuclear Science*, 52(6):2217–2223, 2005.
[11] L. Sterpone and M. Violante. A New Reliability-Oriented Place and Route Algorithm for SRAM-Based FPGAs. *IEEE Trans. on Computers*, 55(6):732–744, 2006.
[12] S. Habinc. *Functional Triple Modular Redundancy (FTMR) VHDL Design Methodology for Redundancy in Combinational and Sequential logic*. Gaisler Research.
[13] F. Lima Kastensmidt, L. Carro, and R. Reis. Designing fault tolerant systems into SRAM-based FPGAs. In *Proc. of the 40th Conf. on Design Automation*, pages 650–655, 2003.
[14] F. Lima Kastensmidt, L. Sterpone, L. Carro, and M. Sonza Reorda. On the Optimal Design of Triple Modular Redundancy Logic for SRAM-based FPGAs. In *Proc. Conf. Design, Automation and Test in Europe*, pages 1290–1295, 2005.
[15] C. Bolchini, A. Miele, and C. Sandionigi. A design flow for implementing reliability-aware systems on SRAM-based FPGAs. Technical Report 2009.26, DEI, Politecnico di Milano, 2009.
[16] L. Sterpone, M. Violante, R. H. Sorensen, D. Merodio, F. Sturesson, R. Weigand, and S. Mattsson. Experimental Validation of a Tool for Predicting the Effects of Soft Errors in SRAM-Based FPGAs. *IEEE Trans. on Nuclear Science*, 54(6):2576–2583, Dec. 2007.
[17] L. Sterpone. Timing driven placement for fault tolerant circuits implemented on SRAM-based FPGAs. In *ACM Int. Conf. Applied Reconfigurable Computing*, pages 85–96, 2009.
[18] Xilinx Inc. *Virtex-4 FPGA Configuration User Guide*, 2009.