An Automated Approach to Estimating Hardness Assurance Issues in Triple-Modular Redundancy Circuits in Xilinx FPGAs

Heather Quinn, Paul Graham, and Brian Pratt

Abstract— **The Xilinx Virtex family of static random access memory (SRAM) based field programmable gate array (FPGA) devices have made inroads into space-based computational platforms over the past decade. These devices are well-suited for digital signal processing (DSP) algorithms that are often used on orbit, providing the speedup of custom hardware without the cost of fabricating an application-specific integrated circuit (ASIC). SRAM FPGAs store the circuit in radiation-tolerant SRAM and SEUs can affect both the circuit functionality and the circuit state. Triple-modular redundancy (TMR) can be used to mask SEUs so that malfunctioning circuitry will not affect the output data. Unfortunately, applying TMR to a user circuit is difficult and unprotected cross-section is possible due to problems with the circuit design, device constraints, or the implementation of the user circuit on the FPGA. Given the complexity of these designs, estimating hardness assurance issues is not simple. This paper will present a tool, called the Scalable Tool for the Analysis of Reliable Circuits (STARC), that can automatically estimate unprotected cross-section and other hardness assurance issues for TMR-protected circuits.**

I. Introduction

Static random-access memory (SRAM) based field-programmable gate arrays (FPGAs) have become increasingly more common in space-based computing. Single-event upsets (SEUs) are a problem for these devices, as both the user's circuit implementation and the data are stored in SRAM. Therefore, even a single bit flip could cause the user's circuit to output incorrect data. To this end, most FPGA-based systems attempt to mask SEUs by protecting the user's circuit with triple-modular redundancy (TMR) [1]–[3]. The current suggestion for space-based FPGA designs is to triplicate all logic (modules and voters) and all signals (inputs, outputs, clock, and reset). We have shown that when these guidelines are followed it is possible to completely remove all unprotected cross-section [4] and the design will be susceptible only to multiple-bit upsets (MBUs), multiple independent upsets, and single-event functional interrupts. As the occurrence of single-bit SEUs dominate events on these devices, we believe that most designs that use these TMR criterion should be adequately protected on orbit.

Unfortunately, applying TMR techniques to user's circuits can be error-prone, leading to unprotected cross-section in the resulting circuits. Designers are not necessarily at fault in these scenarios. In particular, a number of problems can be tied to the design flow tools, which can only be circumvented entirely by avoiding many of the design automation tools — a choice most designers will not make. In other cases, designers are forced to apply TMR only partially to a design to meet device or resource constraints and the design will have remaining unprotected cross-section.

In all of these scenarios, hardness assurance issues with TMR-protected circuits can be very difficult to ascertain, especially in complex systems. In the past, we have used fault injection [3] to estimate hardness assurance issues that might exist in FPGA designs. Unfortunately, designers cannot always perform fault injection effectively on their designs due to flight system limitations or the limitations of hardware prototypes amenable to fault injection. In these cases, a non-hardware method for estimating the unprotected cross-section and for finding design flaws is necessary.

In this paper, we present a modeling tool, called the Scalable Tool for Analysis of Reliable Circuits (STARC)[1], that helps designers identify hardness as-

Document release number: LA-UR-08-04475. This work was funded by the Department of Energy through the Deployable Adaptive Processing Systems and Sensor-Oriented Processing and Networking projects, NASA through the Reconfigurable Hardware in Orbit project under AIST contract #NAG5-13516, and the Air Force Research Laboratory under the FPGA Mission Assurance Center.

H. Quinn and P. Graham, are with ISR-3 Space Data Systems, Los Alamos National Laboratory, Los Alamos, NM, 87545 USA (e-mail: hquinn@lanl.gov)

B. Pratt is with Brigham Young University, Provo, UT, 84602 USA

[1] Despite being similarly named, the STAR tool from Politecnico di Torino works at a much different level of abstraction than STARC [5]. While providing different information to the designer, these tools can be used in concert.

surance issues in their TMR-protected designs. With this tool, designers are able to assess whether TMR has been applied properly and whether unprotected cross-section remains. The tool can also quantify any unprotected cross-section, and determine architectural implementation issues that could affect the reliability of a TMR-protected design. STARC was designed to work within the existing design flow tools so that the time commitment needed to assess reliability issues would be minimal. In this way, we believe designers can start to address reliability early in the design phase, where fixing design flaws is inexpensive.

The paper will be structured as follows. Section II will present the related work for reliability analysis tools. Section III catalogs a number of potential reliability issues for TMR-protected circuit designs as they are implemented on Xilinx FPGAs. Section IV provides an overview of the STARC tool and how it can be used to help designers iterate over the tradespace of designs to minimize the unprotected cross-section in the design. The paper concludes with a case study that looks at the reliability tradespace for two image processing algorithms in Section V.

II. Related Work

Traditionally, circuit reliability has been determined using purely analytical approaches [6], [7] or techniques that model Boolean networks as probabilistic systems [8]–[11]. These modeling techniques represent circuits as probabilistic transfer matrices, stochastic Petri-nets, Markov chains or Bayesian networks. The combinatorics-based analytical approaches have been found to be error-prone and computationally complex for the analysis of large designs. Similarly, a number of limitations have been identified for many modeling-based approaches. First of all, model creation and input data sets greatly increase the time commitment of using these tools. Transforming circuits into intermediate probabilistic system models is an additional, computationally complex task. Within an analytical tool a state space is generated from the input model and input data vector set. The state space encodes all of the possible failure states in the circuit and grows exponentially with circuit size. The exception to these problems is the SETRA tool [12] that directly addresses the state space issues as well as automated model generation. Attempts at reducing computational complexity through circuit partitioning and hierarchical modeling of large circuits requires additional modeling effort. These limitations lead to the STARC tool, which uses EDIF circuit representations, no input data vectors, and simpler combinatorial reasoning to decrease the time commitment for the designer and reduce computational complexity in the tool.

Besides these differences between STARC and traditional reliability analysis tools, the tool methodologies differ greatly. There are two distinct methods [13] that can be used to analyze the reliability of circuits: generalized or instance-based. The *generalized* approach entails the combinatorial modeling of circuits without considering specific failure distributions of the inputs, gates and interconnects. A circuit's output's probability distribution is computed through combinatorics under the assumption that each gate can fail independently. Thus, the reliability is evaluated in stages using conditional probabilities. Generalized techniques to compute the reliability of large circuits require complex combinatorial reasonings. Reusing sub-circuit analysis to reduce the combinatorial complexity in the analysis of a larger circuit is difficult. Since specific input probability distributions are not considered during analysis, the generalized approach determines either the circuit's lower or upper bound on reliability.

Several *instance-based* methodologies have been proposed recently [10], [11], [14]. Instance-based reliability circuit analysis uses probability distributions on the primary inputs as well as gate and interconnect failure probabilities to develop an instance of the circuit. Each instance is then transformed into probabilistic circuit models. This method computes the exact reliability of the circuit for the input distribution. The main drawback of these tools is that several instances of the circuit needs to be analyzed to predict performance trends, which can be computationally expensive. Therefore, the input vector set needs to be limited to bound the computational cost, yet, provide enough intuition on the circuit's reliability.

The STARC methodology is a hybrid of the two approaches. STARC, as with other generalized approaches, is independent of specific input vectors and their probability distributions, yet uses specific gate distribution instances. Hence, this approach avoids the complex combinatorial reasonings that cause bottlenecks in generalized approaches and also bounds the computational complexity that affects instance-based methods. STARC computes a lower bound on reliability. When we have compared STARC with a purely instance-based approach based on PRISM [11], [15], the results of our comparison of STARC and PRISM were favorable. We tested four different designs with two probability-of-failure models based on estimated yield defects on a Dell Linux machine with 4 GB of RAM and dual 3.4 GHz Xeon microprocessors. We then compared the calculated reliability values and execution times. The ratio of the two calculated reliability values indicated that STARC was within three to seven digits of significance to PRISM. STARC also executes faster that PRISM, and for several designs was more than nine times faster.

It should be noted that a reliability analysis tool, called SEUper_fast [16], was designed by Boeing in the 1990s uses many of the same reasonings as probability transfer matrix tools. This tool approached the problem far more generally than STARC and was hindered by solving a much more complex reliability equation than STARC uses. While currently not as generally applicable as SEUper_fast, we believe we will be able to generalize this technique to different problems without having to employ the more complex reliability analysis technique.

III. Hardness Assurance Issues with TMR-Protected Designs

The potential reliability issues for TMR-protected designs for these devices are three-fold: problems with the circuit design, design constraints, and architectural influences on the circuit design. While the device is inherently radiation-tolerant and, therefore, SEU-sensitive while on-orbit, using TMR to protect the circuit should mask the effects of many SEUs as long as there is only one error in the system at a time. Even still, there are a number of ways in which either the design could be flawed or the design implementation toolset could render the final implementation of the design flawed. Furthermore, there might be design constraints placed on the circuit — such as not enough input/output pins for full triplication — that affect the reliability of the design. These issues are presented below.

A. Circuit Design Problems

The first issue is the design of the TMR-protected circuit. Many FPGA circuit designers use a hardware description language (HDL), such as VHDL or Verilog, to describe the FPGA circuit. The circuit description is then optimized for area and translated to an industry standard circuit representation, called Electronic Design Interchange Format (EDIF), using circuit synthesis tools, such as Synplify or Synopsys. Even the most careful descriptions of TMR-protected circuits are often undermined by the synthesis tools. As FPGA synthesis and implementation tools are designed to remove redundant logic to optimize the circuit for area and speed, these tools usually recognize and remove the functional redundancy intended to improve reliability. More subtly, though, sometimes the redundant modules remain, but are no longer functionally equivalent or independent. In this case, part of the redundant logic is reduced to a single implementation in one module that is shared by all three modules. This problem is shown in Figure 1. In this situation, the inverter that is used for the least significant bit in the counter has been removed from all three modules and the inverted data is shared by all three counter modules. While the circuit is still functionally equivalent to a correctly TMR-protected design, untriplicated logic now exists in the circuit. In large circuit designs, detecting this issue is difficult.

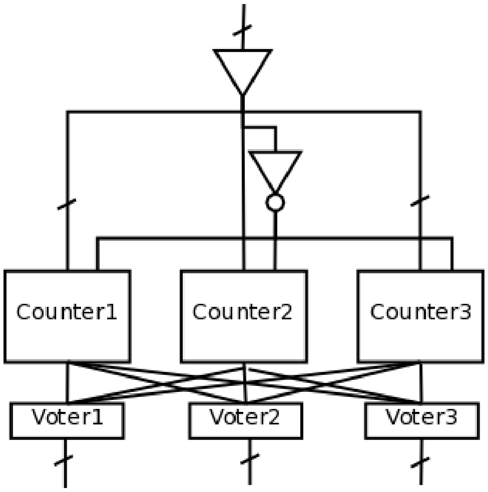

Fig. 1. An Example of a TMR-protected Counter Design with a Number of Design Flaws

Figure 1 also highlights a common problem in TMR-protected circuits with feedback loops. Feedback loops in TMR-protected systems are also sensitive to *persistent errors* [17], and need to use triplication and voters to break the feedback loops. The counter in Figure 1 shows a feedback loop that has not been cut properly and the counters will not be able to autonomously resynchronize after the SEU is removed. In this scenario, while the first SEU in the feedback loop will be masked by the voters, another SEU in the feedback loop is not guaranteed to be masked. To fix the counter design, the output of the voters will need to be fed back to the input of counters to remove the persistent cross-section.

To circumvent issues with the synthesis tools, the recommended approach for applying TMR is to apply TMR to the EDIF circuit descriptions while this can be done in a text editor for small designs, the authors suggest using one of the two automated tools (BL-TMR [17] and TMRTool [18]). As these tools work with the post-synthesis circuit representation, the synthesis tools are able to optimize the basic circuit without affecting the application of TMR. The optimization of the circuit after synthesis is usually limited to removing signals that do not route to output pins. Therefore, optimization of the redundant modules is unlikely. Also, these tools have been built with an understanding of persistence issues so that feedback loops are properly protected by TMR.

B. Device Constraint Problems

The second issue regards design constraints. Since these devices can be pin- and area-constrained, designers are sometimes unable to implement a fully triplicated design. In particular, not being able to triplicate input, output, clock or reset signals is common, and SEUs in the input/output blocks, routing, global clock network, and flip-flops could cause errors to manifest across all three logic modules. The counter in Figure 1 shows the three counters are sharing the same inputs. While this design is not uncommon in cases where the data stream originates from a single sensor, unprotected cross-section exists between the input pins and the inputs of the counters. Furthermore, we have found that, when not using automated tools to apply TMR to a design, that the optimization by the synthesis tools of the TMR-protected circuit with shared inputs is more likely to remove most of the reliability-based redundancy. While it is possible to triplicate some of these signals internally on the device[2], an unprotected cross-section still exists in the system between the input pins and the triplicated flip-flops responsible for splitting the signal.

Designers might also find themselves constrained by the device's size, and are unable to fully triplicate the circuit logic. The BL-TMR tool addresses this problem by balancing the need to protect the most essential parts of the design and meeting area constraints by applying TMR partially to the circuit. BL-TMR gives highest priority to sub-circuits that may reach a persistent error state due to feedback, since error recovery may require external intervention. In cases where TMR has only been partially applied to the circuit, there exists an unprotected cross-section. The effect of this unprotected cross-section can be hard to quantify.

C. Circuit Implementation and Architectural Problems

The third issue is the implementation of the circuit on the architecture. There are several problems that are directly tied to the placement of the circuit onto the device, such as domain crossing errors and logical constants. These devices are very complex and have a number of architectural components, such as the resources for fast carry-chains, shift registers, and embedded arithmetic functions, to improve the speed, power, and silicon utilization of user circuits. As an artifact of translating a design to the specific resources available on the FPGA, sometimes the inputs to carry-chains and multipliers need to be tied to a ground, as when the multiplication is using fewer inputs than the embedded multipliers have. These grounds are tied to a logical constant on the power network, called the *global logic network*. The power network is a virtual network of grounds and VCCs that use constant LUTs. Since the power network is load balanced by the tools, redundant logic could share the same power network, introducing potential single points of failure into the design. Further complicating the issue, the power network is implemented in SEU-sensitive logic, which could translate to a large, unprotected cross-section in the design. Since the load balancing affects the number of constant LUTs that are used, the exact quantity of single points of failure caused by them cannot be determined until after the design is placed.

Both BL-TMR and TMRTool tools address this issue by extracting the half-latches and the constant LUTs to input/output pins to provide these constant logic values in a TMR domain-aware manner. Since this solution elevates logical constants to a global signal, like the clock tree, the input/output pins used for the logical constants will need to be triplicated.

The final reliability problem involves the placement of the design on the device. Since many of the tools involved in converting a designer's circuit description to a bitstream are attempting to minimize the implemented circuit's area and maximize the clock speed, redundant logic can be placed in close proximity. We have shown in the Virtex-II that, when area and timing constraints cause the device to be highly utilized, there is a chance an MBU can defeat TMR by introducing errors into multiple redundant modules, a situation referred to as a domain crossing event [4]. In this study, we observed Virtex-II circuits that had no single bit upset (SBU) cross-section, but still had a DCE cross-section. We are currently investigating recent test results that indicate that domain crossing errors from single-bit SEUs are possible on the Virtex-4.

IV. Circuit Reliability Analysis

STARC was designed to address the limitations of traditional reliability modeling tools in modeling user circuits for FPGAs, as well as address domain-specific issues with implementing TMR in FPGA circuits. In the past, this tool has been used to model both the reliability of supercomputers in the presence of neutron radiation [19] and nano-scale electronics in the presence of permanent yielding defects [20]. The main drivers for STARC are usability, computational complexity, scalability and modularity. STARC addresses these limitations with these solutions:

[2] Clocks should only be triplicated using the global clock buffers, and skew should be carefully monitored.

- **Usability:** the industry-standard EDIF circuit representation is used for the input model, and input vector sets are not used. STARC was also designed to assess domain-specific problems of applying TMR to FPGA user circuits and can detect imbalances between the modules, find untriplicated logic, estimate unprotected cross-section, and detect logical constant usage.
- **Computational Complexity:** memoization of reliability values reduces recomputation of similar components, and the use of combinatorial reasonings simplifies the reliability calculation.
- **Scalability:** without input vectors the state space scales linearly with the circuit size.
- **Modularity:** the architectural and fault models that provide the basis of the reliability calculation are inputs to STARC and can be replaced with user-specified architectures and fault models.

By using the EDIF circuit representation, the designer can assess the reliability of a circuit during the design process, even if the design is not complete, the design does not work, or the hardware is not available. Without the use of input vector sets reliability is determined through the probability of device or input failure and is not dependent on specific input data sets. Without input data sets, the reliability of components are determined by type, such as a two-bit adder, and can be memoized for reuse. In this manner, large-scale circuits are analyzed in a fraction of the time and memory required by traditional approaches, making design exploration more worthwhile.

As STARC can estimate the hardness assurance of FPGA user circuits within minutes, STARC can also be used for designers facing area and resource constraints. Under these circumstances, it is possible to generate a range of designs in BL-TMR with different balances of unprotected cross-sections and resource utilization. In this manner, STARC can help designers choose among a range of possible design choices by quantifying the remaining unprotected cross-section for each.

There are a few disadvantages to this approach. First, since EDIF does not contain information about the routing, information regarding placement and routing is absent from the calculation. As routing can have a large impact on the protected and unprotected cross-sections, the routing cross-section is estimated statistically based on an analysis we did of several designs using JBits [21]. The point of the statistical model is to provide a good estimate of the single-bit cross-section, as the only way to fix unprotected configuration bits in the routing is to mitigate the unprotected logic. Furthermore, currently there is no way to assess placement-related issues, such as MBU-induced TMR defeats. We are currently working on a solution for this limitation for designs that have completed the design flow. Second, without input vector sets, logic masking cannot be taken into account, and STARC estimates the worst case failure rate. While this value may be lower than the value determined by other tools [14], STARC provides a useful lower bound on the circuit's reliability.

A. STARC Overview

In the remaining half of this section, we will provide an overview of STARC. The reliability of the circuit is determined from dependency graphs of the circuit that are created during a hierarchical exploration of the circuit. By using the EDIF circuit representation, the hierarchy in the circuit should be preserved. Since designers tend to create complex circuits by creating less complex components or sub-circuits, maintaining this structure can be very useful in calculating the reliability. In particular, STARC can determine the reliability of a circuit hierarchically. STARC navigates through the layers of the circuit hierarchy to determine the smallest circuit component that needs to have it's reliability calculated. Once an entire layer of the circuit hierarchy is completed, these values can be used to determine the reliability of the next higher layer. This hierarchical nature allows circuits to be examined at the highest level of abstraction or the most minute level of detail. STARC automatically determines the appropriate level of the hierarchy that needs to be explored.

Since input vectors are not used in the reliability calculation, the reliability is determined by component type. For example, one component type might be a two-bit adder. The first time a two-bit adder is found during hierarchical exploration these three steps are executed:

1) a dependency graph is determined,
2) the reliability of the dependency graph is calculated, and
3) the reliability value of the dependency graph is memoized.

The next time another two-bit adder is found in a design, the memoized value is used, and the first two steps of the process are eliminated. It is in this way the state space of the circuit grows with circuit size, since the state space is limited to the unique number of components in the circuit. Even if a circuit has very little component reuse, the state space will never grow larger than the number of components in the circuit. Since the size of the state space has a first order effect on the speed of computation, STARC is able to analyze the reliability of a circuit in polynomial time, instead of the exponential time necessary for most traditional reliability tools. Therefore, STARC should be able to compute the reliability of circuits with thousands of components in the design in a matter of minutes.

As stated above, during hierarchical exploration dependency graphs are determined for each unique component. For maximum re-use, dependency graphs for each primary output at each level of the hierarchy is determined. These dependency graphs indicate all of the components that exist in the path between a particular output and the reachable inputs. Since not all logic or inputs are reachable from every output, this technique removes unrelated logic from dependency graph and, hence, the reliability calculation.

Once the dependency graph for an output is determined, the reliability can be calculated. In unmitigated designs, the cross-section is the total area of the dependency graph:

$$A(O) = \sum_{i=0}^{m} A(C_i), \tag{1}$$

where $A(X)$ is the sensitive area of X (where X is either a wire or a component) and $C = \{C_0, ..., C_m\}$ is the set of components that can be reached from output wire O. STARC also applies a modular approach to the fault model and the architectural model. Since reliability is determined hierarchically in STARC, the only devices that need to be pre-calculated are the primitives for the given architecture. Figure 2 shows our methodology for library characterization. The primitives for a hardware platform are defined in an architectural model. Fault models for transient and permanent defects are combined with the architectural models to create the characterized primitive library. Traditional probability of failure equations are also available to calculate the reliability of defect-based architecture models. Our automation framework is designed so that users can define primitive libraries for their own architectural models or use our models for basic logic and the Xilinx architecture. To be used in our methodology, user-defined libraries have to be characterized for specific fault models to define their reliability.

In this manner, STARC was designed to be architecturally independent. While this paper focuses on reliability as it relates to Xilinx FPGAs, STARC is modular in nature and the Xilinx cross-section model is an input to this system. The tool has also been used for probability of failure calculations for nanoscale electronics based on yield estimates. In the future, we would like to expand into models for probability of failure and cross-section models for structured application-specific integrated circuits (ASICs), as these devices are frequently being used in space-based systems as well.

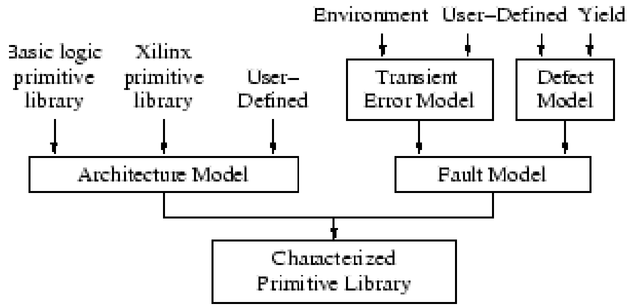

Fig. 2. Library Characterization

Finally, STARC was also designed to help designers find problems in the application of TMR. For mitigated circuits, the sensitive area is confined to the part of the design that is not triplicated, as triplication will mask errors as long as there is one voter for each redundant module. STARC also checks to make certain the modules have equivalent components. Any logical elements that might be shared by two or more TMR domains are considered unprotected cross-section, even if the elements reside within one of the modules. STARC also checks to make sure the feedback loops are properly triplicated and cut. If persistent cross-section is found, a warning is displayed to inform the designer that a particular component has not had TMR applied correctly.

In all of these cases, STARC provides warnings and information about the design to the designer. The output of the tool provides the designer a list of sub-circuits that are untriplicated, a quantity for the unprotected cross-section, and warnings about potential single points of failures from functionally nonequivalent modules and logical constants. Since EDIF is tightly coupled to the circuit design, the designer should be able to directly use STARC's output to find and fix the design flaws in the user circuit.

V. Case Study: Tradespace of Reliability Issues under Area Constraints

In this section we present a case study of two image processing algorithms that use STARC to explore the tradespace of reliability issues under an area-constrained design process. The two image processing algorithms we examined are an edge detection algorithm and a noise filtering algorithm. The edge detection algorithm uses the Sobel convolution masks [22] as the computational basis. These convolution masks are well-matched to FPGA implementation, since the multiplication can be reduced to shifts. The noise filtering algorithm breaks the image into a series of small windows. The pixel in the center of the window is replaced with the minimum pixel value

TABLE I
STARC RESULTS FOR TWO IMAGE PROCESSING ALGORITHMS

Design	Implementation	Total Unprotected Cross-Section (bits)	Unprotected Logic (bits)	Unprotected Routing (bits)	Number of Components	Time to Calculate (sec)
Edge Detection	No TMR	15,418	3,641	11,777	1,356	56
	Partial TMR (1)	21,800	19	21,781	3,787	426
	Partial TMR (2)	24	16	8	3,793	401
	Full TMR	0	0	0	3,799	230
Noise Filter	No TMR	14,914	4,522	10,392	1,603	95
	Partial TMR (1)	14,332	19	14,313	4,273	785
	Partial TMR (2)	24	16	8	4,279	565
	Full TMR	0	0	0	4,285	309

in the window. Both of these circuits are feed forward, and, therefore, do not have error persistence issues. Since both algorithms use nine eight-bit pixels as input, the algorithms both use the same data input circuit.

Several implementations of these circuits were developed: without TMR, with full TMR, and two partial TMR approaches. To avoid design issues with applying TMR, BL-TMR was used. It should be noted that STARC has been modified to automatically recognize designs that have been mitigated through BL-TMR. Support for the Xilinx TMRTool is currently being added. Logical constants were also extracted to input pins. For the partial TMR approaches, we had BL-TMR triplicate the logic in both implementations for both algorithms and varied how the input and output signals were handled. In the partial TMR 1 implementations we had BL-TMR not triplicate any input or output signals, and in the partial TMR 2 implementations we had BL-TMR triplicate only the reset, logical constant and clock input signals.

STARC was used to determine the unprotected cross-section of all of the implementations, as shown in Table I. The first thing to note from these values is that applying TMR to just the design's logic (partial TMR 1) provided little improvement for the noise filter and actually increased the cross-section for the edge detection algorithm. When we looked through the STARC results we found the large unprotected cross-section in the partial TMR 1 versions were due to the unmitigated signals. As Table I shows, all of the unprotected cross-section for these implementations are in the routing network, indicating that the logic was properly triplicated. Since the triplicated logic has three times as many flip-flops, the untriplicated clock, reset, and logical constant trees now have to route to three times as many locations. In a heavily pipelined design, like the edge detection algorithm, this decision was disastrous. When we went back to BL-TMR and chose to triplicate the logic and the global signals, the unprotected cross-section for both designs was 99.8% smaller than the unprotected cross-section in the unmitigated design. When full TMR is applied to both algorithms, there was no unprotected cross-section.

TABLE II
STARC VALIDATION RESULTS FOR THE UNMITIGATED IMPLEMENTATION OF TWO IMAGE PROCESSING ALGORITHMS

Design	Total Unprotected Cross-Section (bits)	Unprotected Logic (bits)	Unprotected Routing (bits)
Edge Detection	14,461	2,291	12,170
Noise Filter	9,507	1,462	8,045

Finally, STARC was able to find the hardness assurance issues that existed in the implementations without TMR and with partial TMR. In both algorithms the implementations without TMR used the device-provided logical zeros and STARC correctly identified this as a potential problem. Also, the implementation of the two algorithms with partial TMR had input signals, a voter, and input/output registers that were not triplicated. STARC was able to find these untriplicated signals and logic, report them, and properly calculate the cross-section for them.

We have recently begun validation of the STARC tool. Table II shows some results from fault injection of the unmitigated implementations of the two image processing algorithms. While the edge detection algorithm is within 93.8% of the STARC-predicted cross-section, the noise filtering algorithm is not as close at 63.8%. When looking at the numbers closer, for both designs the routing estimates look reasonable, but the logic is over-estimated in both cases. We believe that the reason why there is such a gap in the logic values is due to logical masking on the fault injection hardware. In particular, we found that the outputs of the edge detection algorithm are much more sensitive to data changes than the noise filter. In examining the execution times we found that the tool was able to compete on average 12 components/second.

Note that the execution time tripled from the unmitigated implementations to the mitigated implementations. As BL-TMR flattens the circuit hierarchy while applying TMR, the entire circuit's state space must be analyzed to determine the reliability of the circuit.

VI. Conclusion

In summary, we have presented a number of hardness assurance pitfalls with TMR-protected designs for the Xilinx devices, including redundant modules that share logic, the inability to fully triplicate designs, and device-provided logical constants. We have also introduced a tool, called the Scalable Tool for the Analysis of Reliable Circuits (STARC), that automates the process for identifying hardness assurance issues with TMR-protected circuits for Xilinx FPGAs as well as estimating their unprotected SEU cross-sections. As an illustration, we used STARC to analyze four implementations of two different image processing algorithms with different approaches to TMR. While we found that full TMR provided a 100% reduction in cross-section, triplicating only the internal logic either barely reduced or increased the cross-section of the original design. We found that not triplicating the reset and clock signals for TMR-protected logic had a profound effect on the cross-section, and that triplicating just those two signals with the logic could reduce the unprotected cross-section by 99.8%.

References

[1] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex FPGAs," Xilinx Corporation, Tech. Rep., November 1, 2001, XAPP197 (v1.0).

[2] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis, "A Fault Injection Analysis of Virtex FPGA TMR Design Methodology," in *Proceedings of the 6th European Conference on Radiation and its Effects on Components and Systems (RADECS 2001)*, 2001.

[3] N. Rollins, M. Wirthlin, M. Caffrey, and P. Graham, "Evaluating TMR Techniques in the Presence of Single Event Upsets," in *Proceedings fo the 6th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*. Washington, D.C.: NASA Office of Logic Design, AIAA, September 2003, p. P63.

[4] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, and K. Lundgreen, "Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs," *IEEE Transactions on Nuclear Science*, Vol. 54, No. 6, pp. 2037 – 43, 2007.

[5] 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," *Transactions on Nuclear Science*, Vol. 54, No. 6, pp. 2576–2583, 2007.

[6] J. A. Abraham, "A Combinatorial Solution to the Reliability of Interwoven Redundant Logic Networks," *IEEE Transactions on Computers*, Vol. 24, No. 6, pp. 578–584, May 1975.

[7] J. A. Abraham and D. P. Siewiorek, "An Algorithm for the Accurate Reliability Evaluation of Triple Modular Redundancy Networks," *IEEE Transactions on Computers*, Vol. 23, No. 7, pp. 682–692, July 1974.

[8] C. Hirel, R. Sahner, X. Zang, and K. Trivedi, "Reliability and Performability Using SHARPE 2000," in 11^{th} *Int'l Conf. on Computer Performance Evaluation: Modeling Techniques and Tools*, Vol. 1786, 2000, pp. 345–349.

[9] F. V. Jensen, *Bayesian Networks and Decision Graphs*. New York: Springer-Verlag, 2001.

[10] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices," in *Design, Automation and Test in Europe (DATE'05)*, Vol. 1. New York, NY, USA: ACM Press, 2005, pp. 282–287.

[11] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, "Evaluating the Reliability of NAND Multiplexing with PRISM," *IEEE Transactions on CAD*, Vol. 24, No. 10, pp. 1629–1637, 2005.

[12] D. Bhaduri, S. K. Shukla, P. S. Graham, and M. B. Gokhale, "Reliability Analysis of Large Circuits Using Scalable Techniques and Tools," *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, Vol. 54, No. 11, pp. 2447 – 60, November 2007.

[13] D. Bhaduri, S. K. Shukla, P. Graham, and M. Gokhale, "Comparing Reliability-Redundancy Trade-offs for Two Von Neumann Multiplexing Architectures," *IEEE Transactions on Nanotechnology*, 2006.

[14] D. Bhaduri and S. Shukla, "NANOLAB—A Tool for Evaluating Reliability of Defect-Tolerant Nanoarchitectures," *IEEE Transactions on Nanotechnology*, Vol. 4, No. 4, pp. 381–394, 2005.

[15] D. Bhaduri and S. Shukla, "NANOPRISM: A Tool for Evaluating Granularity vs. Reliability Trade-Offs in Nano-Architectures," in *14th GLSVLSI*. Boston, MA: ACM, April 2004, pp. 109–112.

[16] M. Baze, S. Buchner, W. Bartholet, and T. Dao, "An SEU Analysis Approach for Error Propagation in Digital VLSI CMOS ASICs," *IEEE Transactions on Nuclear Science*, Vol. 42, No. 6, pp. 1863–9, December 1995.

[17] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "SEU-Induced Persistent Error Propagation in FPGAs," *IEEE Transactions on Nuclear Science*, Vol. 52, No. 6, pp. 2438 – 45, 2005.

[18] "Xilinx TMRTool User Guide," on web: http://www.xilinx.com/products/milaero/ug156.pdf.

[19] H. Quinn, D. Bhaduri, C. Teuscher, P. Graham, and M. Gohkale, "The STAR Systems Toolset for Analyzing Reconfigurable System Cross-Section," in *Military and Aerospace Programmable Logic Devices*, 2006, p. 162.

[20] H. Quinn, D. Bhaduri, C. Teuscher, P. Graham, and M. Gohkale, "The STARC Truth: Analyzing Reconfigurable Supercomputing Reliability," in *Field-Programmable Custom Computing Machines*, 2005.

[21] "http://www.xilinx.com/labs/projects/jbits/."

[22] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall Information and System Sciences Series, 1989.