Improving FPGA Design Robustness with Partial TMR

Brian Pratt, Michael Caffrey, Paul Graham, Keith Morgan, Michael Wirthlin

***Abstract*— This paper describes an efficient approach of applying mitigation to an FPGA design to protect against Single Event Upsets (SEUs). This approach applies mitigation selectively to FPGA circuit structures depending on their importance within the design. Higher priority is given to structures causing "persistent" errors within the design. For certain applications, applying selective mitigation to the persistent components can yield higher returns in reliability per unit cost than full mitigation. A software tool is also introduced which automatically classifies circuit structures based on this concept and applies Triple Modular Redundancy (TMR) selectively based on the classification of the circuit structure.**

***Index Terms*— SEU, FPGA, TMR, persistence, error propagation, simulator, radiation, selective mitigation**

I. Introduction

Field-programmable gate arrays (FPGAs) are an increasingly attractive solution for space systems. They perform well in high-throughput signal processing applications often used in space. Furthermore, their programmability allows in-field application adjustments.

Unfortunately, FPGAs are susceptible to radiation-induced single-event upsets (SEU). Since FPGAs store their programming data, or configuration in an SRAM-like configuration memory, an SEU can actually alter the intended circuit. As such, SRAM-based FPGAs destined for a radiation environment must employ a form of SEU mitigation to insure reliable operation [1]–[3].

SEU sensitivity in the configuration memory can be mitigated through techniques such as Triple Modular Redundancy (TMR) [1], [2]. These mitigation strategies offer tremendous improvements in reliability, but are often expensive in terms of FPGA resource utilization, power consumption, etc. [1], [4]. We seek new ways to offer acceptable levels of reliability at much lower costs.

One way to reduce the cost of mitigation is to apply mitigation selectively on a subset of an FPGA design. By selectively applying design mitigation, the designer can trade off the improvements in reliability with the cost of design mitigation. In this paper we will show how applying TMR to a subset of the user design allows the designer to most effectively make this trade-off. Applying TMR selectively to a user design is called "Partial TMR". This paper also introduces a tool for *automatically* applying TMR selectively and reports on the improvements in reliability obtained from this tool.

This work was supported by the Department of Energy at Los Alamos National Laboratory. Approved for public release under LA-UR-06-0339; distribution is unlimited.

B. Pratt and K. Morgan are with both Los Alamos National Laboratory and Brigham Young University.

P. Graham and M. Caffrey are with Los Alamos National Laboratory, Los Alamos, NM.

M. Wirthlin is with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT.

II. FPGA Single Event Effects

Much like SRAM and DRAM memory, programmable FPGAs contain a large number of memory cells susceptible to radiation-induced upset. Within FPGAs, the vast majority of memory cells are devoted to the configuration memory. Configuration memory upsets are particularly troublesome since they may change the users's circuit, possibly altering the function of configurable logic, I/O, or other resources as well as changing the structure of the routing network.

One technique used to reduce the effects of upsets within the configuration memory is called *configuration scrubbing* [5]. Configuration scrubbing involves a periodic refresh of the FPGA's programming data while the FPGA is operating. Configuration scrubbing prevents the build-up of multiple configuration faults and reduces the time in which an invalid circuit configuration is allowed to operate. Furthermore, systems must employ configuration scrubbing for redundancy-based mitigation techniques such as TMR before any reliability enhancement is observed. Without scrubbing, the build-up of multiple faults would eventually break the redundancy.

A. FPGA Design Sensitivity

Although a modern FPGA contains a large number of configuration memory cells, not all of the cells are utilized by every design. Configuration cells that are not utilized by a specific design will not affect the behavior of that design. A configuration memory cell that impacts the behavior of a particular design is called a *sensitive* configuration bit. The set of sensitive configuration bits is unique for each design and depends on the logic, I/O, routing, and other FPGA resources used by the given design.

In a previous effort, we developed a fault injection tool that identifies the *sensitive* configuration bits of the device for any given FPGA design [6], [7]. The tool operates by artificially injecting faults within the configuration bitstream and monitoring the behavior of the device. By comparing the behavior of the device under test against a golden device, we can determine when the device behavior changes. By testing every configuration bit of the device, a detailed sensitivity profile of a given design can be created.

This fault injection tool has been used to characterize the sensitivity of many FPGA designs. A sample of these results

0-7803-9498-4/06/$20.00 ©2006 IEEE

IEEE 06CH37728 44th Annual International Reliability Physics Symposium, San Jose, 2006

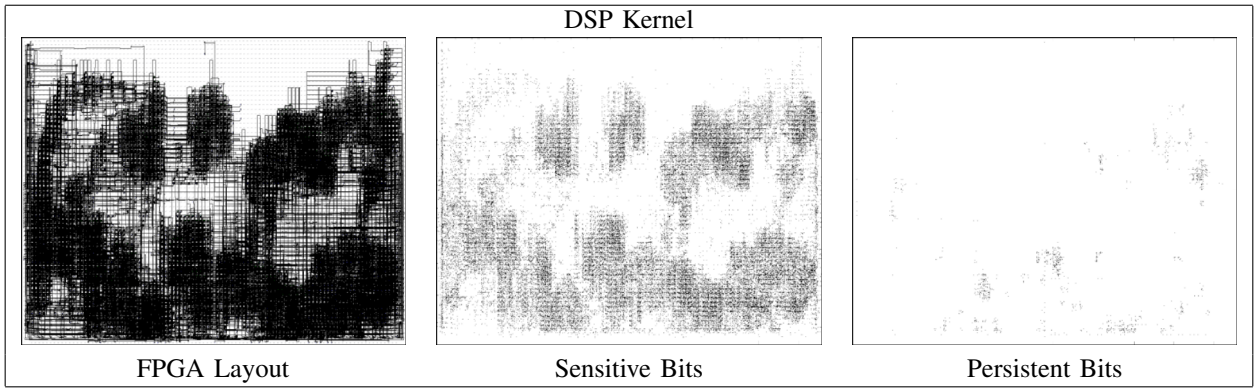

Fig. 1. The diagram on the left is a screen capture of the layout of the DSP Kernel design. The center and right diagrams are graphical representations of the portion of the DSP Kernel design layout which correspond to the sensitive and persistent bits respectively.

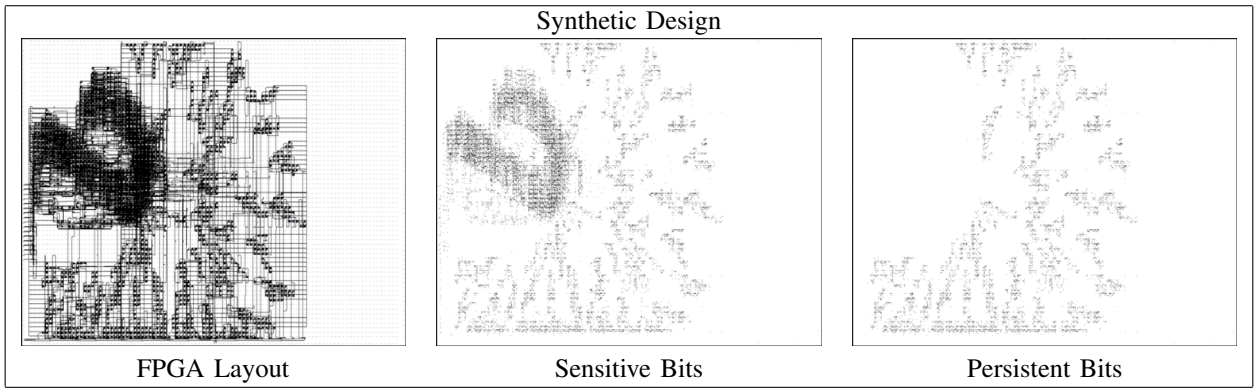

Fig. 2. The diagram on the left is a screen capture of the layout of the Synthetic design. The center and right diagrams are graphical representations of the portion of the Synthetic design layout which correspond to the sensitive and persistent bits respectively.

is shown in Table I. For each design tested, the table provides the utilization (in logic slices and as a percentage) and the number of sensitive configuration bits. In addition, it provides an overall "sensitivity" measure that indicates the percentage of the total set of configuration bits that are sensitive in the design. It is important to note that configuration sensitivity is design dependent and must be characterized on a design by design basis.

Design	Utilization (Slices)	Sensitive Bits	Persistent Bits
DSP Kernel	5,746 (46.8%)	575,448 (9.9%)	13,841 (0.24%)
Synthetic	2,538 (20.6%)	189,835 (3.3%)	77,159 (1.3%)
Multiplier	10,305 (83.9%)	550,228 (9.5%)	0 (0%)
Counter	2,151 (17.5%)	201,691 (3.5%)	108,750 (1.9%)

TABLE I
CONFIGURATION SENSITIVITY AND PERSISTENCE FOR SEVERAL DESIGNS.

Figures 1 and 2 visually demonstrate the concept of configuration sensitivity for two different FPGA designs. The first design (Figure 1) is a digital signal processing (DSP) kernel developed at Los Alamos National Laboratory. The second design (Figure 2) is a synthetic design made from feedback shift registers (LFSRs) that feed an array of multipliers and adders.

The left most figure for both design representations is a depiction of the logic resources used by the design. This was obtained by taking a screen capture of `fpga_editor`, the resource editor software tool provided by Xilinx. The middle figure is a plot of the sensitive configuration bits within the device as determined using the fault injection tool. Points marked in this plot indicate a sensitive configuration bit. Note the correspondence between the resource utilization and sensitivity map. As expected, sensitive configuration bits are located in areas where the FPGA device is utilized.

III. CLASSIFICATION OF SENSITIVE CONFIGURATION BITS

To successfully pursue partial mitigation techniques, we need a method of more finely categorizing or ranking the sensitive configuration bits. Since mitigation techniques will be applied to only a subset of the circuit, only a subset of the corresponding sensitive configuration bits will be addressed. To maximize the benefit of partial mitigation techniques, the importance of the all circuit structures will be ranked and mitigation will be applied to the most important circuit structures in the design. This section will describe a method for characterizing the sensitive configuration bits into two categories. We will use this classification to direct our partial mitigation strategy.

A. Persistent Configuration Upsets

In [8] and [9] we introduced a new way to categorize the sensitive configuration bits by separating them into two

categories called "persistent" and "non-persistent". A non-persistent configuration bit is a sensitive configuration bit that will cause a design fault when upset through radiation. This fault will introduce functional errors. When the non-persistent configuration bit is repaired through configuration scrubbing, however, the design returns to normal operation. Eventually all previously induced functional errors will disappear. No additional intervention is required to return the circuit to normal functionality.

A persistent configuration bit is a sensitive configuration bit that will cause a design fault when upset. However, even after repairing persistent configuration bits through configuration scrubbing, the FPGA circuit *does not* return to normal operation. Upsets of persistent configuration bits introduce functional errors which persist after the bit is repaired through scrubbing. Upsets of persistent bits put the design into an incorrect state that cannot self-correct. In this case, a global reset is needed to return the circuit to a proper state, or normal operation. This global reset takes the circuit offline for the time needed to reset the circuit and start up in normal operating mode. For brevity, we refer to an upset of a persistent configuration bit as a "persistent upset" and that of a non-persistent bit as a "non-persistent upset."

The differing behavior of the two types of sensitive configuration bits can be seen in the plots of Figure 3. Each figure plots the arithmetic difference of the output signal between the design under test (DUT) and a golden circuit design as captured by our fault injection simulation environment. When the two circuits are operating correctly, the arithmetic difference is zero. When one of the circuits has functional errors, the arithmetic difference is non-zero.

The left plot demonstrates the behavior of non-persistent upsets. After an upset, the difference between the two circuits is non-zero. The output of the corrupted design is incorrect. Once repaired through scrubbing, the difference returns to zero since the corrupted circuit returns to normal behavior.

The right plot demonstrates the behavior of persistent upsets. Almost immediately after the configuration upset, the arithmetic difference is non-zero. The outputs of the DUT and golden circuits do not match. In this case, even after the upset bit is repaired through scrubbing, the corrupted circuit continues to generate incorrect output. In other words, the circuit fails to return to normal operation. Only a system reset will restore the circuit to a proper state.

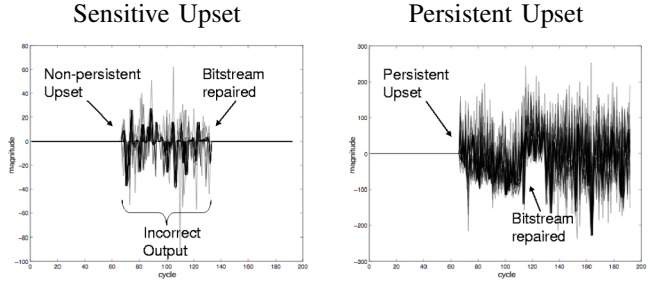

Fig. 3. The left and right plots show the delta between the outputs of a DUT and golden circuit before, during and after a sensitive and persistent upset respectively.

The fault injection tool of [7] was augmented to measure configuration persistence. The tool begins by determining the sensitive configuration bits within the design. Once the sensitive configuration bits are identified, the tool performs additional tests to identify the subset of sensitive configuration bits that are also persistent configuration bits. Table I lists the configuration persistence for several designs tested within the fault injection framework. In some designs, the persistent component is a major portion of the design (e.g. Synthetic) while the persistence in other designs is relatively small (e.g. Multiplier). The persistence of two designs can be seen visually in the right-most plot of Figure 1 and Figure 2. Frequently, the persistent configuration bits represent a small fraction of the total set of configuration bits within the device.

B. Persistent Circuit Structures

The circuit structures that correspond to persistent configuration bits can be identified statically by analyzing the structure of the circuit netlist. Specifically, circuit primitives that are part of feedback structures within the design contribute to the persistent error behavior. If a circuit fault occurs within a feedback structure, the incorrect values produced by the faulty circuit are propagated into the feedback state. Once the state of the feedback circuit has been corrupted, the circuit may not behave correctly until the circuit state has been reinitialized with a global reset. Although configuration scrubbing will repair the faulty circuit, it may not restore the proper circuit state.

The sample schematics in Figure 4 illustrate the major subsections of a simple circuit that may cause persistent configuration faults. Figure 4(a) highlights a feedback structure in a sample design. A fault within this feedback structure will generate and/or indefinitely propagate incorrect values within the state flip flops. Figure 4(b) highlights the logic which feeds *into* feedback structures. Upsets within this section may also cause persistent configuration faults. Logic calculations made in these feedback input structures contribute to the state values within the feedback structure circuit. Figure 4(c) highlights logic structures driven by a feedback structure or that operate independently of feedback. Upsets within these sections do not cause persistent faults. To maximize the benefits of partial mitigation, redundancy should be added to the components highlighted in Figure 4(a) and (b) before addressing those highlighted in Figure 4(c).

IV. Partial Mitigation

SEU mitigation is essential to ensure absolute reliable operation of FPGA devices in a radiation environment. Triple Modular Redundancy is one of the common methods of SEU mitigation. It has been shown to greatly reduce the dynamic cross section of an FPGA design and, when combined with bitstream scrubbing, virtually eliminate the configuration bitstream from SEU susceptibility [1], [2]. Therefore, TMR can greatly decrease the downtime of a circuit in radiation environments.

Full mitigation of an FPGA design using techniques such as TMR, however, is costly in terms of FPGA resource

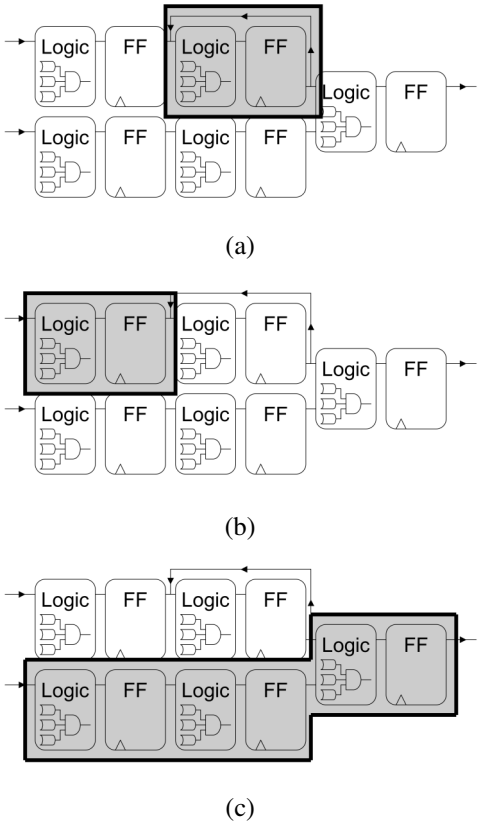

Fig. 4. A simple representation of a circuit with different sections highlighted (a) The feedback section (b) The input to the feedback section (c) The feed-forward logic section

utilization, power consumption, and circuit performance. Many user designs, in fact, cannot be mitigated with full TMR due to the more than $3\times$ increase in design resource utilization.

An attractive alternative to full mitigation is to mitigate only the most critical sections of a design. This reduces mitigation costs while efficiently reducing circuit downtime [3]. By selectively applying mitigation to a design, one can find the most effective balance between mitigation cost and reliability.

A. Partial Mitigation With Respect to Persistence

We can leverage the concept of persistence, discussed in Section III, to efficiently apply partial mitigation to an FPGA design. Since a persistent upset causes a permanent interruption in service, we would like to first apply mitigation to persistent circuit components. A non-persistent upset simply causes a temporary service interruption, thus non-persistent circuit structures are a lower mitigation priority.

A partial mitigation strategy based on the concept of persistence should rank the different structures of a circuit by their contribution to the persistence of a design. The more critical sections of a design should be mitigated first, with those of lower priority following as constraints allow. Referring back to Figure 4, the feedback structures of the design should be mitigated first. Any logic feeding into the feedback structures should follow since these contribute to the state of the design and thus the persistence. The feed-forward logic—the non-persistent circuit components—does not contribute to the persistence of a design and should be mitigated last.

To be clear, only certain FPGA applications are candidates for partial mitigation. The non-persistent portion of the sensitive configuration bits remains after partial mitigation. The output of a system may be incorrect after an SEU in the remaining set of unmitigated bits. However, applications that can tolerate this brief data loss stand to realize tremendous gains in mean time between failures (MTBF).

An application that can tolerate temporary data loss may still fail after a "persistent" bit is upset. As discussed in Section III, these upsets cause the circuit to enter an incorrect state, which will not self-correct, even after configuration scrubbing. However, these tolerant circuits may take advantage of incremental partial mitigation, starting with the persistent circuit structures. Valuable circuit resources are utilized to first eliminate failure. In Section V we will show that, in some cases, non-linear improvements in MTBF can be achieved at a linear cost.

B. BYU-LANL Partial TMR Tool

In cooperation with Los Alamos National Laboratory (LANL), a software tool was developed at Brigham Young University (BYU) to automatically apply partial mitigation on any EDIF-format design.The BYU-LANL Triple Modular Redundancy (BLTmr) tool uses TMR and the concepts presented in this paper to increase the uptime of the design. The tool requires minimal user intervention.

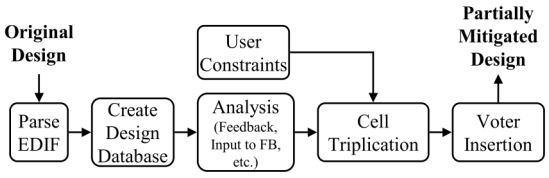

Fig. 5. Basic flow of the BYU-LANL Partial TMR (BLTmr) tool.

Figure 5 shows the basic flow of the BLTmr tool. The tool first parses input EDIF file(s) into a netlist data structure. The data structure is analyzed to identify feedback structures by searching for strongly connected graph components. Once all feedback structures are identified, the feedback input and feedback output structures are identified and classified. These circuit classes are separated for later analysis.

The BLTmr tool uses the classification information to select circuit structures for triplication. Based on the user and device constraints, the BLTmr tool selects as much of the feedback, feedback input, and output logic as possible. Once triplicated circuit elements are selected, the tool determines where in the design to insert the voters necessary to support TMR. The newly constructed circuit, which has the same functionality as the original, is then written to file in EDIF format. The new circuit structure can then be mapped to the corresponding FPGA technology.

Design	BLTmr Level	Slices	Utilization	Sensitive Bits	Sensitivity	Persistent Bits	Persistence
DSP Kernel design	Unmitigated	5,746	46.8%	575,448	9.9%	13,841	0.24%
	Feedback	7,276	59.2%	572,605	9.9%	5,074	0.087%
	Feedback & Input to FB	8,036	65.4%	569,700	9.8%	152	0.0026%
	Max TMR†	11,114	90.4%	556,062	9.6%	154	0.0027%
Synthetic design	Unmitigated	2,538	20.7%	189,835	3.3%	77,159	1.3%
	Feedback	9,867	80.3%	126,790	2.2%	806	0.014%
	Feedback & Input to FB	9,867	80.3%	126,790	2.2%	806	0.014%
	Full TMR‡	11,961	97.3%	20,256	0.3%	671	0.012%

TABLE II
MEASUREMENTS OF THE SENSITIVITY AND PERSISTENCE OF THE DSP KERNEL AND SYNTHETIC DESIGNS.
THE XILINX XCV1000 CONTAINS 12,288 SLICES AND 5,810,048 CONFIGURATION BITS.

† FULL TMR COULD NOT BE APPLIED DUE TO FPGA RESOURCE CONSTRAINTS.
‡ "FULL" TMR HERE DOES NOT INCLUDE TRIPLICATION OF THE CLOCK AND OUTPUT SIGNALS.

Our tool follows the basic decision-making process described in Section IV-A, prioritizing the application of TMR based on a circuit component's contribution to design persistence. The feedback structures of a design are of the most concern and thus are triplicated first. If resource limits allow, TMR is applied to the input to the feedback to further reduce persistence. Finally, mitigation is applied to the non-persistent circuit structures to reduce the remaining design sensitivity.

V. EXPERIMENTAL RESULTS

As a verification of our partial TMR tool, we used the tool in various configurations on two FPGA designs. The first is a digital signal processing (DSP) kernel developed at Los Alamos National Laboratory. The second is a synthetic design made up of linear feedback shift registers (LFSRs) that feed into an array of multipliers and adders.

A. Design Measurements

The BLTmr tool was used to investigate the improvements in reliability for various levels of partial TMR. Four different mitigation levels were tested for each of the two designs: first, an unmitigated design was created as a baseline (Unmitigated); second, TMR was applied to feedback structures only (Feedback); third, TMR was applied to feedback plus the input to the feedback (Feedback & Input to FB); and, fourth, TMR was applied to as much of the circuit as allowed by device density limitations (Full/Max TMR). The sensitive and persistent configuration bits were measured and plotted for each of these test cases.

The fault injection results for the DSP Kernel and Synthetic designs are summarized in Table II. For the DSP Kernel at the "Feedback" BLTmr level, the number of persistent bits decreased by 63%. At this level, hardware increased by 26%. At the "Feedback & Input to FB" BLTmr level, the number of persistent bits decreased by two orders of magnitude at a hardware cost of 40% over the unmitigated design. After mitigating the feedback and input to feedback, by definition, as much of the persistence as possible was eliminated. Note that at the next level, the number of persistent bits is virtually the same. We hypothesize that the two extra persistent bits at the "Max TMR" level are simply due to differences in the routing of the designs. Applying "Max TMR" does result in a small decrease in sensitivity over the previous level but at a hardware cost of over 3,000 slices.

The Synthetic design also demonstrated two orders of magnitude reduction in persistent configuration bits, but at the "Feedback" level. Since the number of persistent bits within feedback is higher in the DSP Kernel than the Synthetic design, the absolute hardware cost to eliminate persistence in the DSP Kernel is much lower. However, the number of persistent bits eliminated per added slice is actually higher in the Synthetic design. At the "Feedback & Input to FB" and "Full TMR" levels, the redundant logic does not significantly decrease persistence because no logic actually feeds into the feedback section of this design.

Figure 7 and Figure 8 show layouts, sensitivity plots, and persistence plots of the mitigated DSP Kernel and Synthetic designs, respectively. The figures of the DSP Kernel correspond to the "Feedback & Input to FB" mitigation level in Table II, while the figures of the Synthetic design correspond to the "Full TMR" mitigation level. Notice that the persistence of each design was virtually eliminated.

B. Mean Time Between Failures

An important motivation for measuring the sensitivity and/or persistence of a design is to determine how often a given system will fail. Like sensitivity and persistence, mean time between failures (MTBF) is application dependent. However, MTBF also depends on the destined system environment. To estimate MTBF we modeled the environment energy spectra for a few sample orbits. For the trapped proton and solar proton environments we used the AP-8 and JPL models respectively. For the heavy ion environment we used the CREME96 model.[1] We then used the Xilinx static proton and heavy ion cross section data for a single Xilinx Virtex

[1] A detailed explanation of the process and software we used to predict static SEU rates can be found in [10].

Orbit	Alt. (km)	Inc. (deg)	Unmitigated - Data Loss MTBF (days) Typical Solar Min	Stormy Solar Max	Worst Day Solar Max	Unmitigated - Persistent Failures MTBF (days) Typical Solar Min	Stormy Solar Max	Worst Day Solar Max	Partially Mitigated - Persistent Failures MTBF (days) Typical Solar Min	Stormy Solar Max	Worst Day Solar Max
LEO	560	35.0^o	22.9	34.4	32.8	950.5	1428.5	1365.6	86555.7	130077.0	124634.5
Polar	833	98.7^o	9.1	11.5	0.2	378.2	479.5	7.0	53350.3	51590.9	179.8
GPS	22,200	55.0^o	14.1	13.6	4.7×10^{-2}	585.9	566.6	2.0	53350.3	51590.9	179.8
GEO	36,000	0.0^o	13.9	11.4	4.7×10^{-2}	579.6	473.3	2.0	52781.9	43101.2	177.8

TABLE III

MEAN TIME BETWEEN FAILURES (MTBF) FOR THE DSP KERNEL DESIGN IN SEVERAL ORBITS FOR THREE DIFFERENT SITUATIONS: UNMITIGATED DESIGN FAILING WITH ANY SENSITIVE UPSET, UNMITIGATED DESIGN FAILING ONLY WITH PERSISTENT UPSET, PARTIALLY MITIGATED DESIGN FAILING ONLY WITH PERSISTENT UPSETS.

XCV1000 reported in [11], combined with the forecast energy spectra, to predict MTBF for our sample orbits [12].

Table III shows our predictions of MTBF for the DSP Kernel design. The first set of values (columns 4-6) shows the MTBF prediction for an application that does not tolerate any service interruptions. Applications in this category will "fail" after all dynamic upsets. The second set of values (columns 7-9) corresponds to applications that can function with temporary data-loss. Applications in this category "fail" only after persistent upsets. The final set of values in Table III corresponds to the DSP Kernel with mitigation applied to just persistent structures. Here too, the application will only "fail" after persistent upsets. However, they have been virtually eliminated.

Figure 6 is a plot of MTBF vs. resource utilization for the DSP kernel and Synthetic circuit designs in a GPS orbit. In this plot we treat the applications as tolerant of temporary data loss, therefore MTBF refers to failures caused by persistent upsets. This graph clearly shows a non-linear relationship between resources and MTBF improvement. In both cases, MTBF reaches a "saturation" point. After this point, additional mitigation logic primarily only reduces data-loss rates. For both designs saturation occurred after all persistent circuit elements were mitigated (since the feedback in the Synthetic design consists of input-less LFSRs, the feedback and feedback-plus-input circuits are the same). This indicates that, once the persistent circuit elements are triplicated, MTBF will saturate. Additional mitigation logic will only improve data-loss rates, not MTBF in terms of persistent upsets.

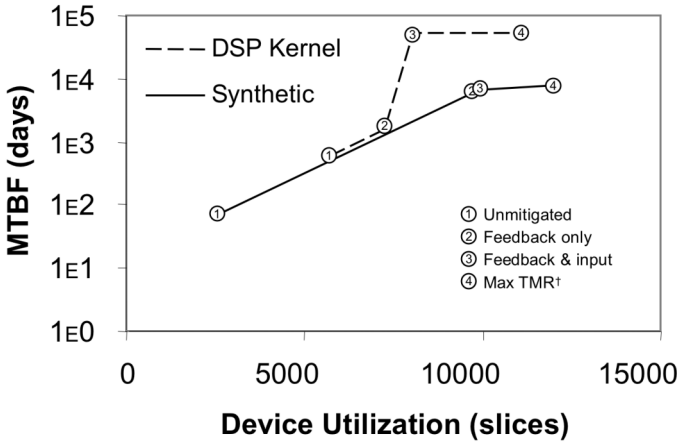

Fig. 6. Plot of MTBF vs. resource utilization for the DSP kernel and Synthetic circuit designs in a GPS orbit.

It is important to analyze the trade-offs made by applying TMR to just the persistence of a design. Since full TMR and other comprehensive mitigation techniques are costly in terms of area and power [1], [4], the positive benefit of partial TMR is a reduction in mitigation circuitry and, consequently, power required. For example, Table I shows that a completely unmitigated implementation of the DSP Kernel design utilized 5,746 slices. Full TMR would require at least a 200% increase. The partial TMR implementation done with the BLTmr tool, on the other hand, needed only 8,036 slices, just a 40% increase.

The negative trade-off for only applying partial TMR is that the non-persistent configuration bits are still vulnerable to SEUs. However, if only persistent upsets are considered failures, as with some applications, the system will only temporarily lose data after non-persistent upsets.

VI. CONCLUSIONS

The selective use of TMR on FPGA circuits was shown to provide improved reliability at a lower cost than full TMR. In fact, controlling the amount of triplication inserted into a design allows a designer to trade off the incremental cost of selective triplication with improvements in reliability. The BLTmr tool was created to perform this selective TMR automatically to a degree as directed by the user.

We used this tool to apply selective TMR to several designs and demonstrated the benefits of this approach. Results obtained from the BLTmr tool confirm that, for certain applications, MTBF can be more efficiently increased when mitigation is focused on the persistent structures of a design. Specifically, this tool improved the MTBF in terms of persistent failures of both designs by two orders of magnitude for a fraction of the cost of full TMR.

Efforts to improve the BLTmr tool and associated circuit analysis continues. Specifically, additional analysis approaches will be added to improve the classification of FPGA circuit structures. We are also investigating whether architecture-specific mitigation techniques may provide greater improvements in reliability at a lower hardware cost. Other efforts include more design testing, radiation testing for tool validation, and improvements in the user interface. We expect this approach to be used on several designs operating on a

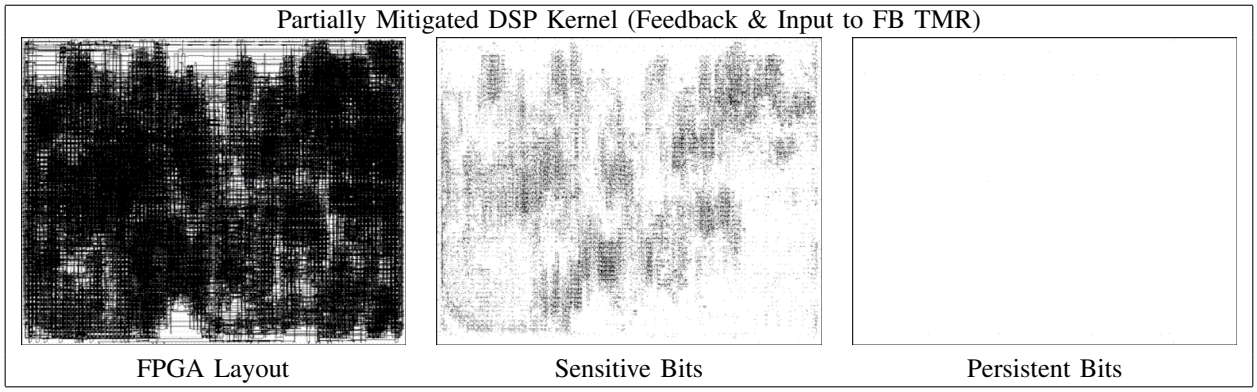

Fig. 7. The diagram on the left is a screen capture of the layout of a version of the DSP Kernel design which has been mitigated with the BLTmr partial mitigation tool. The center and right diagrams are graphical representations of the portion of the DSP Kernel design layout which correspond to the sensitive and persistent bits respectively.

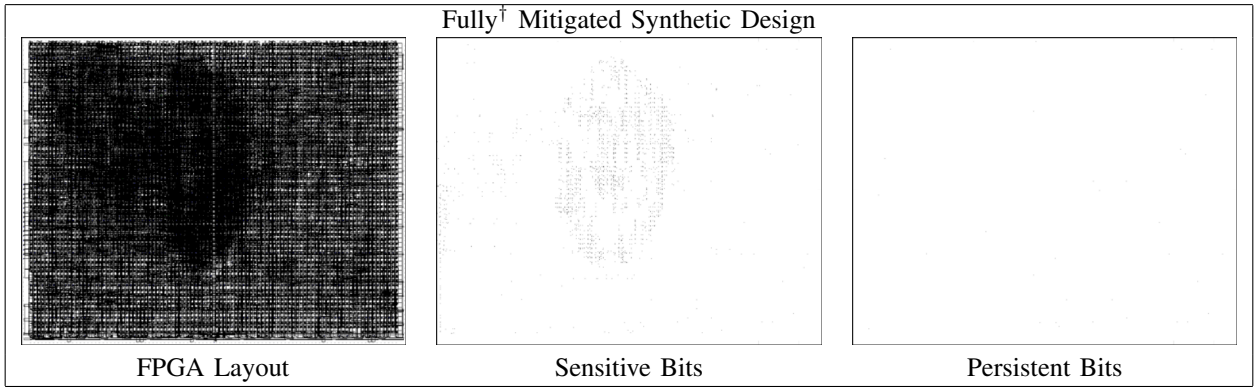

Fig. 8. The diagram on the left is a screen capture of the layout of a version of the Synthetic design which has been mitigated with the BLTmr partial mitigation tool. The center and right diagrams are graphical representations of the portion of the Synthetic design layout which correspond to the sensitive and persistent bits respectively.
† "Full" Mitigation here does not include triplication of the clock and output signals.

spacecraft to be launched later this year.

VII. Acknowledgment

We would like to acknowledge the support of Joe Fabula, Howard Bogrow and others at Xilinx. This work was funded by the Department of Energy through the Deployable Adaptive Processing Systems, the Cibola Flight Experiment, and the Sensor-Oriented Processing and Networking projects at LANL as well as by NASA through the Reconfigurable Hardware in Orbit project under AIST contract #NAG5-13516.

References

[1] 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., September 2003, NASA Office of Logic Design, AIAA, p. P63.

[2] C. Carmichael, "Triple module redundancy design techniques for Virtex FPGAs", Tech. Rep., Xilinx Corporation, November 1, 2001, XAPP197 (v1.0).

[3] P. K. Samudrala, J. Ramos, , and S. Katkoori, "Selective triple modular redundancy for SEU mitigation in FPGAs", in *Proceedings fo the 6th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, Washington, D.C., 2003, NASA Office of Logic Design, AIAA, p. C1.

[4] N. Rollins, M. Wirthlin, and P. Graham, "Evaluation of power costs in triplicated FPGA designs", in *Proceedings of the MAPLD Conference*, September 2004.

[5] C. Carmichael, M. Caffrey, and A. Salazar, "Correcting single-event upsets through Virtex partial configuration", Tech. Rep., Xilinx Corporation, June 1, 2000, XAPP216 (v1.0).

[6] M. Wirthlin, E. Johnson, N. Rollins, M. Caffrey, and P. Graham, "The reliability of FPGA circuit designs in the presence of radiation induced configuration upsets", in *Proceedings of the 2003 IEEE Symposium on Field-Programmable Custom Computing Machines*, K. Pocek and J. Arnold, Eds., Napa, CA, April 2003, IEEE Computer Society, p. TBA, IEEE Computer Society Press.

[7] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an FPGA SEU simulator", *IEEE Transactions on Nuclear Science*, vol. 50, no. 6, pp. 2147–2157, December 2003.

[8] 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*, December 2005.

[9] E. Johnson, K. Morgan, N. Rollins, M. Wirthlin, M. Caffrey, and P. Graham, "Detection of configuration memory upsets causing persistent errors in SRAM-based FPGAs", in *Proceedings of the MAPLD Conference*, September 2004.

[10] K. Morgan and M. Wirthlin, "Predicting on-orbit SEU rates", *http://dspace.byu.edu*, July 2005.

[11] "Qpro Virtex 2.5v radiation hardened FPGAs", Tech. Rep., Xilinx Corporation, November 5, 2001, DS028 (v1.2).

[12] E. Fuller, M. Caffrey, A. Salazar, C. Carmichael, and J. Fabula, "Radiation testing update, SEU mitigation, and availability analysis of the Virtex FPGA for space reconfigurable computing", in *3rd Annual Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, 2000, p. P30.