# Dynamic Scheduling of Voter Checks in FPGA-based TMR Systems

Nguyen T. H. Nguyen*, Dimitris Agiakatsikas*, Ediz Cetin†, and Oliver Diessel*
*School of Computer Science and Engineering, UNSW Australia
†School of Electrical Engineering and Telecommunications, UNSW Australia

***Abstract*—SRAM-based *Field-Programmable Gate Arrays* (FPGAs) are susceptible to radiation-induced *Single Event Upsets* (SEUs). Techniques for partially reconfiguring corrupted modules of *Triple Modular Redundant* (TMR) FPGA-based designs have been described in the literature. Most of these techniques require some form of network-on-chip for aggregating voter error messages from the system's TMR components to a central reconfiguration controller in order to trigger the partial reconfiguration of modules when they become faulty. The frequency at which TMR components fail in the system depends on their soft-error vulnerability. However, most error recovery techniques adopt a static voter error checking schedule, which leads to delays in checking TMR components with high failure probability. In this paper we propose a *Voter Scheduling Engine* (VSE) for dynamically prioritizing and managing TMR voter checks so as to minimize the error detection time in the system and to thereby maximize the system's reliability. Software and hardware implementations of the VSE are proposed. Moreover, we have implemented the classic static voter checking schedule and the VSE on a real TMR system and evaluated the reliabilities of both approaches for varying radiation environments. Results demonstrate that the likelihood of system failure can be decreased by up to $50\%$ when the VSE, rather than static voter checking, is incorporated into the TMR system.**

## I. Introduction

Space missions are integrating increasingly many applications within a single SRAM-based *Field-Programmable Gate Array* (FPGA) to reduce mass, power consumption and to achieve desired processing performance. The considerable amount of configuration memory in these devices is susceptible to radiation-induced *Single Event Upsets* (SEUs). One of the most common approaches to dealing with this problem is to use *Triple Modular Redundancy* with repair, such as scrubbing [1], [2], or *Module-based Error Recovery* (MER) [3].

Both TMR-scrubbing and TMR-MER rely on *Dynamic Partial Reconfiguration* (DPR) to correct configuration memory errors. TMR-scrubbing is typically initiated periodically and commonly involves reading back each configuration memory frame, checking for errors, correcting any that are found, and writing back the corrected frame when necessary. TMR-MER, on the other hand, is commonly triggered when repeated errors are detected by the voter associated with a TMR component and involves rewriting the configuration memory for the entire module that has been found to be in error. While TMR-scrubbing occurs periodically, whether or not errors are present, TMR-MER relies upon the repeated detection of an error by the same TMR voter to trigger a reconfiguration of the module presenting the error [1]. Both methods utilize a controller to operate. However, MER also requires a *Reconfiguration Control Network* (RCN) to relay error requests from the voters in the system to the *Reconfiguration Controller* (RC) [4].

Agiakatsikas et al. provided a comprehensive study on TMR-MER using different RCNs [4]. They showed that TMR-MER has higher reliability than TMR-scrubbing if the RCN is triplicated and repaired when it suffers configuration memory errors. Morever, they also demonstrated that TMR-MER using an ICAP-based network to aggregate voter statuses provides the highest reliability in comparison with systems implementing soft star, bus or token ring network. The ICAP-based network utilizes the *Internal Configuration Access Port* (ICAP) available in advanced Xilinx devices to readback configuration frames that contain the health status of the system's TMR components [4], [5]. As with other soft networks, using the ICAP, the voter status bits cannot be readback in parallel. They must be read sequentially [6], typically in a round-robin order [4]. However, doing so increases the response time in checking the voters of highly vulnerable TMR components.

Intuitively, the rate at which a TMR component may requests error recovery depends on its failure rate. Therefore, the RCN arbiter should check components with high failure rates more frequently than those with lower failure rates in order to minimize the chance that the system becomes unusable. However, in [4], it was found that the failure rate of a component depends on the number of sensitive bits in its configuration (which affect the circuit operation when they are flipped), the recovery time of a TMR module and the amount of time that has elapsed since it was last checked, which is not constant. It is thus difficult and unlikely to be optimal to check components in a static order.

In this paper, we propose and evaluate a *Voter Scheduling Engine* (VSE) that dynamically prioritizes and manages the voter checks in a TMR-MER system using an ICAP-based network. Furthermore, we assess and compare the reliability of TMR-MER systems, in which the TMR voter statuses are checked in a round-robin fashion with those implementing our VSE. We show that TMR-MER systems that incorporate the VSE are overwhelmingly more reliable than those using a round-robin order. Results show that the failure probability of the TMR system incorporating VSE is up to 50% lower than that of the same system using round-robin during a simulated 30-day mission at *Geostationary Equatorial Orbit* (GEO).

## II. Scheduling voter checks

In this section, we detail a *Voter Scheduling Engine* (VSE) along with its implementations in either software or hardware that respectively require linear or logarithmic time to determine which TMR component to check next.

### A. Voter scheduling engine (VSE)

Scheduling is based on the idea that the likelihood of a configuration memory error being present in a component is

This research was supported in part by the Australian Research Council's Linkage (LP140100328) and Discovery (DP150103866) Projects funding schemes.

exponentially proportional to the product of the number of sensitive bits contained in the component and the amount of time elapsed since it was last checked. Therefore, components with a large number of sensitive bits must be checked more frequently than those with less sensitive bits. We propose the VSE to implement this idea.

The probability that the three modules of TMR component $i$ are still working after time $\tau$ is given by [7]:

$$P_i(\tau) = e^{-3\lambda M_i \tau},\ i = 1..N, \tag{1}$$

where $\lambda$ denotes the bit error rate, $M_i$ denotes the number of sensitive bits of a module of component $i$, and $N$ denotes the number of TMR components in a system.

For a constant bit error rate $\lambda$, $P_i(\tau)$ is the same over any fixed period $\tau$. In other words, in the arbitrary time window between two consecutive checks of component $i$, we can consider that $P_i(\tau)$ starts from 1 when the voter of component $i$ is checked and no error is detected ($\tau = 0$) and that $P_i(\tau)$ decreases until the next check. $P_i(\tau)$ is estimated by:

$$P_i(\tau) = e^{-3\lambda M_i \tau} = e^{-3\lambda M_i n_i \Delta t_o}, \tag{2}$$

where $n_i$ denotes the number of other components that have been checked since the last time component $i$ was checked and $\Delta t_o$ denotes the period between any two successive checks of any pair of components (assumed to be a constant).

In this paper, the VSE simply searches the list of $P_i(\tau)$ of all TMR components to obtain the component with the smallest $P_i(\tau)$, i.e. the component with the highest probability that at least one of the three TMR modules will fail during the next time window, and therefore should be checked next.

## B. Implementations of the proposed VSE

The VSE plays the role of the arbiter of the ICAP-based RCN. The VSE selects the component that needs to be checked next and asserts the component ID to the RC. While the RC reads the associated configuration frame and checks the status message of the voter associated with the component ID, it sends an $Update$ signal to the VSE, which commences the search for the component to check during the next cycle.

In the VSE implementation, we use the $M_i n_i$ product in terms of $\Delta t_o$ units as a proxy for representing $P_i(\tau)$ in (2). The $M_i n_i$ product ranks the records in the list of $N$ elements and eliminates the overhead of calculating the exponential function and multiplications. The algorithm involves two steps, which include (1) finding the largest $M_i n_i$ product and (2) updating the $M_i n_i$ product. These steps require $O(N)$ time in sequential software.

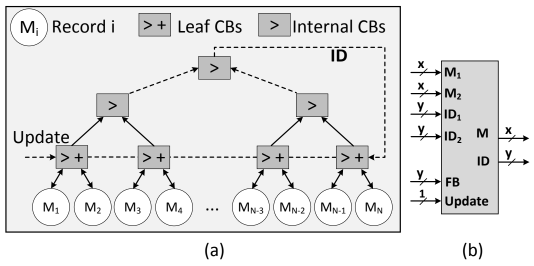

Fig. 1: (a) VSE and (b) Conditional Block interface.

With the proposed hardware implementation, as depicted in Fig. 1(a), $N-1$ *Conditional Blocks* (CBs) are connected together as a binary tree in which $N$ records are stored at the leaf nodes. Each record consists of the current $M_i n_i$ product and an identifier ($ID_i$) corresponding to component $i$. The $ID_i$ are assigned to components according to their $M_i$ values with the component with the largest $M_i$ being assigned 1. The CBs that are connected to the leaf nodes and which are referred to as the leaf CBs, perform two functions which involve (1) comparing the two records and forwarding the record with the greater $M_i n_i$ value to the node in the next level-up; and (2) updating the records at the leaf nodes. The internal CBs only perform the first function. If the two records contain the same $M_i n_i$ values, the component with the smaller $ID_i$ is chosen. The record with the greatest $M_i n_i$ product reaches the root CB after $log(N)$ steps. The root CB asserts the $ID$ of the selected component to the RC and to the leaf CBs. These CBs update their records after they receive an $Update$ signal from the RC. The record of the component that was just selected for checking is reset, while all other records are incremented by their corresponding $M_i$ amount.

The CB interface is depicted in Fig. 1(b) where the $x$-bit signals $M_1$, $M_2$ and $M$ denote the $M_i n_i$ products, and the $y$-bit signals $ID_1$, $ID_2$, and $ID$ represent the component IDs. $x$ depends on the size of the maximum $M_i n_i$ product while $y$ depends on the number of TMR components in a system.

Since the VSE scheduling of voter checks is based on the $M_i n_i$ product, the user can modify the $M_i$ values to impose user-defined priorities, if so desired. One limitation of the VSE is that when the number of TMR components is increased, the search time is increased logarithmically while the hardware overhead increases linearly as shown in Section III-B.

## III. Experimental Analysis

In this section we evaluate the VSE in terms of resource utilization, operating frequency, and scalability. We also evaluate and compare the reliability of a TMR system that checks voters in a round-robin fashion with one that uses the VSE to schedule voter checks. The systems were implemented on a Xilinx Artix-7 XC7A200TFBG484-1 FPGA using Vivado 2014.4 CAD tools.

## A. Experiments

We conducted two types of experiments. In the first type, the VSE is implemented as a standalone module on the Artix-7 FPGA in order to obtain resource utilization and performance results. We tested the VSE with systems containing 4, 8, 16 and 32 TMR components.

In the second set of experiments, we compared the reliability of a specific system using three different voter checking approaches as listed in Table I. In the first configuration, we checked the voters in round-robin order. In the second configuration, we used an off-chip hardware implementation of the VSE. And in the third and final configuration, we implemented the VSE on-chip as an additional triplicated system component.

TABLE I: Second Experiment configurations

| Configuration | # of TMR components | Voter Checking |
|---|---|---|
| I | 9 | Round robin |
| II | 9 | Off-chip VSE |
| III | 10 (including the VSE) | On-chip VSE |

To implement these approaches, three system configurations comprising the first 9 TMR components listed in Table III were created — note that the VSE was only included as a TMR component in the third configuration. The TMR components include a single MAC-based 21-tap *Finite Impulse*

TABLE II: Area and performance of the VSE

| $x - y$ | N | Slices | LUTs | FFs | Freq (MHz) |
|---|---|---|---|---|---|
| 32 – 5 | 32 | 894 (2.53%) | 1922 (1.43%) | 1527 (0.57%) | 104 |
| 32 – 4 | 16 | 579 (1.73%) | 1595 (1.19%) | 968 (0.36%) | 117 |
| 32 – 3 | 8 | 282 (0.84%) | 749 (0.55%) | 454 (0.16%) | 141 |
| 32 – 2 | 4 | 114 (0.34%) | 305 (0.22%) | 196 (0.07%) | 191 |
| 16 – 5 | 32 | 419 (1.25%) | 1032 (0.77%) | 775 (0.28%) | 122 |
| 16 – 4 | 16 | 314 (0.93%) | 832 (0.62%) | 488 (0.18%) | 164 |
| 16 – 3 | 8 | 146 (0.43%) | 380 (0.28%) | 230 (0.08%) | 197 |
| 16 – 2 | 4 | 58 (0.16%) | 154 (0.11%) | 100 (0.03%) | 270 |
| 8 – 5 | 32 | 232 (0.69%) | 583 (0.43%) | 399 (0.14%) | 130 |
| 8 – 4 | 16 | 158 (0.47%) | 426 (0.31%) | 248 (0.09%) | 174 |
| 8 – 3 | 8 | 77 (0.23%) | 298 (0.14%) | 118 (0.04%) | 220 |
| 8 – 2 | 4 | 31 (0.09%) | 79 (0.05%) | 52 (0.01%) | 271 |

*Response* (FIR) filter with 16-bit signal width, an 8-to-3-bit *Block Adaptive Quantizer* (BAQ), an 8,096-word deep 32-bit *FIFO*, three 32-bit *Shift Registers* (SRs) having different lengths and a variety of combinational functions between the stages and three 32-bit *Binary Search Trees* (BSTs) of different heights and a variety of combinational functions at each node. The VSE used in the second and third configurations was configured with $x = 10$ and $y = 4$. A MicroBlaze (MB) processor was used to implement the RC and the AXI HWICAP IP was used to reconfigure faulty modules in all of the three configurations.

We also injected faults into each TMR component in order to verify the operations of the MicroBlaze processor and to measure the correction time ($t_c$) for each TMR module. The MB controls the fault-injection procedure which involves reading the corresponding frame, flipping one bit, and writing the frame back using the HWICAP. After injecting a fault, the MB commences checking the voter statuses of all TMR components and reconfiguring any faulty module as needed. The correction time accounts for the time interval between the error being detected until the last word of the partial bitstream is written to the HWICAP.

## *B. Results*

*1) VSE component:* Table II presents the resource utilization and maximum clock frequency of the VSE with varying number of TMR components and the sizes $x$ and $y$ of $M_i$ and $ID_i$. When the number of TMR components increases, the resources needed to implement the VSE increase almost linearly. The VSE utilises a small amount of resources and can handle a design with 32 components in which $x = 32$ and $y = 5$. This small footprint ensures that the placement and routing of other components is not affected and that the VSE can also be readily implemented as a TMR protected component with a small overhead.

So as to reduce the wire requirements of the VSE search tree, we have normalized the $M_i$ value to the smallest $M_i$. This mean that the $M_i n_i$ product does not overflow an $8-bit$ or a $16-bit$ value. In this case, the CBs with $x = 8$ or $x = 16$ can be used to implement the VSE. As can be observed from Table II, the resource utilization of the VSE with $x = 8$ and of the VSE with $x = 16$ is reduced by approximately 75% and 50%, respectively, while the maximum clock frequencies are substantially higher in comparison with the VSE with $x = 32$.

Moreover, the VSE can update and find which component to check next in $O(log\ N)$ clock cycles. Therefore, even if the ICAP is used at its maximum throughput, we can maintain a list of over $2^{100}$ components in the time it takes to check one voter. This is because it requires at least 101 clock cycles to read the voter status bits in Xilinx 7-series FPGAs [6].

TABLE III: Results of mapping 10 TMR components to Xilinx Artix-7 XC7A200TFBG-484 FPGA

| Design | Essential Bits $\mathbf{M_i}$ | Norm $\mathbf{M_i}$ (relative to FIR) | $\mathbf{n_f}$ | $\mathbf{t_c}$ (ms) MBlaze | $\mathbf{t_d}$ ($\Delta t_o$) I | II | III |
|---|---|---|---|---|---|---|---|
| **BST3** | 1,833,235 | 152 | 1,483 | 26.7 | 4.5 | 1.7 | 1.7 |
| **SR3** | 1,403,647 | 117 | 1,090 | 19.6 | 4.5 | 1.7 | 1.7 |
| **BST2** | 793,534 | 66 | 610 | 11.0 | 4.5 | 3.3 | 3.4 |
| **SR2** | 515,904 | 43 | 474 | 8.5 | 4.5 | 4.8 | 4.8 |
| **SR1** | 285,914 | 24 | 378 | 6.8 | 4.5 | 8.2 | 8.4 |
| **BST1** | 281,604 | 23 | 145 | 2.6 | 4.5 | 8.2 | 8.4 |
| **BAQ** | 48,963 | 4 | 73 | 1.3 | 4.5 | 41.0 | 41.3 |
| **FIFO** | 41,842 | 3 | 192 | 3.5 | 4.5 | 56.0 | 54.7 |
| **FIR** | 12,042 | 1 | 65 | 1.2 | 4.5 | 164.4 | 157.4 |
| **VSE** | 51,293 | 4 | 144 | 2.6 | – | – | 41.3 |

*2) Case study results:* Table III presents the number of essential bits ($M_i$) (as a worst case estimate of the sensitive bits), the normalized $M_i$ (ratio of each component's $M_i$ relative to the component with smallest $M_i$), the number of frames ($n_f$), the correction time ($t_c$) and the average detection time ($t_d$) in $\Delta t_o$ units of each TMR component for the three configurations studied. In our design, the MB-based RC and off-chip flash configuration storage support a sustained frame write period of 18 us per frame which was used to calculate the $t_c$ of each component in Table III. These data were used as inputs to a Markov reliability model derived from [4], [8]. The system failure probability ($FP_s^{TMR}(t)$) is the complement of the system reliability. In this paper, we derived a "high radiation level" of bit failure rates as shown in Table IV for Xilinx 7-series FPGAs [9] at different orbits using the CREME96 model [10] assuming 2.54 mm aluminium shielding.

TABLE IV: Bit failure rates in different orbits [11]

| Orbit | Altitude (km)/ Inclination | Solar Min | Worst Week | Worst Day | Peak 5-Min |
|---|---|---|---|---|---|
| | | $\lambda$ (SEUs/Bit/s) | | | |
| GEO | 35,768/0$^o$ | 1.71E-13 | 2.16E-11 | 7.34E-11 | 2.66E-10 |
| GPS | 20,200/55$^o$ | 1.54E-13 | 1.43E-11 | 4.84E-11 | 1.75E-10 |
| LEO | 2,000/51.6$^o$ | 1.92E-14 | 7.01E-13 | 2.33E-12 | 8.41E-12 |

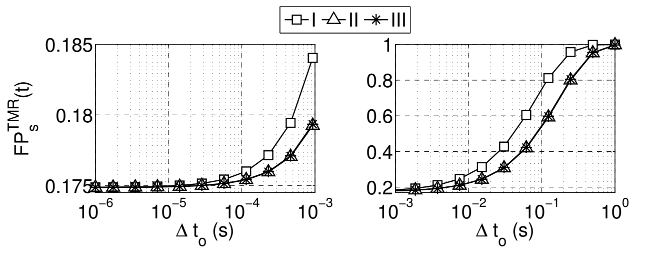

Fig. 2: Failure probabilities of the three configurations in GEO orbit at the peak 5-min condition during a mission of 30 days.

Fig. 2 plots the failure probabilities of the three configurations during a 30-day mission in GEO orbit at the peak 5-min radiation condition for each $\Delta t_o$. For clarity, we plotted 2 sub-figures in which $\Delta t_o$ ranges from 1 us to 1 ms and from 1 ms to 1 s. As can be seen in Fig. 2, configuration II is generally less vulnerable than configuration I. When $\Delta t_o$ is increased, the gap between the failure probabilities of the two configurations increase, but eventually it decreases because of the proximity of the failure probabilities of the two configurations to 1. Moreover, the reliabilities of configurations II and III are similar as errors occur infrequently in the relatively small VSE component.

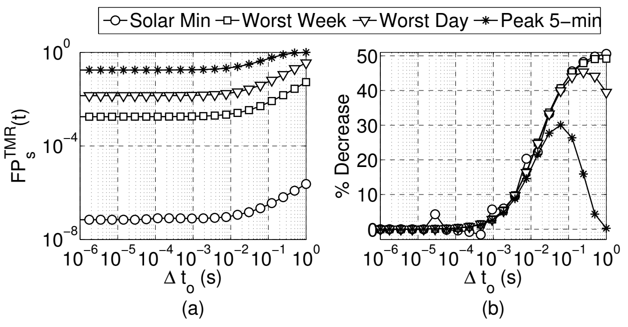

Fig. 3: (a): Failure probabilities of configuration III with four radiation conditions in GEO orbit during a mission of 30 days, (b) Percentage decrease in the probability of failure in configuration III versus configuration I.

TABLE V: Failure probabilities of the three configurations when $\Delta t_o$ = 10 ms

| **Orbit\Configuration** | | **I** | **II** | **III** |
|---|---|---|---|---|
| **GEO** | Solar min | $1.23E-7$ | $1.00E-7$ | $1.03E-7$ |
| | Worst week | $2.89E-3$ | $2.31E-3$ | $2.32E-3$ |
| | Worst day | $2.28E-2$ | $1.83E-2$ | $1.83E-2$ |
| | Peak 5-min | $2.68E-1$ | $2.21E-1$ | $2.22E-1$ |
| **GPS** | Solar min | $9.98E-8$ | $7.79E-8$ | $7.81E-8$ |
| | Worst week | $8.75E-4$ | $6.99E-4$ | $7.02E-4$ |
| | Worst day | $9.98E-3$ | $7.98E-3$ | $8.02E-3$ |
| | Peak 5-min | $1.23E-1$ | $0.99E-1$ | $0.99E-1$ |
| **LEO** | Solar min | Negligible | | |
| | Worst week | $2.10E-6$ | $1.68E-6$ | $1.69E-6$ |
| | Worst day | $2.33E-5$ | $1.86E-5$ | $1.87E-5$ |
| | Peak 5-min | $3.03E-3$ | $2.42E-3$ | $2.43E-3$ |

Fig. 3(a) presents the failure probabilities of configuration III during a 30-day mission in GEO orbit conditions for various $\Delta t_o$. It can be observed that similar system failure probabilities can be achieved at different conditions by adjusting $\Delta t_o$. This implies that depending upon the environment that the system is operating in, we can vary the operating frequency of the voter checks to save energy, while maintaining a desired reliability.

Fig. 3(b) depicts the percentage decrease in the failure probability of configuration III relative to that of configuration I during a 30-day mission in GEO orbit conditions for various $\Delta t_o$. It can be seen that the percentage decrease is similar at different GEO orbit conditions when $\Delta t_o$ is less than 0.01 second. For the peak 5-minute condition, the highest percentage decrease is 30%, while it is 45% for the worst day and 50% for the worst week and solar min conditions. However, when $\Delta t_o$ is increased, the percentage decreases in all conditions eventually decline to $0\%$ because the failure probabilities of both configurations approach 1. Please note that due to space limitations and for clarity, this paper has only plotted results for a 30-day mission in GEO orbit. Similar results can also be obtained for longer missions in the three orbits. For example, the percentage decrease is up to 50% for four radiation conditions for a 10-year mission in LEO orbit, while it is also up to 50% for solar min and worst week, 40% for worst day and 10% for peak 5-min conditions for a 1-year mission in GPS orbit.

From Figs. 2 and 3, one may need to examine the duration of a mission and the voter checking frequency in order to evaluate the robustness of TMR systems that employ either voter checking approach. For the sake of energy saving in space-based applications in long-term missions, the checking frequencies may be decreased [2]. In this case, the VSE is capable of ensuring a higher reliability than round robin.

Table V shows the failure probabilities of configurations I, II, and III during a 30-day mission in GEO, GPS and LEO orbits. For brevity, we present the failure probabilities of a TMR system with MER when $\Delta t_o$ = 10 ms. At these operating points, we confirm that configurations II and III are generally more reliable than configuration I and that they have similar reliability. Furthermore, the failure probabilities of the three configurations operating in LEO orbit are two to three orders of magnitude less than those in GPS and GEO orbits.

## IV. Concluding Remarks and Future Work

In this paper, we have proposed the use of a *Voter Scheduling Engine* (VSE) to schedule the voter checks in an FPGA-based TMR system with MER that employs an ICAP-based RCN. The VSE plays the role of an arbiter in prioritizing checks of critical components. We have also argued that any system using an RCN that provides random access to component voters can benefit from using a VSE to prioritize checks of more vulnerable components. We also presented practical algorithms for checking voters in both hardware and software to maximize the system reliability.

We have shown that a TMR system that includes a VSE is generally more reliable than one that checks voters in a round-robin fashion. The failure probabilities of the former are up to $50\%$ lower than those of the latter for a 30-day mission in GEO orbit with solar min and worst week conditions, and up to $45\%$ and $30\%$ lower with worst day and peak 5-min conditions, respectively.

The proposed approach is fair since each voter will eventually be checked. By modifying the record of the number of essential bits, we can impose a user-defined priority with which to check the voters. The cost of our method is the design and hardware cost of the moderately complex VSE. Our future work consider the static scheduling of voter checks to optimize the system reliability and the reconfiguration order when multiple errors occur in different modules simultaneously.

## References

[1] C. Carmichael et al., "Correcting single event upsets through virtex partial configuration," *Xilinx xapp216 (v1.0).*, 2000.

[2] I. Herrera-Alzu et al., "Design techniques for Xilinx Virtex FPGA configuration memory scrubbers," *IEEE Trans. on Nuclear Science*, vol. 60, no. 1, pp. 376–385, Feb 2013.

[3] C. Bolchini et al., "A novel design methodology for implementing reliability-aware systems on SRAM-based FPGAs," *IEEE Trans. on Computers*, vol. 60, no. 12, pp. 1744–1758, 2011.

[4] D. Agiakatsikas et al., "Reconfiguration Control Networks for TMR Systems with Module-based Recovery," in *FCCM*, 2016, pp. 88–91.

[5] F. Veljkovic et al., "Adaptive reconfigurable voting for enhanced reliability in medium-grained fault tolerant architectures," in *AHS*, June 2015, pp. 1–8.

[6] *UG470: Xilinx 7 Series FPGAs configuration user guide v1.10, 2015.*

[7] M. L. Shooman, *Reliability of Computer Systems and Networks: Fault Tolerance, Analysis and Design.* New York, NY, USA: John Wiley & Sons, Inc., 2002.

[8] D. McMurtrey et al., "Estimating TMR reliability on FPGAs using Markov models," *All Faculty Publications. Paper 149.*, 2008. [Online]. Available: http://scholarsarchive.byu.edu/facpub/149

[9] D. Hiemstra et al., "Single event upset characterization of the Kintex-7 Field Programmable Gate Array using proton irradiation," in *REDW*, July 2014, pp. 1–4.

[10] A. Tylka et al., "CREME96: A revision of the cosmic ray effects on micro-electronics code," *IEEE Trans. on Nuclear Science*, vol. 44, no. 6, pp. 2150–2160, Dec 1997.

[11] D. Heynderickx et al., "ESAs SPace ENVironment Information System (SPENVIS): a WWW interface to models of the space environment and its effects," *in AIAA*, vol. 371, 2000.