Scheduling Considerations for Voter Checking in FPGA-based TMR Systems

Nguyen T.H. Nguyen[1] Ediz Cetin[2]
Oliver Diessel[1]

[1] University of New South Wales, Australia
{h.nguyentran,o.diessel}@unsw.edu.au
[2] Macquarie University, Australia
ediz.cetin@mq.edu.au

Technical Report
UNSW-CSE-TR-201705
March 2017

UNSW
SYDNEY

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Field-Programmable Gate Arrays (FPGAs) are susceptible to radiation-induced *Single Event Upsets* (SEUs). A common technique for dealing with SEUs is *Triple Modular Redundancy* (TMR) combined with *Module-based configuration memory Error Recovery* (MER). By triplicating components and voting on their outputs, TMR helps localize the configuration memory errors, and by reconfiguring the faulty component, MER swiftly corrects the errors. However, the order in which the voters of TMR components are checked has an inevitable impact on the overall system reliability. In this paper, we outline an approach for computing the reliability of TMR-MER systems that consist of finitely many components. Using the derived reliability models we demonstrate that the system reliability is improved when the critical components are checked more frequently for the presence of configuration memory errors than when they are checked in round-robin order. We propose a genetic algorithm for finding a voter checking schedule that maximizes system reliability for systems consisting of finitely many TMR components. Simulation results indicate that the *mean time to failure* of TMR-MER systems can be increased by up to 100% when *Variable-Rate Voter Checking* (VRVC) rather than round robin, is used. We show that the power used to eliminate configuration memory errors in an exemplar TMR-MER system employing VRVC is reduced while system reliability remains high. We also demonstrate that errors can be detected 30% faster on average when the system employs VRVC instead of round robin for voter checking.

1 Introduction

Space missions increasingly integrate many applications within a single SRAM-based *Field-Programmable Gate Array* (FPGA) to reduce mass, power consumption and to achieve high performance. The considerable amount of configuration memory in these devices makes them susceptible to radiation-induced *Single Event Upsets* (SEUs). Alleviating this vulnerability is of paramount importance for the widespread use of SRAM-based FPGAs in space missions.

One approach to dealing with SEUs is to use *Triple Modular Redundancy* (TMR) with *Module-based Error Recovery* (MER) [1, 2]. TMR-MER relies on *Dynamic Partial Reconfiguration* (DPR) to correct configuration memory errors. This approach is commonly triggered when repeated errors are detected by the voter(s) associated with a TMR component and involves rewriting the configuration memory of the module that has been found to be in error. However, the order in which the voters of TMR components are checked has an inevitable impact on the overall system reliability.

To improve the reliability of FPGA-based TMR-MER systems, the authors in [3] proposed dynamically adjusting the order in which TMR voters are checked for module errors rather than checking them sequentially. The approach was implemented based on the idea that the more vulnerable components (e.g., those comprising a greater number of essential bits [4] and longer recovery times) are checked more frequently than less vulnerable ones. A question that work raised, and which we here answer in the affirmative, is whether a static voter checking schedule can be found to maximize system reliability.

It has also been noted that while TMR-MER is generally effective for mitigating SEUs affecting the configuration memory [5], it is not well suited for protecting systems against multiple coincident SEUs that affect multiple modules of a TMR component, thereby defeating the protection afforded by redundancy. In this work, we investigate the reliability of TMR-MER systems consisting of an arbitrary number of triplicated components operating in harsh radiation environments, such as in geosynchronous orbit during solar flares and in high-energy physics laboratories like the Large Hadron Collider located at CERN, where multiple coincident SEUs are more probable [6]. Our main interest is in determining the impact on overall system reliability of varying the order and rate at which the voters of TMR components are checked for errors.

Our contributions are:

- To derive reliability models of TMR-MER systems that comprise finitely many TMR components whose voters are checked in round-robin order and at a variable rate. We refer to such a schedule as *Variable-Rate Voter Checking* (VRVC). Previous work has primarily focused on the effects of SEUs on SRAM FPGA-based systems while our analysis considers the impact of multiple consecutive events, which is an important consideration in providing a more accurate analysis of the system reliability.

- To propose a *genetic algorithm* (GA) for finding the optimal rate at which to check all components so as to maximize the *Mean Time To Failure* (MTTF) and the reliability of TMR-MER systems.

- To show that power consumed checking for errors can be reduced by reducing the checking frequency. In this case, VRVC is capable of ensuring

a higher system reliability than round robin.

- To demonstrate that the *Mean Time To Detect* (MTTD) errors is reduced by 30% on average when VRVC is used instead of round robin.

This paper is organized as follows: Section 2 briefly provides background on the Xilinx FPGAs we use for hosting our applications and related work on TMR-MER systems. Section 3 presents reliability models for TMR-MER systems that consist of finitely many components whose voters are checked in round-robin order or at a variable rate. Section 4 describes a GA to derive a voter checking schedule that maximizes the system reliability. Section 5 describes our experimental method, reports on our findings and discusses the results. Concluding remarks and directions for further study are given in Section 6.

2 Background on FPGA and Related Work

Memory elements in an SRAM-based FPGA device can be classified into two groups: configuration and user memory bits. The configuration memory bits are used to specify the particular circuit mapped into the FPGA, whereas the user memory bits, such as flip-flops or block RAMs, hold the current state of the circuit. Unless the user design is dynamically reconfigurable, the contents of the configuration memory bits should remain unchanged, while the contents of the user memory bits may change on any clock cycle. Xilinx FPGA configuration memory is organized into frames, which are the smallest addressable segment of the configuration memory space. The frame size varies among FPGA families; in the case of Xilinx 7–Series FPGAs, it consists of 101 32-bit words. Moreover, the configuration memory bits account for the largest proportion of all the memory cells in SRAM-based FPGAs e.g., more than 80% in the latest Xilinx FPGA (UltraScale XCVU440). Therefore, there is a far greater probability of SEUs occurring in configuration memory bits than in user memory bits. Since the configuration memory upsets have the potential to alter the function of a *look up table* (LUT) or the routing between nodes, they can lead to "permanent" errors manifesting in user circuits until the altered configuration state is corrected. In this work, we study the impact on reliability of multiple SEUs that affect the configuration memory bits in TMR-MER systems.

The design of TMR-MER systems to recover from configuration memory errors in SRAM-based FPGAs has been described in a number of articles [1, 3, 5, 7, 8]. These systems utilize a *Reconfiguration Control Network* (RCN) such as a star-based [1, 7], a bus-based [5], or an ICAP-based network [8] to convey the status of the individual TMR component voters to a *Reconfiguration Controller* (RC), which determines whether configuration memory errors are present. To determine whether any configuration memory upsets have occurred, most TMR-MER systems check the voters of the TMR components in round-robin order. Our work aims to enhance the system's error detection capabilities and thereby raise overall system reliability by checking the TMR component voters for module errors at different rates.

Reliability models for TMR-MER systems have not yet been studied in detail. When they are mentioned, Markov models are used to compute the system reliability with the assumption that the recovery of modules of multiple TMR components occurs independently [5]. While acceptable at low error rates,

the problem with this assumption at high error rates is that the methods for correcting configuration memory errors are inherently sequential, hence the models do not consider the effect of configuration memory errors on other TMR components while a faulty module is being reconfigured.

3 Reliability Model

In this section, we introduce models that estimate the reliability of TMR-MER systems. These models are then used to estimate the reliability of FPGA-based designs in harsh radiation environments when multiple coincident upsets are more probable[1]. We describe a general reliability model that has been widely used to estimate the reliability of FPGA-based systems. Based on this general model, we outline a procedure for estimating the reliability of TMR-MER systems that consist of an arbitrary number of TMR components and whose voters are checked in either round-robin order or at a variable rate.

3.1 General Reliability Model

The reliability of a TMR component k over time Δt, $R_k(\Delta t)$, can be expressed w.r.t. the failure probability of the component, $FP_k(\Delta t)$, which is the sum of the individual likelihoods that the component fails for all u SEUs that may affect the device during Δt. These relationships are given in [6] as:

$$\begin{aligned} R_k(\Delta t) &= 1 - FP_k(\Delta t), \\ FP_k(\Delta t) &= \sum_{u=1}^{\infty} P(F_k|E_u)P(E_u, \Delta t), \end{aligned} \tag{3.1}$$

where event F_k is the failure of component k during the period of time Δt and event E_u is that u SEUs have occurred in the device during the period of time Δt. Failure of TMR component k means that at least two of the three modules suffer from errors and that the component's voter therefore fails to produce the correct output.

$P(F_k|E_u)$ can be estimated for various values of u using the number of sensitive bits per component, for which we use the number of essential bits reported by the vendor's tools as a worst case estimate. Sensitive bits are those bits that cause a functional error if they change state, while essential bits are those bits associated with the circuitry of the design [4].

$P(E_u, \Delta t)$, the probability of event E_u occurring during Δt, can be modelled with a Poisson distribution, $P(E_u, \Delta t) = e^{-\nu}\frac{\nu^u}{u!}$, where ν is the expected number of SEUs suffered by the device during a period of time Δt and is obtained from the product of the failure rate of one configuration memory bit of a device (λ_{bit}), the number of configuration memory bits of a device (n_c) and the time period (Δt): $\nu = \lambda_{bit} \times n_c \times \Delta t$. λ_{bit} depends upon the radiation level, the IC process technology and the circuit architecture of the FPGA fabric.

[1] Please note that the model presented does not take into account *Multiple-bit upsets* (MBUs) i.e., more than one upset in a configuration word or frame from a single charged particle [9]. We plan to consider these in future work.

Once the failure probability of component k is known, the failure rate λ_k of component k is given by [6]:

$$\lambda_k = \frac{FP_k(\Delta t)}{\Delta t}. \tag{3.2}$$

Since a TMR component can fail in different scenarios (see Fig. 3.1 and associated discussion in Section 3.2) with different failure rates (λ_k^i), it is more meaningful to compute the composite failure rate of each component (λ_k^c). This parameter can be calculated for the expected proportions (ρ_k^i) in which each scenario occurs:

$$\lambda_k^c = \sum_{i=1} \rho_k^i \lambda_k^i. \tag{3.3}$$

where $\sum \rho_k^i = 1$.

Typically, a system contains N interdependent TMR components connected in series such that the failure of any one TMR component causes the system to fail. The failure rate of a series TMR system, λ_s, is the sum of all component failure rates [10]. Furthermore, the MTTF of the series TMR system is given by the reciprocal of the system failure rate. The system reliability is calculated as follows:

$$R_s(t) = e^{-\lambda_s . t}. \tag{3.4}$$

In this paper, we do not consider the impact of non-redundant modules such as the RCN, the RC and the voters on system reliability as these have the same impact on all of the various system settings analysed in the following sections.

3.2 Failure Rates of TMR-MER Systems in which Voters are Checked in Round-robin Order

Based on the general reliability model described in Section 3.1, we estimate the failure rate of systems comprised of two TMR components connected in series. Hereafter, we say that if the output of one module of a TMR component repeatedly differs from that of the other two, that the component is suffering from an "error", and if, after the component suffers another one or more SEUs, the outputs of the remaining two modules repeatedly differ, that the component has "failed". We also assume that once a faulty module is detected, it is dynamically reconfigured to correct the error [5].

In a two-component system, a component may fail in one of four different ways that are classified into two groups as shown in Fig. 3.1 using the notation listed in Table 3.1. Note that Fig. 3.1 only describes the modes in which C1 can fail; the modes in which C2 can fail can be derived in a similar manner.

Group 0: No other component suffers an error

– Case 1 (Fig. 3.1(1)): C1 suffers from two or more SEUs that cause it to fail during the period of time between two consecutive checks of its voters (e.g., during Δt_1 – the period of time between O_{12} and O_{13}).

– Case 2 (Fig. 3.1(2)): C1 suffers an error from one or more SEUs during the period of time between two consecutive checks of its voters (between O_{12} and O_{13} in Fig. 3.1(2)). Thereafter, C1 fails if one or more SEUs affect its remaining working modules during the period of time that it is recovering from the previous error (e.g., during Δt_{r1} – from time O_{13} to the end of the recovery process of C1).

Table 3.1: Notation

Symbol	Definition
N	Number of TMR components in the system
Ck	Component $k, k = 1..N$
O_{kn}	Ck is observed for the n^{th} time by checking its voter(s)
Δt_o	The time period between successive voter observations (assumed to be constant for a given system setting)
Δt_{dk}	The time period between two consecutive observations of Ck
Δt_{rk}	The time period to recover a faulty module of Ck
Δt_k	The total time period over which Ck can fail
Δt_{dij}	The time period between successive observations of Ci and Cj
$\Delta t_{d'ij}$	The average time period between two consecutive observations of Ci in the interval between two consecutive observations of Cj

Group 1: One other component suffers an error

– Case 1 (Fig. 3.1(3)): C1 suffers from two or more SEUs that cause C1 to fail during a period of time between two consecutive checks of its voters that is longer than usual because the system is recovering from an error in C2. C1 fails during the period of time that commences after it is observed to be without an error (at O_{12}), continues while C2 is checked and recovered, and finishes when C1 is observed again at O_{13}.

– Case 2 (Fig. 3.1(4)): C1 suffers an error from one or more SEUs during the period of time between two consecutive checks of it (between O_{12} and O_{13}) while the system is recovering from an error in C2. C1 then fails if one or more SEUs affect a second and/or third module of C1 while it is recovering from the previous error.

To summarize, in case 1 of either group, component k fails, i.e., suffers multiple errors to its different modules, between successive voter checks. In case 2, on the other hand, component k suffers an error to one of its modules during this period, and then fails following subsequent upsets to its other modules while recovering from the first error.

The failure probability of component k in case 1 of either group is computed based on $FP_k(\Delta t)$ in Eq. (3.1) with corresponding Δt_k as shown in Figs. 3.1(1) and 3.1(3). Component k fails only if, having suffered an error due to an SEU, subsequent SEU(s) affect one of the remaining two functioning modules during Δt_k. If only one SEU occurs during Δt_k, the failure probability of component k in case 1 is zero because one SEU cannot cause a malfunction of a TMR component. Empirically, this failure probability is not exactly zero because there are a few single configuration bits that do indeed cause the TMR design to fail [11]. However, applying the techniques that are described in [11] removes such bits from the TMR design.

The failure probability of component k in case 2 of either group is the product of the probability that event M_k (i.e., that component k suffers an error) occurs during the period of time Δt_{dk} as shown in Figs. 3.1(2) and 3.1(4) and that component k fails during the period of time Δt_{rk} given the occurrence of event

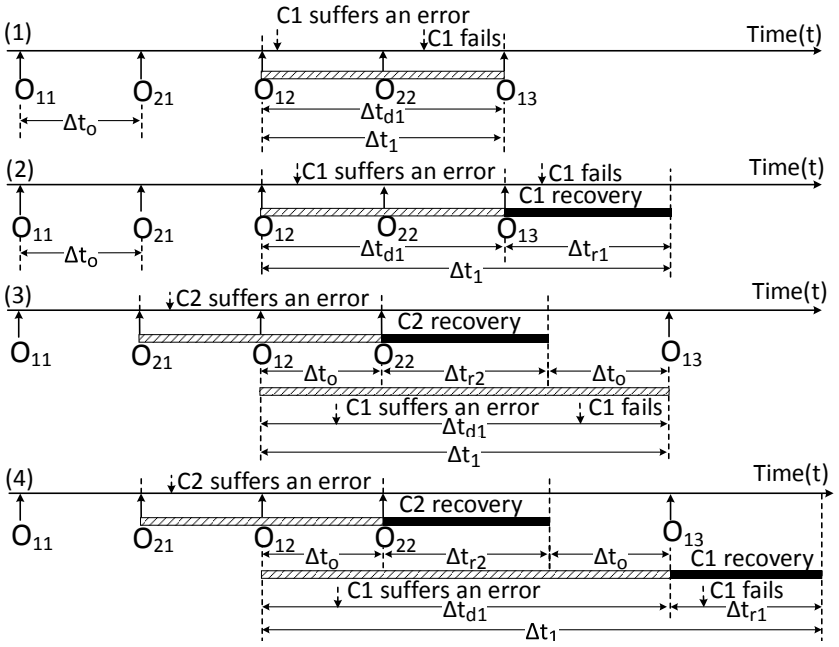

Figure 3.1: Failures of component 1 in two-component systems in which the voters are checked in round-robin order.

M_k. Event M_k occurs in component k only if at least one SEU affects one of the three modules of the TMR component and none of the other SEUs affect the two remaining working modules during Δt_k. Then the component k fails only if at least one SEU affects one of the other two remaining working modules during Δt_{rk}.

Based on Eq. (3.2), the failure rate of component k (λ_k^i) in each case is estimated using the corresponding Δt_k (Fig. 3.1).

The proportions ρ_k^i are calculated for the likelihood by which component k fails in each case. For example, the likelihood of cases in group 0 occurring depends upon the likelihood that component k suffers an error, while that of cases of group 1 depends upon the likelihood that both components suffer an error. It is obvious that the cases of group 0 are more likely to occur than those of group 1.

The composite failure rate of component k (λ_k^c) is calculated by substituting λ_k^i and ρ_k^i into Eq. (3.3). The system failure rate can be computed by summing all λ_k^c.

The reliability of systems comprising any number of TMR components can readily be computed by extending the approach we have outlined for two-component systems. This involves considering all possible cases in which each component may fail. In an N-component system, there are $\sum_{g=0}^{N-1} \sum_{s=1}^{\binom{N-1}{g}} \sum_{i=1}^{2}$ cases in which a component may fail in one of N groups; each group g, where $g = 0..N–1$, consists of $\binom{N-1}{g}$ situations in which g other components suffer an error first; and each situation involves two cases as summarized above. The likelihood of each case occurring depends upon the likelihood of all involved

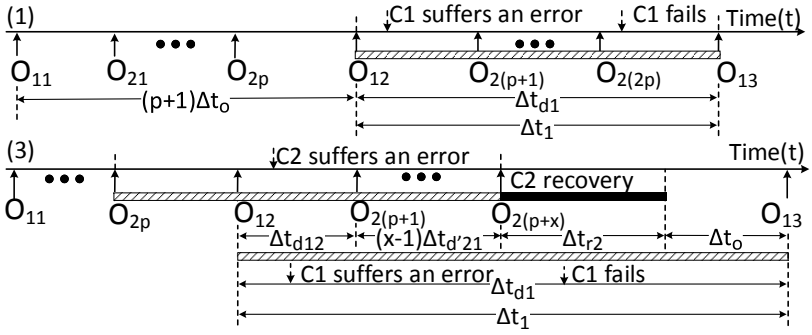

Figure 3.2: Failure of component 1 in systems employing variable-rate voter checking

components suffering an error in each case.

3.3 Failure Rates of TMR-MER Systems employing VRVC

Variable-Rate Voter Checking (VRVC) is defined as a periodic schedule in which component voters are checked at specific times and in which the more vulnerable components' voters are checked more frequently than those of the less vulnerable ones. For example in a system of 4 components, one period of a schedule could be *4-3-4-2-4-3-4-2-3-1* in which each digit represents the component whose voters are to be checked. In this case, component 4 is deemed more vulnerable and hence checked more frequently when compared to the other components, and component 1 is deemed least vulnerable and hence checked less frequently. In this sub-section, we first consider a system that consists of two components and then generalise the model to obtain the reliability of systems that consist of any number of TMR components.

A 2-component system

Similar to the cases described in Section 3.2, we observe that C1 fails in one of four different ways as partly depicted in Fig. 3.2 using the notation of Table 3.1. Note that p in Fig. 3.2 denotes the nominal number of times that C2 is checked between two consecutive checks of C1 due to its greater susceptibility to SEUs than C1's. In case 1 of group 1 (Fig. 3.2(3)), we assume that the system detects an error in C2 x checks after C1 is checked (at O_{12}) where x varies from 1 to p. In this work, we associate with $x = 1..p$ the number of checks that the system performs before it detects an error in C2. Thus, each case of group 1 involves p sub-cases that have the same likelihood of both components suffering an error (ρ_k^i). For example, given a schedule of two components in the following order *1-2-2-2-2-1-2-2-2-2-1...* where each digit denotes the observation of the corresponding component, Δt_{d1} and Δt_{d2} in group 0 are $5\Delta t_o$ and $1.25\Delta t_o$, respectively. Both Δt_{d12} and $\Delta t_{d'21}$ in group 1 are Δt_o. Furthermore, with such a schedule, there are four checks of C2 during the period of time between two consecutive checks of C1. Thus, $x = 1..4$.

The observations of C2 differ slightly from those of C1. C2 may also fail

in one of four different ways, but the number of sub-cases in group 1 is only 1. This is because between any two consecutive checks of C2, C1 is checked at most once.

The above observations allow us to compute the system failure rate.

An N-component system

Without loss of generality, we assume that the components are numbered and ranked $k, k = 1..N$ into non-decreasing vulnerability order, and that component k is therefore checked more frequently than component k–1. After the reconfiguration of a faulty module is finished, the system checks all other components in descending order of vulnerability before recommencing the planned schedule. For example, in a system of 4 components, after component 2 is recovered, components 4, 3 and 1 need to be checked in that order before resuming the variable-rate schedule. If multiple errors occur in a sequence, the system checks all other components that have not been reconfigured. In doing so, the system is fair in reconfiguring all components of the system if an error is detected.

The system failure rate can be computed by considering all possible cases in which each component may fail. A component k may also fail in different groups *g, g = 0..N–1* in which g other components suffer an error first. Each group involves ${}^{k-1}P_1 {}^{N-k}P_{g-1} + {}^{N-k}P_g$ situations (${}^nP_k$ denotes k-permutations of n). This is because group g includes ${}^{k-1}P_1 {}^{N-k}P_{g-1}$ situations in which the first component to suffer an error, f, is such that the rank of $f < k$ and ${}^{N-k}P_g$ situations in which the rank of $f > k$. Each situation involves two cases as summarized in Section 3.2 and each case involves a number of sub-cases that are schedule-dependent.

Most of the timing parameters for computing the composite failure rate of one component (as partly shown in Fig. 3.2) differ from those of the others and are not constant. Fortunately, we can statistically determine their average values based on a real schedule as described in Section 4

4 Scheduling Voter Checks

We surmise the problem of statically *determining the optimal number of voter checks per period* in an N-component system is NP-hard. We therefore propose a *genetic algorithm* (GA), which is a probabilistic search method based on an evolutionary approach, to heuristically determine the rate at which all triplicated components in a system should be checked so as to maximize the system reliability. Once the rate at which components should be checked has been determined, we use a second GA, as detailed in [12], to *generate a schedule in which the determined number of voter checks are distributed as evenly as possible* per schedule period. The schedule produced by the second GA is needed to evaluate the fitness of individual solutions to the first GA, which determines the number of checks to be performed. The second GA is therefore nested within the first GA's evaluation function.

4.1 Proposed genetic algorithm

A typical GA requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. Possible solutions (individuals)

are represented by a data structure called a chromosome. A chromosome is composed of simple elements called genes. An initial population of possible solutions is randomly created. As long as the stopping condition (e.g., exceeding a given number of generations) has not been met, a new generation is created. This involves computing the fitness value of each individual in the population. Individuals that represent desirable solutions (e.g., high fitness values or small system failure rates in our case) are selected with high probability to produce offspring. In a crossover process, some parts of the selected individuals (parent chromosomes) are combined to create a new individual (a child chromosome). Furthermore, in a mutation process the child's chromosome is randomly changed to introduce new genetic information. The children created by crossover and mutation are inserted into the new population, thereby replacing other low-fitness individuals. In our work, a GA is used for finding the number of times each TMR component should be checked per schedule period. The algorithm has the following characteristics:

(i) *Representation*: The solution domain is a population (P) in which each chromosome in P is an array consisting of N genes (e.g., $[d_1 \ ... \ d_N]$) representing N TMR components. $d_k, k = 1..N$, which are each greater than 0 and arranged into monotonically increasing order, that represent the number of times that each of the TMR components must be checked in one period of the schedule.

(ii) *Initialization*: An initial population is formed of individuals that are created by generating N random numbers between 1 and an upper bound value (e.g., 50) that are sorted into ascending order.

(iii) *Evaluation*: The fitness value is the system failure rate, as estimated using the analysis outlined in Section 3, corresponding to each individual in population P. Please note that individuals that do not satisfy the constraints on d_k and duplicate individuals are removed from the population before proceeding to the next step.

(iv) *Selection*: A *tournament selection* is adopted for the application of the selection procedure. This approach involves running several "tournaments" among a few individuals chosen at random from the population. The individuals with the best fitness are selected for the application of crossover and mutation.

(v) *Crossover*: We use a uniform crossover method, which randomly selects genes from the first and the second parent to generate an offspring. For example, with $N = 3$, two chromosomes [1 3 5] and [2 4 6] may create an offspring of [1 4 5] or [2 3 5]. The probability of crossover is a user-defined value (e.g., 0.25, as we expect that on average an offspring inherits 25% genes of the first parent and 75% genes of the second parent).

(vi) *Mutation*: Mutation alters one or more genes with a probability equal to the mutation rate (e.g., 10%) of a parent selected during the tournament. For example, with $N = 3$, assuming the second gene of the chromosome [1 2 5] is selected for mutation. A new value is generated by randomly adding 1 to or subtracting 1 from the mutated number, thus the chromosome after mutation would be [1 3 5] assuming addition was selected.

After the mutation function is finished, a new population is created and the evaluation procedure is repeated. When the GA function meets the stopping condition, it terminates and returns the best individual of the current population.

4.2 Scheduling of voter checks

Calculating the system failure rate requires all timing parameters, most of which can only be obtained from a real schedule. A real schedule must ensure fair voter checking among all TMR components. These voter checks, in turn, are required to be evenly distributed so that the temporal gap between any two consecutive checks of the same TMR component is as constant as possible. The problem of creating such a sequence of voter checks belongs to the class of *Response Time Variability Problems* (RTVP) [13], which arises whenever products, clients, jobs or, as in this work, voter checks need to be sequenced in such a way that the variability in the period between the instants at which they receive the necessary resource is minimized.

Unfortunately, the RTVP is NP-hard [13]. To solve our RTVP, we utilize the GA that is detailed in [12] to find the optimal schedule of voter checks.

4.3 Mean time to detect (MTTD) errors

The MTTD errors is defined as the average elapsed time interval between a configuration bit being affected by a fault and the instant that the erroneous TMR module is detected. The MTTD (in units of Δt_o) can be estimated as follows:

$$MTTD = \frac{\sum_{k=1}^{N} e_k \frac{D}{2d_k}}{\sum_{k=1}^{N} e_k} \tag{4.1}$$

where e_k denotes the number of errors that occur in component k and $D = \sum_1^N d_k$ denotes the length of one period of the voter checking schedule. If the voters are checked in round-robin order, $d_k = 1, \forall k$, $D = N$, and thus $MTTD = \frac{N}{2}$.

5 Experimental Analysis

We use Matlab to evaluate and compare the MTTF and reliability of TMR-MER systems employing VRVC with identical systems that use round-robin order for voter checking. We also evaluate the MTTD errors of both approaches by means of a fault injection experiment on an implemented sample system.

5.1 Experiments

We conducted three experiments.

Experiment 1 - Simulations: We computed the MTTFs of simulated TMR systems containing 2–5 and 10 components. We assumed that voters could be checked in $1\mu s$ in each system that we simulated. We then varied the voter checking period from $1\mu s$ to $1s$ in order to evaluate its impact on the reliability of systems that employ both VRVC and round-robin voter checking.

For the 2-component system, we conducted a case study with components containing 100,000 and 1,000,000 essential bits; and reconfiguration times of $0.2ms$ and $2ms$, respectively. A brute force search was conducted to find the optimal results.

For the systems involving between 3 and 10 components, the number of essential bits per TMR module was chosen randomly in a range varying from 10,000 to 2,000,000 bits using each of uniform, quadratic and exponential distributions.

The recovery time of a TMR module is given as the product of the number of its *Configuration Frames* (CFs) and the reconfiguration time per CF. We assumed that 30% of the configuration memory bits per CF were essential. Once a faulty module is detected, it should be recovered as quickly as possible. The reconfiguration time per CF was therefore assumed to be $2\mu s$, corresponding to the highest throughput of the ICAP, and $60\mu s$, which in our system is needed to retrieve a frame of data from off-chip memory when it is operated at 100MHz.

As detailed in Section 4, two GAs were implemented to obtain a schedule to yield the best possible system reliability. It should be noted that fine tuning the parameters of a GA is almost always a difficult task [14, 15]. In this work, we undertook preliminary experimentation using the following parameter values; further experimentation will be undertaken in the future to assess the full potential of our approach. The GA to determine the ratios in which components should be checked was initialised with 100 random chromosomes in which the value of each gene was randomly chosen in a range from 1 to 50. Since the simulation experiments are time-consuming, particularly for 10-component systems, we decided that the GA should be terminated after 100 generations. In addition, the crossover rate and the mutation rate were set to 25% and 10%, respectively. As discussed, we implemented the GA from [12] to find the best distribution of checks once the check rate was determined.

Table 5.1: Bit failure rates in GEO [16]

Orbit	**Altitude (km)/ Inclination**	**Solar Min**	**Worst Week**	**Worst Day**	**Peak 5-Min**
		λ (SEUs/Bit/s)			
GEO	$35{,}768/0^o$	1.71E-13	2.16E-11	7.34E-11	2.66E-10

For each distribution, we performed 5 runs and calculated the average ratio of the estimated MTTF for the TMR-MER system employing VRVC to that using round-robin sequencing for voter checks. For each run, we varied the bit failure rates as listed in Table 5.1. The "high radiation level" of bit failure rates as shown in Table 5.1 for Xilinx 7-series FPGAs [17] at *Geosynchronous Equatorial Orbit* (GEO) were derived using the CREME96 model [18] assuming $2.54mm$ aluminium shielding and a total number of device configuration memory bits (n_c) of 77,845,216. These parameters were also used in Experiment 2.

Experiment 2 - Implementation: We compared the reliability of an exemplar system comprising 9 TMR components (Fig. 5.1), using VRVC and round robin for voter checking in the four orbit conditions of Table 5.1. We also evaluated the trade-off between reliability and power consumption for both methods with the *Reconfiguration Controller* (RC) operating at clock frequencies of 100MHz, 50MHz, 20MHz, and 10MHz.

The exemplar system consisted of the 9 TMR components listed in Table 5.2. These TMR components were: a single MAC-based 21-tap *Finite Impulse 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. Due to power limitations, all 9 TMR components were operated at 10MHz.

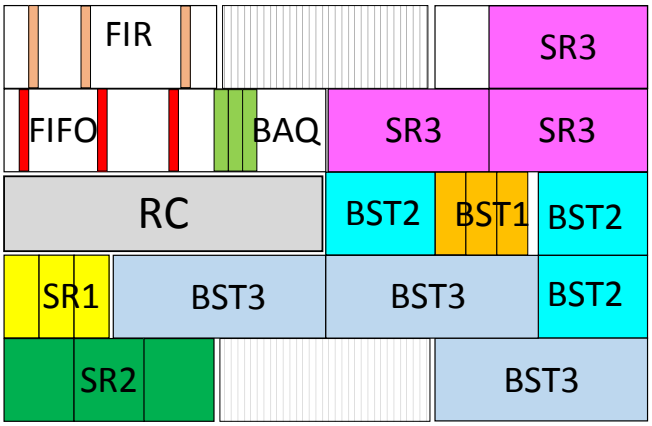

Figure 5.1: System layout for Experiments 2 and 3

An RC using the ICAP-based voter checking approach [8] was used to read the voter status bits. The RC includes a MicroBlaze (MB) processor connected to an *External Memory Controller* (EMC), a DMA *Controller* (DMAC) and the AXI HWICAP IP accessed via an AXI bus. The MB processor configuration is created with minimal features. The AXI HWICAP IP combines with EMC and DMAC to reconfigure faulty modules and is also used for flipping configuration memory bits during the fault injection experiment described in the next sub-section.

Table 5.2 lists the recovery times (t_r) for each TMR module. The recovery time is the time interval between an error being detected in a module until the last word of the partial bitstream used to recover that module is written to the AXI HWICAP IP.

The designs were implemented on a Xilinx Artix-7 XC7A200TFBG484-1 FPGA using Vivado 2014.4 with default settings.

Experiment 3 - Fault injection: We performed a fault injection experiment to assess the MTTD errors of the system of Experiment 2.

The RC was used to manage the fault injection process. The RC received a random configuration bit address generated by a host PC. The RC read the corresponding frame, flipped the addressed bit and wrote the frame back using the HWICAP to emulate an SEU. Note that we did not inject faults into the RC in order to avoid corrupting it during the experiment. Of the 18,300 configuration frames in the Artix-7 XC7A200TFBG-484 targeted in our study, 17,330 frames were contained in the design under test. Once a fault was injected, the RC waited for $10ms$ and checked the error status of all voters and reported which component, if any, was in error.

5.2 Results and Discussions

Through these experiments, it was demonstrated that the use of VRVC improves TMR-MER system reliability and the MTTD errors over the use of round robin for voter checking. The results are detailed in the following:

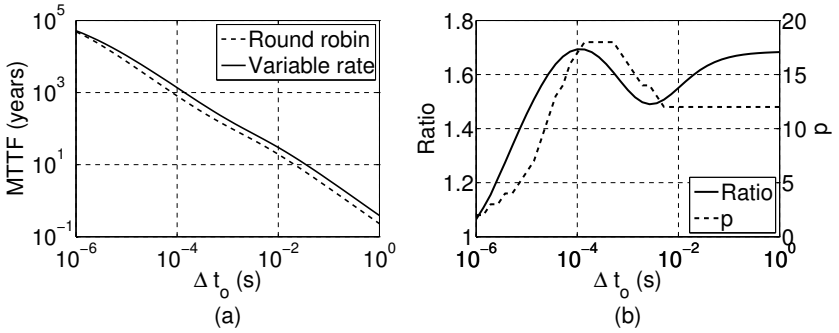

Figure 5.2: GEO worst week results: (a) MTTF (years) of the round-robin and variable-rate voter checking approaches. (b) Peak MTTF ratio achieved when varying the voter checking rate relative to checking voters in round-robin order, and the corresponding rate p at which C2 is checked relative to C1.

Results of 2-5 and 10 component systems

2-component system (GEO worst week results) Fig. 5.2(a) shows that a better MTTF is achieved by applying VRVC, while Fig. 5.2(b) shows that the number of checks p needed to obtain the best result varies depending on Δt_o. Moreover, the MTTF of VRVC improves significantly as Δt_o is increased and, in this case study, the MTTF was observed to be as much as 70% greater than when round robin is used for voter checking.

Systems of 3-5 and 10 components The ratios of MTTFs for systems employing VRVC to those checking voters in round-robin order are higher when the differences in component vulnerabilities are larger and/or when Δt_o is increased (Fig. 5.3). It can be observed that the ratios are almost independent of the orbital/radiation conditions and that the improvement is significant as Δt_o is increased.

Fig. 5.3 also shows that when the number of essential bits of all TMR components are exponentially distributed, the average ratios are more than 60% better for all simulated systems and as much as 100% better in 5 and 10 component systems, while when they are uniformly and quadratically distributed, the average ratios are up to 20% better and 40% better, respectively.

We observed that the increments of the ratios of MTTFs of VRVC to round robin are not consistent. We believe that these observations relate to the relative reconfiguration times (t_r) of the modules, which also influence system reliability.

Example design results

Table 5.2 provides the number of essential bits (n_e) and the recovery times (t_r) of the nine components. It also presents the number of checks (d_k) made per schedule period of each component so that the system MTTFs were maximized when the RC was operated at different clock frequencies under the GEO worst week condition (we observed similar d_k for the other GEO conditions). The time period between successive voter observations (Δt_o) was 71μs, 142μs, 355μs

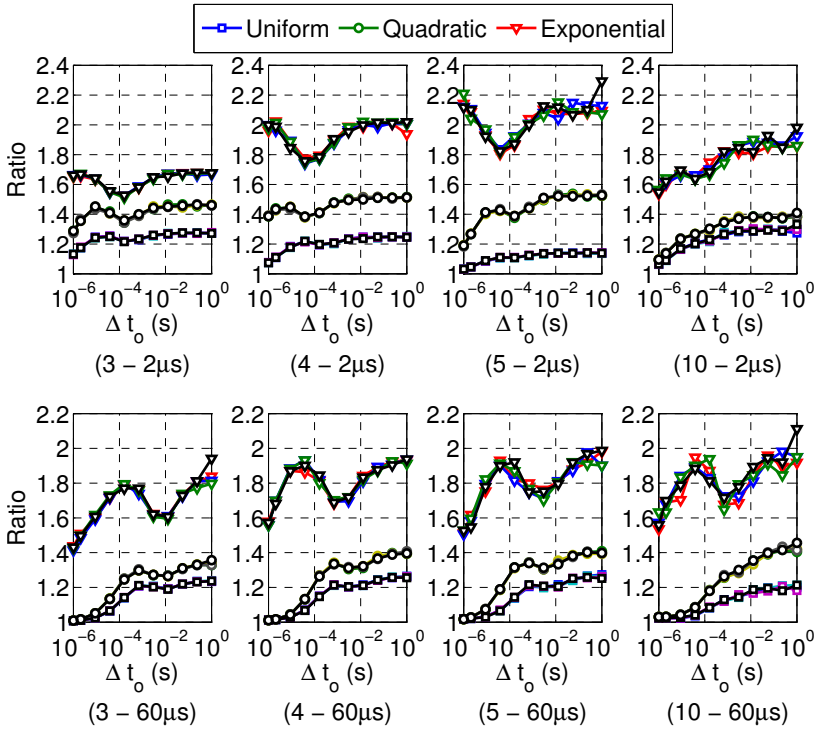

Figure 5.3: Average ratios of MTTFs for VRVC to those for round robin for systems consisting of 3, 4, 5, and 10 components for the four orbit conditions when the assumed reconfiguration time per frame is $2\mu s$ or $60\mu s$.

Table 5.2: Results of mapping 9 TMR components to Xilinx Artix-7 XC7A200TFBG-484 FPGA

Design	**Essential Bits** n_e	**RC t_r (ms) – # checks (d_k)**			
		100MHz	**50MHz**	**20MHz**	**10MHz**
BST3	1,833,235	26.7 – 47	49.5 – 45	72.4 – 47	118.7 – 49
SR3	1,403,647	19.6 – 41	43.8 – 40	64.0 – 39	104.9 – 46
BST2	793,534	11.0 – 28	24.5 – 31	35.8 – 34	58.7 – 36
SR2	515,904	8.5 – 27	21.7 – 29	31.7 – 33	52.0 – 29
SR1	285,914	6.8 – 26	13.6 – 24	19.9 – 25	32.6 – 25
BST1	281,604	2.6 – 23	5.9 – 23	8.6 – 20	14.0 – 25
BAQ	48,963	1.3 – 15	3.0 – 18	4.4 – 18	7.1 – 14
FIFO	41,842	3.5 – 12	7.8 – 12	11.4 – 13	18.7 – 13
FIR	12,042	1.2 – 08	2.6 – 11	3.9 – 10	6.3 – 11

and $711\mu s$ when the RC was operated at 100MHz, 50MHz, 20MHz and 10MHz, respectively.

Fig. 5.4 confirms the results of the previous sub-section that the reliability of the exemplar system using VRVC is always higher than when round robin is used for voter checking over all operating frequencies and across the four GEO

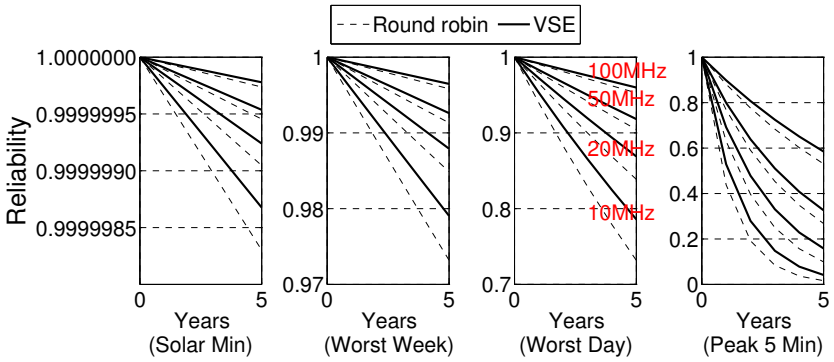

Figure 5.4: Reliabilities of using round-robin and VRVC approaches in the four orbit conditions

Table 5.3: MTTF ratio and power consumption

RC operating frequency		**100MHz**	**50MHz**	**20MHz**	**10MHz**
MTTF Ratio		1.18	1.17	1.25	1.30
Power (mW)	**RC**	252(0%)	196(-22%)	163(-35%)	152(-40%)
	RC+TMR components	456(0%)	394(-14%)	357(-22%)	344(-25%)

conditions.

Table 5.3 reports two metrics. The first is the ratio of MTTF for systems employing VRVC to those checking voters in round-robin order (the ratios are similar for all four orbit conditions). The second is the power consumption of the RC itself and the RC including the 9 components as well as the percentage decrease in the power consumption with the RC operating at various clock frequencies relative to the RC operating at 100MHz (the RC can employ either VRVC or round robin).

The TMR-MER system using VRVC is more reliable than the same system using round robin when the available power in the system is constrained. Fig. 5.4 shows that the system reliabilities are proportional to the rates at which the system recovers from errors. However, for the sake of energy saving in space-based applications in long-term missions, the checking frequencies may be decreased [19]. For example, when the RC runs at 10MHz compared to 100MHz, the energy consumption of the RC itself is reduced by 40% and that of the whole system is reduced by 25% (Table 5.3). In this case, the ratio of MTTF for VRVC to that obtained using round robin for voter checking increases from 18% for the RC operating at 100MHz to 30% at 10MHz (Table 5.3).

Fault injection results

Errors are detected 30% faster on average when using VRVC instead of round robin. Table 5.4(a) provides the average number of errors in each component that we found after four trials of one million injected faults. Table 5.4(b) tabulates the MTTD errors using both the round-robin and VRVC approaches as well as the percentage reduction from round robin to VRVC when the RC is operated at different clock frequencies. The MTTDs are calculated using Eq. (4.1) with

Table 5.4: (a) Number of errors found in components, (b) MTTD in Δt_o units

Design	# Errors e_k (%)
BST3	38,828 (39.1)
SR3	26,701 (26.9)
BST2	13,830 (13.9)
SR2	9,643 (9.7)
SR1	4,522 (4.6)
BST1	4,053 (4.1)
BAQ	684 (0.7)
FIFO	696 (0.7)
FIR	396 (0.4)

(a)

Freq.	Round robin	VRVC	%reduction
100MHz	4.5	3.2	28
50MHz	4.5	3.3	27
20MHz	4.5	3.1	31
10MHz	4.5	2.9	35

(b)

the number of checks listed in Table 5.2 and the average number of errors in each design tabulated in Table 5.4(a).

5.3 Further Discussion

Of considerable concern is that much of the additional logic used to implement and support TMR-MER may be implemented in a non-redundant manner such as the RC, the RCN and voters, and therefore introduces additional single points of failure. Nevertheless, irrespective of the configuration memory error recovery approach taken, FPGA-based TMR systems inevitably include non-redundant components such as the clock network, ICAP and off-chip ports, which also introduce single points of failure when used. Therefore, in order to further improve system reliability, the unreplicated modules need to be triplicated, if possible, and considered as TMR components. Since these components may be spread across the whole device, a standard partial reconfiguration design flow [20] cannot be used. One solution is to use FMER [21] that combines scrubbing and MER to recover configuration memory errors. In this case, our reliability model can also be applied to find a voter checking schedule that enhances, if not maximizes, system reliability.

A limitation of the proposed reliability model is that the number of failure cases increases exponentially with the number of TMR components. Approximation methods that reduce the complexity of the reliability models will be considered in the next stage of our work.

Finally, it should be noted that in our work we have neglected the additional system vulnerability that arises from the additional memory needed to store the schedule. However, this overhead is in the order of tens of bytes and should not pose a concern for overall system reliability.

6 Concluding Remarks and Future Work

In this paper, we have presented reliability models for TMR-MER systems that consist of an arbitrary number of components whose voters are checked in either round-robin order or at variable rates. We have proposed a genetic algorithm to derive a voter checking schedule that has the potential to significantly enhance the system reliability. We assert that any FPGA-based TMR system that uses a

reconfiguration control network to provide random access to component voters can benefit from using variable-rate scheduling to prioritize checks of more vulnerable components. The benefits become more significant as the radiation level increases and/or as the checking frequency decreases.

The results show that using VRVC improves the mean time for the system to fail by up to 70% in a case study of a two-component system compared to checking voters in a round-robin manner. The improvements were found to be over 60%, 40% and 20% in case studies of systems consisting of 3–5, and 10 components with assumptions that the numbers of essential bits of each of the TMR components are exponentially, quadratically and uniformly distributed, respectively. We have also shown that the power consumption of TMR-MER systems can be significantly reduced by reducing the operating clock frequency of the RC without compromising system reliability. Finally, through fault injection testing, we demonstrated that the mean time to detect errors can be reduced by 30% when using VRVC instead of round robin.

Our future work considers adaptive scheduling of voter checks based on the radiation level of the surrounding environment with the aim of optimizing the system reliability and power consumption. Another direction is to consider a user-defined metric (e.g., criticality level of each TMR component) in the reliability models. This is because some components such as clock managers are more critical than others, although they are small in terms of the number of their essential bits.

Acknowledgements

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

Bibliography

[1] C. Bolchini, A. Miele, and C. Sandionigi, "A novel design methodology for implementing reliability-aware systems on SRAM-based FPGAs," *IEEE Transactions on Computers*, vol. 60, no. 12, pp. 1744–1758, 2011.

[2] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, "Mitigation of radiation effects in SRAM-based FPGAs for space applications," *ACM Comp. Surveys (CSUR), vol. 47, no. 2*, 2015.

[3] N. T. H. Nguyen, E. Cetin, and O. Diessel, "Dynamic scheduling of voter checks in FPGA-based TMR systems," in *FPT*, Dec 2016, pp. 169–172.

[4] R. Le, "Soft error mitigation using prioritized essential bits," *Xilinx XAPP538 (v1.0)*, 2012.

[5] D. Agiakatsikas, N. T. H. Nguyen, Z. Zhao, T. Wu, E. Cetin, O. Diessel, and L. Gong, "Reconfiguration control networks for TMR systems with module-based recovery," in *IEEE International Symposium on Field-Programmable Custom Computing Machines*, May 2016, pp. 88–91.

[6] P. Ostler, M. Caffrey, D. Gibelyou, P. Graham, K. Morgan, B. Pratt, H. Quinn, and M. Wirthlin, "SRAM FPGA reliability analysis for harsh radiation environments," *IEEE Transactions on Nuclear Science*, vol. 56, no. 6, pp. 3519–3526, Dec 2009.

[7] M. Straka, J. Kastil, Z. Kotasek, and L. Miculka, "Fault tolerant system design and SEU injection based testing," *Microprocessors and Microsystems, vol. 37, no. 2*, pp. 155–173, 2013.

[8] F. Veljkovic, T. Riesgo, and E. de la Torre, "Adaptive reconfigurable voting for enhanced reliability in medium-grained fault tolerant architectures," in *AHS*, June 2015, pp. 1–8.

[9] H. Quinn, P. Graham, J. Krone, M. Caffrey, and S. Rezgui, "Radiation-induced multi-bit upsets in SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2455–2461, Dec 2005.

[10] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[11] L. Sterpone and M. Violante, "A new reliability-oriented place and route algorithm for SRAM-based FPGAs," *IEEE Transactions on Computers*, vol. 55, no. 6, pp. 732–744, June 2006.

[12] A. Garca-Villoria and R. Pastor, "Solving the response time variability problem by means of a genetic algorithm," *European Journal of Operational Research*, vol. 202, no. 2, pp. 320 – 327, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221709003567

[13] A. Corominas, W. Kubiak, and N. M. Palli, "Response time variability," *Journal of Scheduling*, vol. 10, no. 2, pp. 97–110, 2007. [Online]. Available: http://dx.doi.org/10.1007/s10951-006-0002-8

[14] T. Back, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*, 1st ed. Bristol, UK, UK: IOP Publishing Ltd., 1997.

[15] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on evolutionary computation*, vol. 3, no. 2, pp. 124–141, 1999.

[16] D. Heynderickx, B. Quaghebeur, E. Speelman, and E. Daly, "ESAs SPace ENVironment Information System (SPENVIS): a WWW interface to models of the space environment and its effects," *in 38th Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics*, vol. 371, 2000.

[17] D. Hiemstra and V. Kirischian, "Single event upset characterization of the Kintex-7 Field Programmable Gate Array using proton irradiation," in *2014 IEEE Radiation Effects Data Workshop (REDW)*, July 2014, pp. 1–4.

[18] A. Tylka, J. Adams, P. Boberg, B. Brownstein, W. Dietrich, E. Flueckiger, E. Petersen, M. Shea, D. Smart, and E. Smith, "CREME96: A revision of the cosmic ray effects on micro-electronics code," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, pp. 2150–2160, Dec 1997.

[19] I. Herrera-Alzu and M. Lopez-Vallejo, "Design techniques for xilinx virtex fpga configuration memory scrubbers," *IEEE Transactions on Nuclear Science*, vol. 60, no. 1, pp. 376–385, Feb 2013.

[20] *UG909: Vivado Design Suite User Guide - Partial Reconfiguration (v2016.1) April 6, 2016.*

[21] D. Agiakatsikas, E. Cetin, and O. Diessel, "FMER: A hybrid configuration memory error recovery scheme for highly reliable FPGA SoCs," in *FPL*, Sept 2016, pp. 88–91.