IET Computers & Digital Techniques

Research Article

IET Journals
The Institution of Engineering and Technology

Scheduling configuration memory error checks to improve the reliability of FPGA-based systems

ISSN 1751-8601
Received on 18th March 2018
Revised 20th July 2018
Accepted on 28th August 2018
doi: 10.1049/iet-cdt.2018.5001
www.ietdl.org

Nguyen Tran Huu Nguyen[1] ✉, Ediz Cetin[2], Oliver Diessel[3]

[1]*School of Computer Science and Engineering, Ho Chi Minh City University of Technology, 268 Ly Thuong Kiet, Ho Chi Minh City, Vietnam*
[2]*School of Engineering, Macquarie University, Sydney 2109, New South Wales, Australia*
[3]*School of Computer Science and Engineering, UNSW Sydney, Sydney 2052, New South Wales, Australia*
✉ *E-mail: nthnguyen@hcmut.edu.vn*

Abstract: Field-programmable gate arrays are susceptible to radiation-induced single event upsets. These are commonly dealt with using triple modular redundancy (TMR) and module-based configuration memory error recovery (MER). By triplicating components and voting on their outputs, TMR helps localise configuration memory errors, and by reconfiguring faulty components, MER swiftly corrects them. However, the order in which TMR voters are checked inevitably impacts the overall system reliability. In this study, the authors outline an approach for computing the reliability of TMR–MER systems that consist of finitely many components. They demonstrate that system reliability is improved when the more vulnerable components are checked more frequently than when they are checked in round-robin order. They propose a genetic algorithm for finding a voter checking schedule that maximises the reliability of TMR–MER systems. Results indicate that the mean time to failure (MTTF) of these systems can be increased by up to 400% when variable-rate voter checking (VRVC) is used instead of round robin. They show that VRVC achieves 15–23% increase in MTTF with a 10× reduction in checking frequency to reduce system power. They also found that VRVC detects errors 44% faster on average than round robin.

Nomenclature

Symbol Definition

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

1 Introduction

Owing to their low cost, flexibility, and impressive processing performance, static random access memory (SRAM)-based *field-programmable gate arrays* (FPGAs) are increasingly looked to as suitable candidates for hosting complex, high-performance space-based digital systems. However, the designers of such applications must consider the impact of ionising radiation, i.e. high-energy charged particles, has on the device [1]. In particular, *single event upsets* (SEUs) may alter the logic state of any static memory element, i.e. configuration latches, user flip-flops, internal block memory, and other device-specific control registers. Since millions of configuration latches within an FPGA are programmed to implement the user functionality, an SEU in the configuration memory can adversely affect the expected FPGA functionality.

The safe use of FPGAs in harsh radiation environments requires the implementation of robust SEU mitigation techniques. Hardware redundancy such as *triple modular redundancy* (TMR) is a commonly used technique [2, 3]. TMR can mask any single design failure by voting on the result of three functionally equivalent modules. However, TMR is unable to correct errors or eliminate erroneous values that have become trapped within cyclic user circuitry or within the configuration memory. Errors trapped in user circuitry can be corrected by resetting the faulty module or by resynchronising the module with its functionally equivalent siblings. To deal with configuration memory errors, TMR is usually combined with an error recovery technique such as scrubbing [1, 4, 5] or *module-based error recovery* (MER) [6]. We use the term TMR-scrubbing to refer to a TMR system in which the configuration memory errors are recovered by scrubbing, whereas TMR–MER is hereinafter used to refer to TMR systems that rely on MER to correct configuration memory errors.

Both TMR-scrubbing and TMR–MER rely on *dynamic partial reconfiguration* (DPR) to correct configuration memory errors. TMR-scrubbing is typically performed periodically and typically involves reading back each configuration memory frame of the device, checking it for errors using in-built error-correcting code (ECC) or by comparing it with a golden reference, correcting any errors that are found and writing back any corrected frame. In contrast, TMR–MER is commonly triggered when repeated errors are detected by the voter associated with a TMR component and involves rewriting the configuration memory for the specific module that has been found to be in error. TMR-scrubbing is thus more fine-grained than TMR–MER in its corrective action, but involves reading or writing the entire configuration memory contents of the device. On the other hand, TMR–MER is more coarse grained than TMR-scrubbing since the configuration memory contents of a complete module are rewritten; multiple configuration memory errors affecting the one frame/module are thus corrected in a single action, and since only the configuration memory of the affected module is rewritten, correction is typically faster with TMR–MER than with TMR-scrubbing.

During the last 20 years, more research has focused on the use of TMR-scrubbing than on TMR–MER to enhance the reliability of SRAM-based FPGA systems [2]. However, TMR–MER is being seen as offering some advantages over TMR-scrubbing. A significant drawback of TMR-scrubbing is that it results in unnecessary power consumption when it is invoked despite no SEU having been occurred. Furthermore, the delay in correcting errors using TMR-scrubbing may be excessive, particularly for large FPGA devices. TMR–MER aims to avoid these costs by

© The Institution of Engineering and Technology 2018

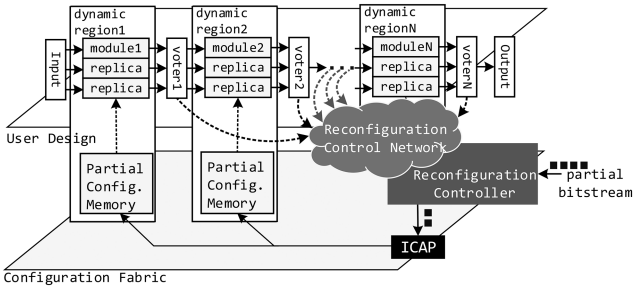

Fig. 1 *Example TMR–MER system diagram*

reconfiguring just the portion of the device that is suspected of being in error and by providing low-latency error detection via the TMR voters [6]. TMR–MER, thus, aids both the system power consumption and reliability [7, 8], which are both desirable outcomes for space-based systems.

Both TMR-scrubbing and TMR–MER utilise a controller to operate, but TMR–MER also require a means of relaying *reconfiguration requests* (RRs) from the voters in the system to a central *reconfiguration controller* (RC) (Fig. 1). A star-, bus- or ring-based *RC network* (RCN) is often employed to perform this function [9]. Another alternative is to use the in-built FPGA configuration infrastructure to access distributed status registers containing the RRs [9]. Irrespective of the RCN topology and technique employed, the RRs from voters cannot be checked concurrently. They must be checked sequentially. Typically, the RRs are checked in round-robin order [9]. However, the order in which the voter RRs of TMR components are checked has an inevitable impact on the overall system reliability.

In [10], we proposed an on-chip *voter scheduling engine* (VSE) to help the RC dynamically adjust the order in which RRs from TMR voters are checked based on the likelihood of the next component that is checked being in error. The approach was implemented based on the idea that the RRs from the more vulnerable components, i.e. those comprising a greater number of essential bits [11], are checked more frequently than the less vulnerable ones. We found that VSE boosted system reliability by 50% relative to a static round-robin voter checking schedule. A question that work raised, and *which we here answer in the affirmative*, is whether a static voter checking schedule could be found to enhance TMR–MER system reliability beyond the improvement possible with the dynamic voter checking method.

In this work, we investigate the reliability of TMR–MER systems consisting of multiple triplicated components operating in harsh radiation environments such as in geosynchronous orbit during solar flares, and in high-energy physics laboratories such as the large Hadron collider, where multiple coincident SEUs are more likely [12]. Our main interest is in determining the impact of varying the order and rate at which the voter RRs of TMR components are checked for errors in overall system reliability.

Our contributions are:

- To derive reliability models of TMR–MER systems that comprise finitely many TMR components whose voter RRs are checked in round-robin order and at a variable rate [so-called *variable-rate voter checking* (VRVC)]. Previous work has primarily focused on the effects of single SEU events on SRAM FPGA-based systems while our analysis considers the impact of multiple consecutive SEU events, which is an important consideration in providing a more comprehensive and accurate analysis of system reliability.
- To propose a *genetic algorithm* (GA) for finding the optimal fixed rate at which to check all components so as to maximise the *mean time to failure* (MTTF) and the reliability of TMR–MER systems.
- To show that the 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 the static round robin and the dynamic VSE approaches.
- To demonstrate that the *mean time to detect* (MTTD) errors are reduced by 44 and 30% on average when VRVC is used instead of a static round robin and dynamic VSE voter checking regimes, respectively.

In this paper, we build on and substantially extend our earlier work in [13, 14] as: (i) we provide a background on common error recovery techniques including TMR with scrubbing and MER as described in the literature (Section 1); (ii) we provide background on the reasons we use TMR–MER and the associated control logic as well as related work on techniques used to improve the reliability of FPGA-based systems for aerospace applications (Section 2); (iii) we detail the GAs used to obtain static schedules of voter checks that maximise the TMR–MER system reliability (Section 5); and (iv) finally, we report and discuss simulation results of systems containing 2, 4, 5, 10 and 20 components, and of an actual CubeSat system containing nine components (Sections 6 and 7).

This paper is organised as follows: Section 2 briefly provides background on related work on error recovery techniques used to improve system reliability. 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 presents the results of a small simulation study used to assess the models derived in Section 3 on a system comprising two components. Section 5 describes two GAs used to derive a voter checking schedule with the objective of maximising the system reliability when systems are composed of n components. Section 6 describes our simulation experiments to assess the performance of the proposed VRVC relative to VSE and round-robin voter checking, while Section 7 details our experimental method, reports on our findings and discusses the results. Concluding remarks and directions for further study are given in Section 8.

2 Background and related work

In this section, we provide a brief background on the TMR–MER approach to recover from configuration memory errors and review several research activities related to TMR–MER and TMR-scrubbing methods that use novel techniques to improve overall system reliability. We conclude with a brief survey of reliability models for SRAM FPGAs in the presence of SEUs.

2.1 TMR–MER approach

TMR–MER utilises the DPR capability available in modern FPGAs to recover a faulty module of a TMR component from configuration memory errors. DPR involves a post-configuration write operation to the configuration memory that is performed while the FPGA is operating. After a full configuration is loaded to the FPGA, DPR allows a system to repair SEUs in the configuration memory by loading a golden partial configuration file stored in a radiation-hardened off-chip memory. This file can be written to a pre-defined reconfigurable region in the FPGA without affecting the function of user circuits that are located in those regions of the device that are not being reconfigured [15].

Fig. 1 illustrates an FPGA-based TMR–MER system. The *voter* associated with each TMR component identifies which module, if any, is suffering from a persistent fault, and raises an RR [16]. RRs from the voters of different TMR components across the device are observed by an RC via an RCN. If the RC observes a fault in one of the modules in any of the system's TMR components, it fetches the partial bitstream corresponding to that module from off-chip memory and reconfigures it by writing the bitstream to the *internal configuration access port* (ICAP) present in advanced FPGAs from Xilinx. After the faulty module has been reconfigured and resynchronised with the remaining two modules of the TMR component, the voter resumes its normal checking function. It should be noted that RCNs such as star-based, bus-based, and ICAP-based networks allow the RC to check any one voter in constant time and in any order [9].

In [9], Agiakatsikas *et al.* provided a comprehensive study of TMR–MER using different RCN topologies. They demonstrated that TMR–MER using an ICAP-based network to aggregate voter RRs provides the highest reliability in comparison with systems implementing soft star-, bus- or token ring-based networks. The

© The Institution of Engineering and Technology 2018

ICAP-based network utilises the ICAP to read back configuration frames (CFs) that contain the health status, i.e. the RRs of the system's TMR components in user registers [9, 17]. An ICAP-based communications scheme eliminates the need for a soft network, and therefore improves reliability while reducing routing pressure and design implementation time. This approach has the potential to be scalable as it does not require user routing resources and utilises a moderate amount of logic to implement the central RC [9]. While in this work, we employ an ICAP-based RCN, our conclusions hold for any RCN that allows any voter to be accessed in a constant amount of time.

It would be possible to design a soft RCN to interrupt the RC when an error is detected. However, the reliability of the system is impacted by the considerable additional resources used by a soft RCN, and by the fact that errors in these RCNs are difficult to correct by modular reconfiguration because of their distributed nature. In [9], we compared a polled soft RCN with a polled ICAP-based RCN. A soft RCN that is interrupt based would use more resources than a polled one, for very small gains in MTTD errors, and hence the conclusions of [9] would favour an ICAP-based polled design even more emphatically.

2.2 Related work

A number of publications report on techniques that have been developed to decrease the MTTD and correct configuration memory errors in FPGA-based systems with a view to improve system reliability. These techniques are based either on TMR–MER [6, 9, 10, 16–18] or on scrubbing [19–23].

The use of scrubbing has been extensively researched and numerous studies have shown that scrubbing approaches that are based on the critical metrics that directly affect system reliability are more reliable than conventional scrubbing, which periodically and systematically scrubs from the first to the last frame of the device. Asadi and Tahoori [20] proposed an approach that reduces the MTTD errors by scrubbing only those CFs that contain sensitive bits, which affect the circuit operation when they are flipped. In [19], Lee *et al.* presented a heterogeneous scrubbing approach that varies the scrub rates of different components based on the number of sensitive bits they contain. In [22], Nazar *et al.* presented a mechanism that statistically finds an optimal frame to commence scrubbing in order to reduce the mean time to repair configuration memory errors. In addition, Santos *et al.* [23] showed that overall system reliability is improved when the rate at which scrubbing is performed is based on the criticality of user tasks. Our work presented here checks the voters of TMR components for configuration memory errors based not only on the number of sensitive bits of TMR modules, but also on the recovery times of the modules. This is similar to the approach of [19]; however, differs from it in that our approach does not involve or require any further action such as scrubbing or modular reconfiguration, when errors are not present.

Several models for estimating the reliability of SRAM FPGAs in the presence of SEUs have been introduced in the literature. Heron *et al.* [24] introduced a reliability model in which the overall reliability of the FPGA is calculated based on physical and SEU reliability. This model parses the netlist of a design to estimate the number of essential bits used for configuring essential items such as look-up tables, multiplexers, flip-flops/latches, wires and switch resources in the design. However, the model does not consider the effect of hardware redundancy or the effect of multiple coincident SEUs. Edmonds presented a reliability model of TMR designs without recovery that considers coincident upsets [25]. Ostler *et al.* [12] introduced a reliability model for a one-component TMR design employing TMR-scrubbing under harsh radiation environments, where multiple coincident SEUs are more probable. Their model requires orbit- and condition-specific SEU rates and design-specific estimates of the probability of failure during a single scrubbing period. Our work proposes reliability models of SRAM FPGA designs, also under harsh radiation environments, where multiple coincident SEUs may occur, such as [12], but for TMR–MER designs that contain multiple TMR components. The requirements of our models are similar to those of [12], but the probability of failure of a TMR component is estimated during two consecutive observations of the TMR component voters rather than during a single scrub period.

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 [9]. 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. In this paper, we develop reliability models that consider multiple coincident SEUs that may occur in different TMR components and use these to analyse the impact of the order in which we check voters for RRs.

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. We describe a general reliability model that has been widely used to estimate the reliability of FPGA-based systems. On the basis of 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 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 with respect to (w.r.t.) the component failure probability, $\mathrm{FP}_k(\Delta t)$, 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 [12] as

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

where the event F_k is the failure of the 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 [11].

$P(E_u, \Delta t)$, the probability of event E_u occurring during Δt, can be modelled with a Poisson distribution, $P(E_u, \Delta t) = \mathrm{e}^{-\nu}(\nu^u/u!)$, 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): $\nu = \lambda_{\mathrm{bit}} \times n_{\mathrm{c}} \times \Delta t$. λ_{bit} depends on the radiation level, the integrated circuit process technology and the circuit architecture of the FPGA fabric.

Once the failure probability of component k is known, the failure rate λ_k of a component k is given by [12]

$$\lambda_k = \frac{\mathrm{FP}_k(\Delta t)}{\Delta t}. \tag{2}$$

Since a TMR component can fail in different scenarios (see Fig. 2 and associated discussion in Section 3.2) with different failure rates

© The Institution of Engineering and Technology 2018

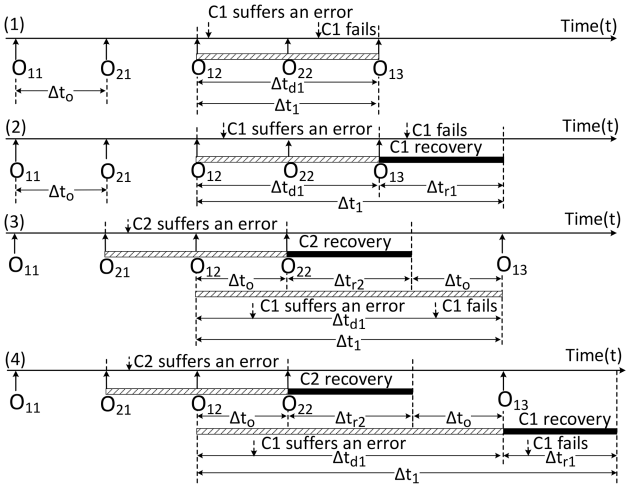

Fig. 2 *Failure modes for component 1 in two-component systems in which the voters are checked in round-robin order*

$\left(\lambda_k^i\right)$, it is more meaningful to compute the composite failure rate of each component $\left(\lambda_k^{\mathrm{c}}\right)$. This parameter can be calculated for the expected proportions $\left(\rho_k^i\right)$ in which each scenario occurs

$$\lambda_k^{\mathrm{c}} = \sum_{i=1} \rho_k^i \lambda_k^i . \tag{3}$$

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

Typically, a system contains N TMR components that are modelled in a series from a reliability perspective such that the failure of any one component causes the system to fail. The failure rate of a series TMR system, λ_{s}, is the sum of all component failure rates [26]. The MTTF of the system is given by the reciprocal of the system failure rate.

3.2 Failure rates of TMR–MER systems in which voters are checked in round-robin order

On the basis of 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 and to reduce the likelihood of the component failing [9].

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. 2 and using the notation listed in Nomenclature section. Note that Fig. 2 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. 2(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. 2(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. 2*(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).

Group 1: One other component suffers an error

- *Case 1 [Fig. 2(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 are 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. 2(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 modules/module of C1 while it is recovering from the previous error.

To summarise, 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 $\mathrm{FP}_k(\Delta t)$ in (1) with corresponding Δt_k as shown in Figs. 2(1) and (3).

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. 2(2) and (4) and that component k fails during the period of time Δt_{rk} given the occurrence of an event M_k.

On the basis of (2), the failure rate of the component k $\left(\lambda_k^i\right)$ in each case is estimated using the corresponding Δt_k (Fig. 2).

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 on the likelihood that component k suffers an error, while that of cases of group 1 occurring depend on the likelihood that both components suffer an error.

The composite failure rate of component k $\left(\lambda_k^{\mathrm{c}}\right)$ is calculated by substituting λ_k^i and ρ_k^i into (3), and the system failure rate can be computed by summing λ_k^{c} for all k.

The reliability of *systems comprising any number of TMR components* can be readily computed by extending the approach we have outlined for two-component systems by considering all possible cases in which each component may fail [27].

3.3 Failure rates of TMR–MER systems employing VRVC

VRVC is defined as a periodic schedule in which TMR 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 four 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 with the other components, and component 1 is deemed least vulnerable and hence checked less frequently.

3.3.1 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 using the notation of Nomenclature section. Note that p in Fig. 3 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(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 $x = 1 \ldots p$ 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 $\left(\rho_k^i\right)$. For example, given a schedule of two components in the following order *1–2–2–2–2–1–*

© The Institution of Engineering and Technology 2018

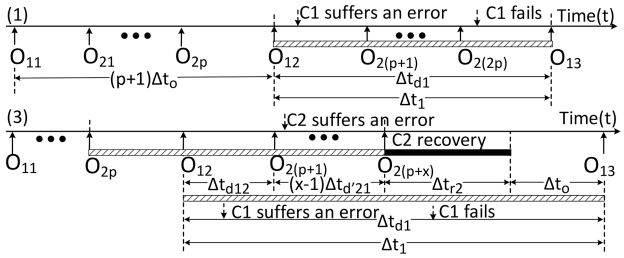

Fig. 3 *Failure modes for component 1 in systems comprising two components, which employs VRVC*

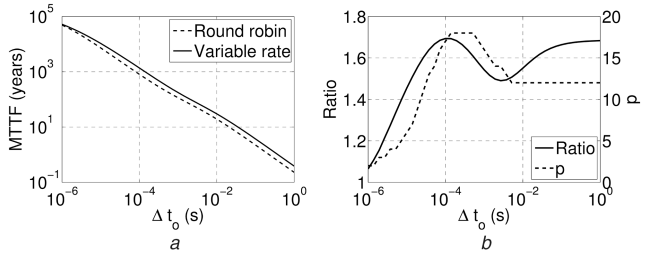

Fig. 4 *Results of the two-component system*
(a) MTTF (years) of the VRVC and round-robin voter checking approaches, ***(b)*** Peak MTTF ratio achieved when varying the voter checking rate relative to checking voters in round-robin order, and the corresponding rate p at which C2 is checked relative to C1 to achieve the peak MTTF ratio

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. Please note that errors can affect a component at any time between two consecutive checks of it. The impact is worst when the error occurs just after the component has been checked and is least when the error occurs just before the component is checked again. The expected impact is, therefore, calculated as the midpoint of the time interval between two consecutive checks of the component [21]. 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, $x = 1\ldots4$.

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.

3.3.2 N-component system: We assume that the components are numbered $k = 1\ldots N$ and ranked into non-decreasing vulnerability (number of essential bits) order, and that component k is, therefore, not checked less 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.

The system failure rate can be computed by considering all possible cases in which each component may fail [27].

4 Simulating two-component systems

We interrupt the main flow of our presentation to report on an early simulation we undertook to assess the performance of VRVC. In this section, we present results of simulations that aimed to assess the reliability of systems comprising two components using both round-robin and VRVC as the time period between successive voter observations was varied. As will be seen, non-linearities in our results suggest that determining an optimal voter checking schedule is likely to be NP-hard and thus motivate the development of evolutionary approaches to find good schedules, as discussed in the next section.

In our paper, we simulated systems comprising two components, C1 and C2, containing 10^5 and 10×10^5 essential bits and having assumed reconfiguration times of 0.2 and 2 ms, respectively. Simulation parameters were based on a Xilinx Artix-7 XC7A200T device, containing $n_c = 77,845,216$ configuration bits in total, and operating in a geostationary equatorial orbit (GEO) with an anticipated upset rate of 2.66×10^{-10} upsets/bit/s [10]. We used expressions that can be derived by extending the analyses of Sections 3.2 and 3.3 to calculate the MTTF for a round-robin checking schedule, as well as for VRVC as the number of checks, p, of C2 was varied relative to the one check made per period of component C1. These calculations were performed as the voter observation period (Δt_o) ranged from 1 μs to 1 s.

Our results, plotted in Fig. 4*a*, show that a better MTTF is achieved by VRVC at all voter observation periods. Fig. 4*b* shows that the number of checks p of C2 that is needed for each check of C1 to obtain the best MTTF for VRVC relative to round robin varies depending on the voter observation period. The results confirm our intuition that the larger components should be checked more frequently than smaller ones in order to maximise system reliability. With the assumptions made in this paper, we found that use of VRVC rather than round robin to check voters lead to a significant improvement in system MTTF of up to 70%.

As can also be seen in Fig. 4*b*, the maximum benefit of VRVC (relative to round robin) and the number of checks p of C2 needed to achieve this optimal MTTF are non-linear functions of the voter observation period. Unfortunately, it is infeasible to use an exhaustive search, as we have done in this paper, to determine the optimal number of checks to perform on each component as the *number of components* in the system increases. For this reason, in the next section, we develop a heuristic approach using evolutionary algorithms to determine a good checking schedule.

5 Scheduling voter checks

We surmise that 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 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 maximise the system reliability. We refer to this GA as the outer GA in Fig. 5. Once the rate at which components should be checked has been determined, we use the second GA, as detailed in [28], to *generate a schedule in which the determined number of voter checks are distributed as evenly as possible* per schedule period. We refer to the second GA as the inner GA in Fig. 5. The schedule produced by the inner GA is needed to evaluate the fitness of individual solutions to the first GA, which determines the number of checks to be performed. The inner GA is, therefore, nested within the outer GA's evaluation function as shown in Fig. 5.

5.1 Proposed GA

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 altered 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, the outer GA is used for finding the number of times each TMR component should be checked per scheduled period. A flow diagram of the algorithm is shown in Fig. 5 and the algorithm has the following characteristics:

© The Institution of Engineering and Technology 2018

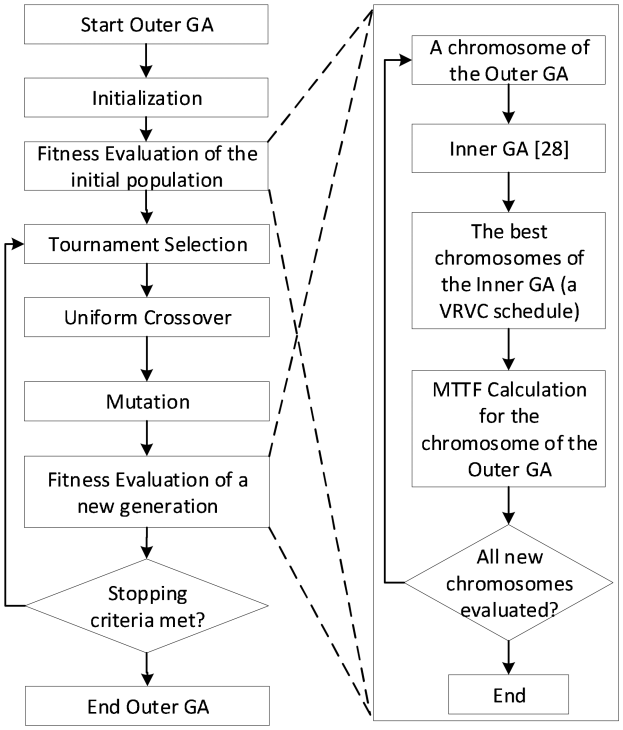

Fig. 5 *Flow diagram of GA-based VRVC schedule generation*

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

(ii) *Initialisation*: An initial population is formed of individuals that are created by generating N random numbers between 1 and an upper bound (UB) 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 the population P. Please note that chromosomes that do not satisfy the constraints on d_k (illegal chromosomes) and duplicate chromosomes are removed from the population before proceeding to the next step. Eliminating duplicate chromosomes prevents super chromosomes from dominating the population and helps maintain the diversity of the population [29].

(iv) *Selection*: A *tournament selection* is adopted for the application of the selection procedure [30]. 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 applications of crossover and mutation. Note that selection is performed on the enlarged sample space in which both parents and offspring have the same chance of competing for survival.

(v) *Crossover*: We use a uniform crossover method in which each gene of the offspring's genome is independently chosen from the two parents according to a given distribution [29]. Using a uniform crossover method eliminates the positional bias of inheritance, whereby genes that are close together on the chromosome are more likely to be inherited together. 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.

5.2 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* (RTVPs) [31], which arise 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 minimised.

The RTVP is formulated as follows. Given N positive integers $d_1 \leq \cdots \leq d_N$ associated with the number of checks of N TMR components, respectively, let $D = \sum_{k=1}^{N} d_k$ and the rates $r_k = (d_k/D)$ for $k = 1 \ldots n$. Let $\boldsymbol{S} = s_1 s_2 \ldots s_D$ be a vector of length D, where TMR component k occurs exactly d_k times, be a solution of an instance of the RTVP that consists of a circular sequence of copies of $\boldsymbol{S}$. For any two consecutive checks of TMR component k we define a distance τ between them as the number of voter checks of other TMR components that separate them plus 1. Since TMR component k is checked exactly d_k times in $\boldsymbol{S}$, then there are exactly d_k distances $\tau_1^k, \ldots, \tau_{d_k}^k$ for k. Note that the sequence is circular, s_1 comes immediately after s_D; therefore, $\tau_{d_k}^k$ is the distance between the last occurrence of TMR component k in the preceding cycle and the first occurrence of the same component in the current cycle. Obviously, the two are the same for $d_k = 1$. Since

$$\tau_1^k + \cdots + \tau_{d_k}^k = D, \tag{4}$$

then the average distance $\bar{\tau}_k$ between the TMR component k in $\boldsymbol{S}$ equals

$$\bar{\tau}_k = \frac{D}{d_k} = \frac{1}{r_k}, \tag{5}$$

and it is the same for each feasible sequence $\boldsymbol{S}$. The RTV for TMR component k is formulated as follows:

$$\mathrm{RTV}_k = \sum_{j=1}^{d_k} \left(\tau_j^k - \bar{\tau}_k\right)^2, \tag{6}$$

and the objective is to minimise the total RTV that is formulated as follows:

$$\mathrm{RTV} = \sum_{k=1}^{N} \mathrm{RTV}_k = \sum_{k=1}^{N} \sum_{j=1}^{d_k} \left(\tau_j^k - \bar{\tau}_k\right)^2. \tag{7}$$

Unfortunately, the RTVP is NP-hard [31]. To solve our RTVP, we utilise the GA that is detailed in [28] to find the optimal schedule of voter checks.

5.3 MTTD errors

The MTTD errors are 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:

© The Institution of Engineering and Technology 2018

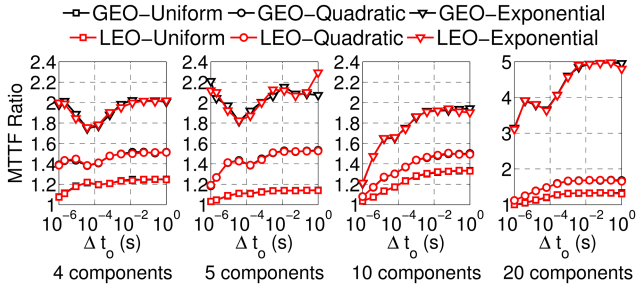

Fig. 6 *Average ratios of MTTFs for VRVC to those for round robin for systems consisting of 4, 5, 10 and 20 components in LEO and GEO*

$$\text{MTTD} = \frac{\sum_{k=1}^{N} e_k \frac{D}{2d_k}}{\sum_{k=1}^{N} e_k} \tag{8}$$

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 $\text{MTTD} = \frac{N}{2}$.

5.4 Power consumption

The power consumption of the components involved in the ICAP-based voter checking approach can be divided into two parts: the power consumed and dissipated by the configuration port and built-in configuration controllers and power consumed and dissipated by the RC circuitry. Both parts substantially contribute to the overall FPGA power consumption and dissipation [4]. It can be expected that the use of the ICAP for readback introduces power overheads for the RC. However, the overhead is expected to be quite small due to the relatively few resources consumed. Unfortunately, current tools do not allow this overhead to be measured.

Checking voters continuously provide the quickest response to errors, and therefore the highest reliability. When error rates are high, this may be necessary. However, for much of the mission, the error rates may be low enough to allow the checking rate, and thus the power consumption, to be reduced.

In this paper, we used the Xilinx design tool Vivado 2014.4 to estimate the static and dynamic powers of the RC and all TMR components when the RC was run at different clock frequencies.

6 Evaluation – synthetic problems

Simulation experiments were carried out to assess the performance of the proposed VRVC schedule relative to both round-robin voter checking and the dynamically determined VSE approach presented in [10]. These experiments involved simulating FPGA-based systems comprising sets of TMR components and assessing the MTTF for each voter checking method at low- and high-radiation orbits. In the following, we describe our experimental set-up before presenting the results of the simulations.

6.1 Assumptions and implementations

Our reliability models require an orbit-specific SEU rate and design-specific parameters such as the total number of configuration bits in a device, the voter observation time, the number of essential bits for each module, the recovery time of each module and the voter observation schedule, which is produced by the GAs.

Similar to the case study of the two-component system reported in Section 4, we simulated a Xilinx Artix-7 XC7A200T device operating in GEO and in low-Earth orbit (LEO). The radiation levels at these orbits are expected to result in 2.66×10^{-10} and 8.46×10^{-12} upsets/bit/s, respectively [10]. The total number of device configuration memory bits for this device is $n_c = 77,845,216$. We assume that the voter RRs are checked in 1 μs, which corresponds to the fastest CF read time that the ICAP-based network can provide for this device. To assess the impact on the reliability of reducing the system power consumption by reducing the voter checking frequency, we varied the voter checking period, Δt_o, from 1 μs to 1 s.

We simulated systems comprising 3–20 TMR components to assess the performance of the schedules being studied as the number of components was varied. For each simulation, the size of each triplicated module, as measured in number of essential bits, was chosen randomly in a range from 10,000 to 2,000,000 bits using one of three size distributions: uniform, quadratic and exponential. For a given number of system components (3–20) and at each orbit level (LEO and GEO), we evaluated the performance of each schedule (VRVC, round robin, VSE) over five trials for each module size distribution (uniform, quadratic and exponential). Please note that there is no difference in hardware cost between VRVC and RR as both are controlled by a programmable RC. However, VSE uses a hardware controller to implement the checking algorithm in addition to an RC, which is needed to reconfigure erroneous modules. When comparing the system reliabilities obtained with VRVC and VSE, we assume that the VSE controller is not susceptible to errors so as not to distort the results with the additional hardware used by VSE. In the following, we report on the average of the five trials carried out for each distribution.

The recovery time of a TMR module is given as the product of the number of its CFs and the reconfiguration time per CF. As the typical ratio of configuration bits not affecting a design to those (essential) bits that do affect a design ranges between 9:1 and 4:1 [32], we made the pessimistic assumption that the ratio is 9:1 in our simulations. In other words, we assumed that 10% of the configuration memory bits per CF were essential. This has the effect of dilating the time to recover module configuration memory errors, which in turn increases the likelihood that subsequent errors occur before recovery from a previous error has been completed.

As detailed in Section 5, two GAs were implemented to obtain a schedule to yield the best possible system reliability. Fine tuning the parameters of a GA is almost always a difficult and time-consuming task [33, 34]. In this section, we undertook experimentation using the following parameter values; further experimentation with the GA parameters is reported on in the next section. The GA to determine the rate at which components should be checked per period was initialised with 100 random chromosomes with the value of each gene being randomly chosen to be in the range 1–50 with a uniform distribution. Since the simulation experiments are time-consuming, particularly for 20-component systems, we decided that the GA should be terminated after 100 generations. In addition, the crossover rate and the mutation rate were set at 25 and 10%, respectively. As discussed, we implemented the inner GA from [28] to find the best distribution of checks once the check rates had been determined. The same parameter values were used to run the inner GA, namely, N (size of the population) = 13, p (mutation probability) = 0.013, B (proportion of best chromosomes) = 0.18 and R (proportion of new chromosomes) = 0.12. The number of generations was set to 1000 for each run. The inner GA was stopped when the maximum number of generations was reached, or the schedule was evenly distributed, which is unlikely.

6.2 Results and discussion

The results of our simulations demonstrate that the use of VRVC improves TMR–MER system reliability. The results are detailed as follows.

6.2.1 VRVC versus round robin: The ratio of MTTFs for systems employing VRVC to those checking voters in round-robin order is consistently >1, and become larger as the voter observation time, Δt_o, is increased (Fig. 6). It can be observed that the ratios are almost completely independent of the orbital/radiation conditions.

Fig. 6 also shows that when the number of essential bits of all TMR components is exponentially distributed, the average ratios are more than 80% better for all simulated systems and as much as 400% better in 20-component systems, while when they are

© The Institution of Engineering and Technology 2018

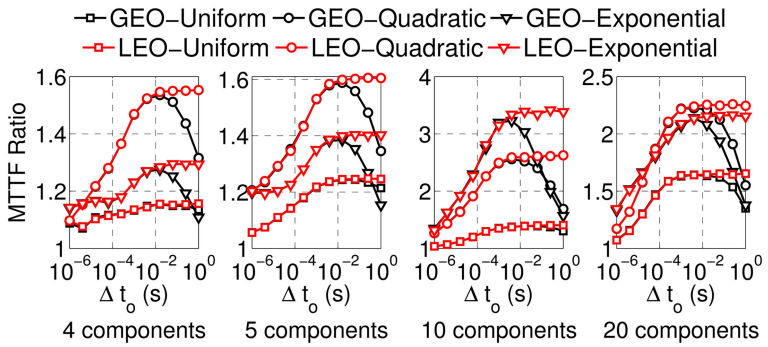

Fig. 7 *Average ratios of MTTFs for VRVC to those for VSE for systems consisting of 4, 5, 10 and 20 components for LEO and GEO*

Table 1 GA execution time profile

Number of Components		5			10			20		
GA	Function	Number of calls	Total time, s	Average, s	Number of calls	Total time, s	Average, s	Number of calls	Total time, s	Average, s
inner GA		180	449	2.5	186	798	4.4	229	2,542	11.1
outer GA	evaluation	180	455	2.5	186	6,140	34	229	18,852	82.3
	selection	100	0.0046	5×10^{-5}	100	0.012	1×10^{-5}	100	0.015	2×10^{-5}
	crossover	100	0.0028	3×10^{-5}	100	0.007	7×10^{-5}	100	0.009	9×10^{-5}
	mutation	100	0.0036	4×10^{-5}	100	0.014	1×10^{-4}	100	0.017	2×10^{-4}
execution time for 100 generations		455	—	—	6140	—	—	18,852	—	—

uniformly and quadratically distributed, the average ratios are up to 40% better and 60% better.

We observed that the change in MTTF ratios was not as consistent as Δt_o was varied. We believe this observation relates to the relative reconfiguration times of the modules, which also influence system reliability.

To summarise, the VRVC approach was found to be always beneficial, but it is of greatest benefit when the system contains many TMR components that differ markedly in size.

6.2.2 VRVC versus VSE: In all simulated systems involving 4–20 components, the MTTFs for VRVC are higher than those for VSE and the gap between the two approaches increases as the voter observation time is increased (Fig. 7). However, we also observed that the gap decreases when the voter observation time becomes > 10^{-2} s at very high-radiation levels, as present in GEO. This is because the MTTFs eventually decline to 0, as partly shown in Fig. 4*a*, when the voter observation time becomes so large as to significantly increase the likelihood of component failures since recoveries are not undertaken frequently enough. Furthermore, since VSE adopts dynamic voter checks based on the number of essential bits per TMR component, the MTTF results of the VSE approach also vary and are unpredictable.

6.2.3 GA performance: The GAs was implemented in MATLAB 2017a running under 64-bit Windows 7 Professional SP1. The execution time was obtained using MATLAB's profiling tool. We used an Intel(R) Xeon(R) 2.8 GHz with 12 GB RAM. The hardware contains four processors and MATLAB utilised around 30% of central processing unit capacity.

Table 1 shows the average number of calls, the total processing time and the average processing time of each function in the outer GA and of the inner GA for systems containing 5, 10 and 20 components. Please note that the processing time of the fitness calculation of the outer GA also includes the processing time of the inner GA. The processing time of a GA depends on the initial population size (PS), the number of new chromosomes created after applying crossover and mutation and its stopping criteria. In our experiments, the execution times of the outer GA stopping at 100 generations when simulating systems containing 5, 10 and 20 components are 455, 6140 and 18,852 seconds respectively (Table 1).

The profiling shows that most of the processing time is spent evaluating values in the outer GA. A small fraction of processor time is used to run the genetic operators of the outer GA. The ratio of the processing time of the outer GA fitness evaluation to that of the inner GA increases as the number of system components increases because of the exponential increase in the number of cases in which errors may occur in the TMR–MER system (Section 3).

Given the results of Table 1, we believe systems with larger numbers of TMR components can feasibly be evaluated depending on the processing power of the system used for scheduling.

7 Evaluation – case study

In this section, we evaluate and compare the performance of the VRVC approach with that of VSE and round-robin voter checking when each is implemented on an experimental CubeSat payload known as RUSH [9, 35]. This exemplar system comprises nine TMR components and a MicroBlaze (MB)-based RC hosted on an Artix-7 XC7A200TFBG-484 FPGA from Xilinx, as depicted in Fig. 8. We assessed the reliability of the system using VRVC, VSE and round robin for voter checking in LEO and GEO. In the course of this assessment, we examined the response of the GA used to schedule VRVC as its parameters were fine tuned. Our experiments also gauged the trade-off between system reliability and power consumption for the three methods studied as the clock frequency of the RC was reduced. Finally, we conducted fault injection testing of the exemplar system in order to evaluate the MTTD errors using each of the three methods.

7.1 Experiments

7.1.1 Implementation: The RUSH system was designed and built to compare the performance of module- and scrubbing-based approaches to configuration memory error recovery in SRAM-based FPGAs as part of the European Commission funded QB50 experiment [9, 35, 37]. The RUSH payload consists of the nine TMR components listed in Table 2. These components were selected as being representative of circuits that are commonly included in space-based applications and that utilise a mixture of

© The Institution of Engineering and Technology 2018

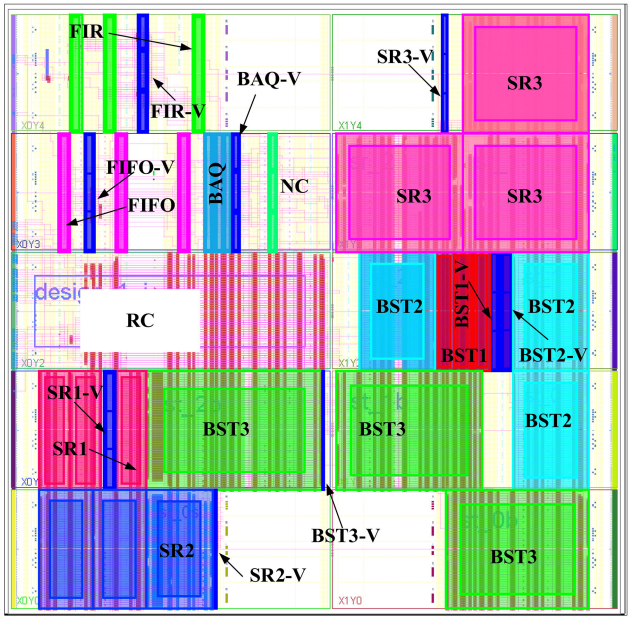

Fig. 8 *System layout of RUSH payload [36]*

FPGA resources. They include: a single media access controller (MAC)-based 21-tap *finite impulse response* (FIR) filter with 16 bit signal width; an 8–3 bit *block adaptive quantiser* (BAQ); an 8096-word deep 32 bit first-in–first out (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. Owing to power limitations of the CubeSat that deploys the exemplar system, all components are operated at 10 MHz.

An RC using the ICAP-based voter checking approach [17] is used to read the voter status bits. Please note that in [36], the component voters were checked in round-robin order while in this paper the RC runs our proposed VRVC algorithm. The RC includes an MB processor connected to an *external memory controller* (EMC), a DMAC and the AXI HWICAP IP accessed via an AXI bus. The MB processor configuration is created with minimal features and can be operated at 100, 50, 20 or 10 MHz. 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 section.

The designs were implemented using Vivado 2014.4 with default settings.

7.1.2 Fault injection: We performed a fault injection experiment to assess the MTTD errors in the RUSH system using each of the three voters checking schedules.

The RC was used to manage the fault injection process. The RC received a random configuration bit address generated by a host personal computer. The RC reads the corresponding frame, flipped the addressed bit and wrote the frame back using the HWICAP to emulate the occurrence of a memory error due to 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 CFs in the FPGA targeted in our study, 17,330 frames were contained in the design under test. Once a fault was injected, the RC waited for 10 ms and checked the error status of all voters before reporting which component, if any, was in error.

7.2 Results and discussion

7.2.1 Example design results: Table 2 reports the number of essential bits (n_e) and the recovery times (t_r) per triplicated module (t_r is the time interval between errors being detected in a module until the last word of the partial bitstream used to recover that module is written back to the FPGA via the AXI HWICAP IP). The table also reports the number of checks (d_k) made of each component per VRVC schedule period so that the system MTTF was maximised when the RC was operated at different clock frequencies under GEO radiation conditions (we observed similar d_k under LEO conditions). The time periods between successive voter observations (Δt_o) were 71, 142 s, 355 and 711 μs when the RC was operated at 100, 50, 20 and 10 MHz, respectively. This is a consequence of the number of clock cycles needed by the RC at that frequency to process the instructions to check a voter.

Table 3 reports two metrics. The first is the MTTF in years (and the percentage MTTF decrease) for systems employing VRVC, VSE and round robin (and w.r.t. the VRVC system) for voter checking in GEO. The second is the power consumption in mW of (i) the RC on its own and (ii) the RC including the nine components when the RC is operated at different clock frequencies. The percentage reduction in power consumption, relative to the RC operating at 100 MHz, is indicated in parentheses. This power consumption figure relates to the energy expended checking the voters at intervals of Δt_o, and therefore applies to all schedules equally.

Table 2 Results of mapping nine TMR components to Xilinx Artix-7 XC7A200TFBG-484 FPGA

Design	Essential bits n_e	RC t_r, ms – number of checks (d_k)			
		100 MHz	50 MHz	20 MHz	10 MHz
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

Table 3 MTTF and power consumption at various RC clock frequencies in GEO

RC operating frequency		100 MHz	50 MHz	20 MHz	10 MHz
MTTF, years	RR	103.0 (–15%)	49.0 (–15%)	28.0 (–20%)	16.0 (–23%)
	VSE	116.4 (–4%)	54.9 (–5%)	32.6 (–7%)	19.2 (–7%)
	VRVC	121.7 (0%)	57.6 (0%)	35.1 (0%)	20.7 (0%)
power, mW	RC	252 (0%)	196 (–22%)	163 (–35%)	152 (–40%)
	RC + TMR	456 (0%)	394 (–14%)	357 (–22%)	344 (–25%)

© The Institution of Engineering and Technology 2018

We found that 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. Table 3 shows that the system reliabilities (as given by the MTTF) are proportional to the rates at which the system recovers from errors. However, for the sake of saving energy in space-based applications during long missions, the voter checking frequency can be reduced [4]. For example, when the RC runs at 10 MHz compared with 100 MHz, the energy consumption of the RC alone is reduced by 40% and that of the whole system is reduced by 25% (Table 3). In this case, the ratio of the MTTF achieved using VRVC to that obtained using round robin for voter checking increases from 118% for the RC operating at 100 MHz to 129% at 10 MHz. It can also be observed that the MTTFs of systems employing VRVC are greater than those that employ VSE at all four RC clock frequencies.

Fig. 9 plots the ratio of the MTTF for systems employing VRVC to the MTTF for those checking voters in round-robin order as the parameters of the proposed GA are varied. The parameters we varied included the UB for the number of checks to be performed per period in the initial population. The UB was set to a small number of 10 initially and increased to 20, 50 and 100. For each UB, we also varied the PS from 10 to 20, 50 and 100. For each combination of UB and PS, we run 1000 generations. The mutation rate and the crossover rate were left at 50 and 25%, respectively. Here, we show the results of the RC operating at 10 MHz under GEO conditions. As can be seen in Fig. 9, the GA tends to attain a similar optimal result in most cases. We also observed similar trends when the mutation rates were set at 10 and 90% and the crossover rates were set at 50 and 75%.

The experiments show that the UB affects the performance of our proposed GA. This is because when UB is small (e.g. 10), the genetic diversity is limited and the GA is likely to become stuck in a local optimum. When UB is large (e.g. >50), the GA is more likely to find the global optimum. On the other hand, PS affects the starting point of the GA, but it does not significantly affect the final results.

To summarise, in order to find a schedule for maximising the system reliability, we found that the proposed GA should be initialised with a small PS (e.g. 10) to aid rapid evaluation, a modest UB (e.g. 50) for the sake of obtaining short schedule lengths and to assist in finding the optimal result, and a large number of generations (e.g. >1000) to have a good chance of attaining the optimal result.

7.2.2 Fault injection results: On average, VRVC allows errors to be detected 44% faster than with round robin and 30% faster than when VSE is used to check voters. Table 4 provides the average number of errors in each component that we found after four trials of one million injected faults. Table 5 tabulates the MTTD errors using the round-robin, VSE and VRVC approaches as well as the percentage reduction from round robin to VSE and VRVC when the RC is operated at different clock frequencies. The MTTDs are calculated using (8) with the number of checks listed in Table 2, and in [10], together with the average number of errors per component, as tabulated in Table 4.

7.3 Further discussion

It is of some concern that much of the additional logic used to implement and support TMR–MER such as the RC, the RCN and voters may be implemented in a non-redundant manner, and therefore introduce single points of failure. Nevertheless, irrespective of the configuration MER 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 non-replicated modules should be triplicated, if possible, so as to be protected from SEUs along with the other TMR components of the system. Since these components may be distributed across the device, a standard partial reconfiguration design flow [15] cannot be used to recover them in a modular fashion. One solution is to use a hybrid approach, called frame- and module-based configuration memory error recovery (FMER), which combines modular recovery of triplicated components with the scrubbing of single points of failure to achieve enhanced reliability and availability with reduced power [8]. VRVC complements FMER in order to provide the best possible protection for the triplicated components.

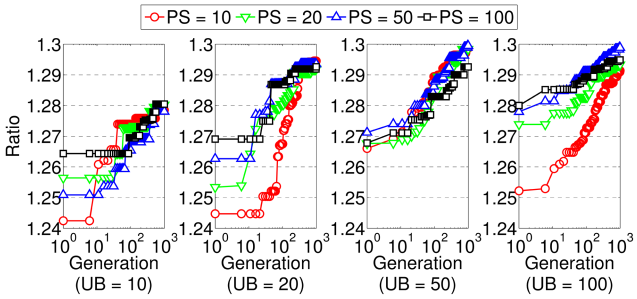

Fig. 9 *Ratio of MTTF for VRVC to MTTF for round robin for the exemplar system while the number of generations, the PS and the UB of the initial check rate are varied*

Table 4 Average number of errors found in components

Design	Number of errors e_k, %
BST3	38,828 (39.1)
SR3	26,701 (26.9)
BST2	13,830 (13.9)
SR2	9643 (9.7)
SR1	4522 (4.6)
BST1	4053 (4.1)
BAQ	684 (0.7)
FIFO	696 (0.7)
FIR	396 (0.4)

Table 5 MTTD errors

Configuration\Δt_o	71 μs	142 μs	355 μs	711 μs
round robin, μs	320 (–39%)	639 (–37%)	1596 (–45%)	3200 (–53%)
VSE, μs	290 (–26%)	580 (–25%)	1451 (–31%)	2905 (–39%)
VRVC, μs	230 (0%)	465 (0%)	1105 (0%)	2088 (0%)

A limitation of the reliability models proposed in this paper is that the number of failure cases increases exponentially with the number of TMR components in the system, which impacts on the scalability of our approach. Approximation methods that reduce the complexity of the reliability models will be considered in the next stage of our work. It should be noted that other approaches such as Markov models [9, 38] face the same problem since the number of states increases exponentially with increasing component numbers.

Our results are based on a limited exploration of the GA parameter space. Further experimentation needs to be undertaken to assess the full potential of our approach. To obtain better system reliabilities, the initial PS and the maximum number of generations of the outer GA may need to be increased and the applications of the selection, mutation and crossover operators may need to be modified.

Finally, it should be noted that in our work we have neglected the additional system vulnerability that accrues from the memory used to store the schedule. However, this overhead is small, being on the order of tens of bytes, and therefore does not pose a concern for overall system reliability, particularly since it can also be readily protected using ECC such as single error correction, double error detection which is readily available in modern FPGAs.

8 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.

© The Institution of Engineering and Technology 2018

We have proposed a GA 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 an RCN to provide random access to component voters can benefit from using variable-rate scheduling to prioritise 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 400% compared with checking voters in a round-robin manner. The results also show that the MTTFs of TMR–MER systems employing VRVC are greater than those that employ a VSE [10] to choose the next component to check at run time. Moreover, we have shown that the power consumption of TMR–MER systems can be significantly reduced by reducing the clock frequency of the RC without compromising system reliability. Finally, through fault injection testing, we have demonstrated that the MTTD errors can be reduced by 44 and 30% on average, respectively, when VRVC is used instead of round robin and VSE.

Our future work will consider adaptive scheduling of voter checks based on the radiation level of the surrounding environment with the aim of optimising the system reliability and power consumption. We also intend to consider including a user-defined metric (e.g. the criticality level of each TMR component) in our reliability models since some components such as clock managers are more critical than others despite being small in terms of their essential bit count.

9 Acknowledgment

This research was supported in part by the Australian Research Council's Discovery (DP150103866) Project funding scheme.

10 References

[1] Carmichael, C., Caffrey, M., Salazar, A.: 'Correcting single event upsets through Virtex partial configuration', Xilinx XAPP216, 2000

[2] Siegle, F., Vladimirova, T., Ilstad, J., *et al.*: 'Mitigation of radiation effects in SRAM-based FPGAs for space applications', *ACM Comput. Surv. (CSUR)*, 2015, **47**, (2), pp. 1–34

[3] von Neumann, J.: 'Probabilistic logics and the synthesis of reliable organisms from unreliable components', in Shannon, C.E., McCathy, J. (Eds.): '*Automata studies (annals of math studies no 34)*' (Princeton University Press, Princeton, New Jersey, 1956), pp. 43–98

[4] Herrera Alzu, I., Lopez Vallejo, M.: 'Design techniques for Xilinx Virtex FPGA configuration memory scrubbers', *IEEE Trans. Nucl. Sci.*, 2013, **60**, (1), pp. 376–385

[5] Heiner, J., Sellers, B., Wirthlin, M., *et al.*: 'FPGA partial reconfiguration via configuration scrubbing'. Int. Conf. Field Programmable Logic and Applications (FPL), Prague, Czech Republic, 2009, pp. 99–104

[6] Bolchini, C., Miele, A., Sandionigi, C.: 'A novel design methodology for implementing reliability-aware systems on SRAM-based FPGAs', *IEEE Trans. Comput.*, 2011, **60**, (12), pp. 1744–1758

[7] Tonfat, J., Kastensmidt, F.L., Rech, P., *et al.*: 'Analyzing the effectiveness of a frame-level redundancy scrubbing technique for SRAM-based FPGAs', *IEEE Trans. Nucl. Sci.*, 2015, **62**, (6), pp. 3080–3087

[8] Agiakatsikas, D., Cetin, E., Diessel, O.: 'FMER: a hybrid configuration memory error recovery scheme for highly reliable FPGA SoCs'. Int. Conf. Field Programmable Logic and Applications (FPL), Lausanne, Switzerland, 2016, pp. 88–91

[9] Agiakatsikas, D., Nguyen, N.T.H., Zhao, Z., *et al.*: 'Reconfiguration control networks for TMR systems with module-based recovery'. IEEE Int. Symp. Field-Programmable Custom Computing Machines (FCCM), Washington DC, USA, 2016, pp. 88–91

[10] Nguyen, N.T.H., Cetin, E., Diessel, O.: 'Dynamic scheduling of voter checks in FPGA-based TMR systems'. IEEE Int. Conf. Field-Programmable Technology (FPT), Xi'an, China, 2016, pp. 169–172

[11] Le, R.: 'Soft error mitigation using prioritized essential bits', Xilinx XAPP538 (v10), 2012

[12] Ostler, P.S., Caffrey, M.P., Gibelyou, D.S., *et al.*: 'SRAM FPGA reliability analysis for harsh radiation environments', *IEEE Trans. Nucl. Sci.*, 2009, **56**, (6), pp. 3519–3526

[13] Nguyen, N.T.H., Cetin, E., Diessel, O.: 'Scheduling considerations for voter checking in TMR–MER systems'. 2017 IEEE 25th Annual Int. Symp. Field-Programmable Custom Computing Machines (FCCM), Napa, CA, USA, 2017. p. 30

[14] Nguyen, N.T.H., Cetin, E., Diessel, O.: 'Scheduling voter checks to detect configuration memory errors in FPGA-based TMR systems'. Int. Symp. Defect and Fault Tolerance In VLSI Systems, Cambridge, UK, 2017

[15] UG909: Vivado Design Suite User Guide – Partial Reconfiguration (v20161), 6 April 2016

[16] Cetin, E., Diessel, O.: 'Guaranteed fault recovery time for FPGA-based TMR circuits employing partial reconfiguration'. Int. Workshop Computing in Heterogeneous, Autonomous 'N' Goal-oriented Environments, San Francisco, California, USA, 2012

[17] Veljkovic, F., Riesgo, T., de la Torre, E.: 'Adaptive reconfigurable voting for enhanced reliability in medium-grained fault tolerant architectures'. NASA/ESA Conf. Adaptive Hardware and Systems (AHS), Montreal QC, Canada, 2015, pp. 1–8

[18] Straka, M., Kastil, J., Kotasek, Z., *et al.*: 'Fault tolerant system design and SEU injection based testing', *Microprocess. Microsyst.*, 2013, **37**, (2), pp. 155–173

[19] Lee, J.Y., Chang, C.R., Jing, N., *et al.*: 'Heterogeneous configuration memory scrubbing for soft error mitigation in FPGAs'. Int. Conf. Field-Programmable Technology (FPT), Seoul, Korea, 2012, pp. 23–28

[20] Asadi, G.H., Tahoori, M.B.: 'Soft error mitigation for SRAM-based FPGAs'. 23rd IEEE VLSI Test Symp. (VTS'05), Palm Springs, California, USA, 2005, pp. 207–212

[21] Sari, A., Psarakis, M.: 'Scrubbing-based SEU mitigation approach for systems-on-programmable-chips'. Int. Conf. Field-Programmable Technology (FPT), New Delhi, India, 2011, pp. 1–8

[22] Nazar, G.L., Santos, L.P., Carro, L.: 'Fine-grained fast field-programmable gate array scrubbing', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2015, **23**, (5), pp. 893–904

[23] Santos, R., Venkataraman, S., Kumar, A.: 'Scrubbing mechanism for heterogeneous applications in reconfigurable devices', *ACM Trans. Des. Autom. Electron. Syst.*, 2017, **22**, (2), pp. 33:1–33:26

[24] Heron, O., Arnaout, T., Wunderlich, H.J.: 'On the reliability evaluation of SRAM-based FPGA designs'. Int. Conf. Field Programmable Logic and Applications (FPL), Tampere, Finland, 2005, pp. 403–408

[25] Edmonds, L.D.: 'Analysis of single-event upset rates in triple-modular redundancy devices'. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2009

[26] Koren, I., Krishna, C.M.: '*Fault-tolerant systems*' (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007)

[27] Nguyen, N.T.H.: 'Repairing FPGA configuration memory errors using dynamic partial reconfiguration', PhD thesis, November 2017. Available at http://handle.unsw.edu.au/1959.4/58878, accessed 12 March 2018

[28] Garcia Villoria, A., Pastor, R.: 'Solving the response time variability problem by means of a genetic algorithm', *Eur. J. Oper. Res.*, 2010, **202**, (2), pp. 320–327

[29] Gen, M., Cheng, R.: 'Genetic Algorithms and Engineering Design', 1997

[30] Goldberg, D., Deb, K., Korb, B.: 'Messy genetic algorithms: motivation, analysis, and first results', *Complex Syst.*, 1989, **3**, pp. 493–530

[31] Corominas, A., Kubiak, W., Palli, N.M.: 'Response time variability', *J. Sched.*, 2007, **10**, (2), pp. 97–110

[32] Allen, G.: 'Mitigation selection and qualification recommendations for Xilinx Virtex, Virtex-II, and Virtex-4 Field Programmable Gate Arrays', 2009

[33] Back, T., Fogel, D.B., Michalewicz, Z. (Eds.): '*Handbook of evolutionary Computation*' (IOP Publishing Ltd., Bristol, UK, 1997, 1st edn.)

[34] Eiben, Á.E., Hinterding, R., Michalewicz, Z.: 'Parameter control in evolutionary algorithms', *IEEE Trans. Evol. Comput.*, 1999, **3**, (2), pp. 124–141

[35] Cetin, E., Diessel, O., Li, T., *et al.*: 'Overview and investigation of SEU detection and recovery approaches for FPGA-based heterogeneous systems', in Kastensmidt, F., Rech, P. (Eds.): *FPGAs and Parallel Architectures for Aerospace Applications*, (Springer, Cham, 2016), pp. 33–46

[36] Zhao, Z., Nguyen, N.T., Agiakatsikas, D., *et al.*: 'Fine-grained module-based error recovery in FPGA-based TMR systems', *ACM Trans. Reconfigurable Technol. Syst. (TRETS)*, 2018, **11**, (1), p. 4

[37] https://www.qb50.eu/, accessed 12 March 2018

[38] McMurtrey, D., Morgan, K.S., Pratt, B., *et al.*: 'Estimating TMR reliability on FPGAs using Markov models', 2008

© The Institution of Engineering and Technology 2018