HAFTA: Highly Available Fault-Tolerant Architecture to Protect SRAM-Based Reconfigurable Devices against Multiple Bit Upsets

Zana Ghaderi, Seyed Ghassem Miremadi, *Senior Member, IEEE*,
Hossein Asadi, *Member, IEEE*, and Mahdi Fazeli

***Abstract*—Despite widespread use of *SRAM-based Reconfigurable Devices* (SRDs) in mainstream applications, their usage has been very limited in enterprise and safety-critical applications due to SRAM susceptibility to soft errors. Previous mitigation techniques to protect SRDs impose significant area and power overheads. Additionally, they suffer from susceptibility of configuration bits to *Multiple Bit Upsets* (MBUs). In this paper, we present a highly available fault-tolerant architecture to protect SRD-based designs against MBUs in both configuration and user bits. In the proposed architecture, the entire design is duplicated with respect to the relative locations of logic blocks within the SRD device and the main and replica *Flip-Flops* (FFs) are compared at each clock cycle to detect any possible mismatch. In addition, the unused FFs available throughout SRDs are employed as history FFs to save the latest correct state of the system. Upon detection of any mismatch between the main and replica FFs, the system is able to roll back to the latest correct state stored in the history FFs. The simulation results extracted using fault injection experiments demonstrate that the proposed architecture provides both higher reliability and availability as compared to the traditional TMR techniques while offering less area and power overheads.**

Keywords: SRAM-Based Reconfigurable Devices, Reliability, Availability, Multiple Bit Upsets

I. Introduction

SRAM-based Reconfigurable Devices (SRDs) are widely used in embedded and high-performance applications due to quick design upgrade, low cost, high performance, less error-prone design process, and fast time-to-market. Such devices can be quickly reconfigured an unlimited number of times to either upgrade design functionality or to re-implement a new design [1], [2]. In addition to general-purpose reconfigurable functional units, state-of-the-art SRDs include hard logic cores such as multipliers, RAM blocks, analog/digital interfacing, and even processing elements, making such devices a ready-to-use platform to implement industrial-scale designs.

Although SRDs offer such appealing features, their susceptibility to radiation-induced soft errors limits their usage in enterprise and safety-critical applications. In the previous technology generations, radiation-induced soft errors mostly caused *Single Event Transients* (SETs) or *Single Event Upsets* (SEUs), but in the current and emerging technologies (45nm and beyond), high energetic particles can result in *Single Event Multiple Transients* (SEMTs) or Single Event Multiple Upsets (SEMUs), which can significantly increase the probability of system failure. An energetic particle hitting hard logic cores in SRDs can result in either SETs or SEMTs while it can produce SEUs or SEMUs when affecting configuration bits or *Flip-Flops* (FFs). While the effect of energetic particles in hard logic cores and faulty FFs can be overwritten in the next clock cycles by employing appropriate redundancy techniques, SEUs or SEMUs in configuration bits have permanent effect unless they are reconfigured by the correct configuration bitstream. The permanent effect of soft errors in configuration bits is particularly more pronounced in high-end SRDs where the number of configuration bits is two orders of magnitude greater than the number of FFs or user bits [1], [2].

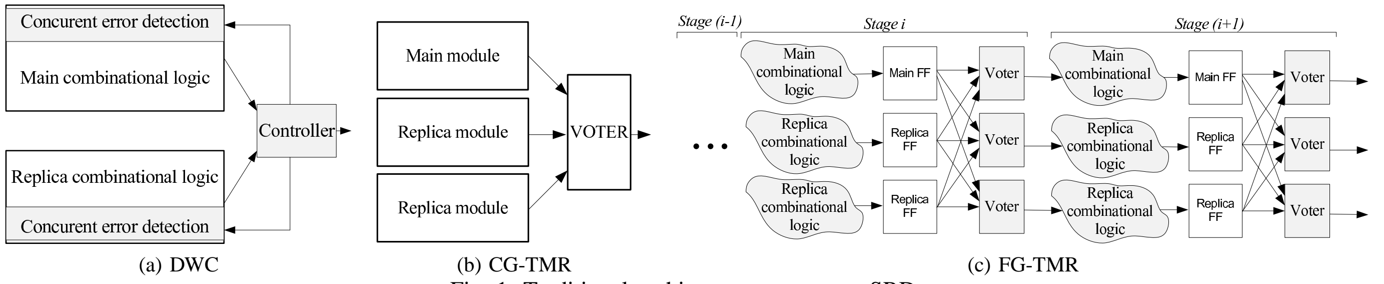
Manuscript received July 21, 2012; revised October 6, 2012; accepted November 5, 2012; All authors are with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, e-mails: xana@ce.sharif.edu, miremadi@sharif.edu, asadi@sharif.edu, m_fazeli@ce.sharif.edu.

Several architectures have been proposed to protect SRD-based designs using either *spatial*, *temporal*, or *information* redundancy techniques [3]–[8]. Architectures employing spatial redundancy techniques such as *Duplication with Comparison* (DWC) or *Triple Modular Redundancy* (TMR) have been extensively explored in the previous work [4]–[8]. Such architectures execute an application on two or multiple redundant modules. The DWC architecture, shown in Fig. 1a, can detect any mismatch by comparing the outputs of two modules. This architecture imposes less area overhead compared to other spatial redundancy techniques, however, no recovery mechanism is offered once an error is detected. To mitigate this shortcoming, concurrent error detection scheme can be used [9] but this scheme can be employed only in designs with regular datapaths where parity or *Error Correction Codes* (ECC) are applicable.

Another commonly used technique employing spatial redundancy is TMR. The traditional TMR architecture, shown in Fig. 1b, employs three redundant modules and a voter to compare the module outputs. This architecture, also called *Coarse-Grained TMR* (CG-TMR), can mask the effect of a SET or SEU in configuration or user bits within either of the redundant modules. CG-TMR, however, comes with several major shortcomings which limits its applicability in enterprise and safety-critical applications. First, CG-TMR imposes significant area and power overheads compared to an unprotected design [2], [3]. Second, CG-TMR is significantly susceptible to SETs or SEUs affecting configuration or user bits in the voter circuitry as well as SEMTs or SEMUs occurring in multiple modules. Although the probability of having two simultaneous particles hitting two redundant modules is very low, but a radiation-induced SEU may remain latent in one of the modules creating an opportunity for the second particle to cause a system failure when the system is running for a long period of time. Such error mechanism, also called *latent error*, can affect configuration bits as well as FFs.

To mitigate the effect of latent errors in TMR-based architectures, voter insertion techniques have been proposed, where a voter is inserted at the output of each FF and its corresponding replicas [8], [10], [11]. TMR-based architectures equipped with voter insertion techniques are called *Fine-Grained TMR* (FG-TMR) as apposed to CG-TMR [12]. FG-TMR, shown in Fig. 1c, can effectively mitigate the effect of latent errors in user bits by overwriting error-free values in the next clock cycles. Despite the major advantage of FG-TMR over CG-TMR, it suffers from several shortcomings. First, although FG-TMR is able to correct faulty user bits, it is unable to correct an erroneous configuration bit. Hence, subsequent soft errors can accumulate in the configuration bits causing a TMR failure in the voters. Second, inclusion of a voter for a pair of main and replica FFs significantly increases both area and power overheads compared to CG-TMR. Third, voter insertion will influence the system performance due to delay incurred to the critical path by the voter circuitry. Forth, the combinational and configuration bits used in the additional voters further increase the system susceptibility to both SETs and SEUs. Lastly, configuration bits and FFs in FG-TMR are significantly susceptible to SEMUs as they are typically located nearby in the physical layout of the SRD device.

In this paper, we present a *Highly Available, Fault-Tolerant Archi-*



(a) DWC (b) CG-TMR (c) FG-TMR

Fig. 1: Traditional architectures to protect SRDs

tecture (HAFTA) to protect both configuration and user bits against *Multiple Bit Upsets* (MBUs) with low area and power overheads. The main aim of HAFTA is to detect and correct a single and multiple faulty configuration or user bit(s) in SRD-based designs before the system outputs are affected. To this end, the entire design is duplicated with respect to the relative locations of logic blocks in the SRD device and the main FFs and replica FFs are compared at each clock cycle to detect any possible mismatch. In order to fast recover the system state, we employ unused FFs available throughout the SRD device as *History FFs* (HFFs) to store the correct state of the design at each clock cycle. By investigating the micro-architecture of state-of-the-art SRD devices, unused FFs can be efficiently allocated as HFFs such that area and power overheads are minimized. Upon detection of any mismatch in the user bits, the system is recovered in one clock cycle. In case of faulty configuration bits, the system state is restored using partial reconfiguration.

To evaluate the efficiency of the proposed architecture, both HAFTA and conventional architectures (FG-TMR and CG-TMR) have been implemented on a state-of-the-art SRD device. To investigate the reliability of the proposed architecture, we have conducted comprehensive fault injection experiments on ISCAS89 benchmarks [13] implemented on a SRD device. The fault injection results reveal that HAFTA is able to detect and correct 100% of both SEUs and SEMUs affecting user and configuration bits.

We also propose an analytical study to examine the availability of the proposed architecture as compared to the availability of FG-TMR and CG-TMR. The analytical results reveal that HAFTA provides at least two orders of magnitude higher availability compared to FG-TMR and CG-TMR. The higher reliability and availability is achieved while HAFTA imposes less area and power overheads compared to both CG-TMR and FG-TMR. The experimental results over ISCAS89 circuits reveal that HAFTA, CG-TMR, and FG-TMR, on average, impose 175%, 251%, and 338% area overhead as compared to unprotected designs, respectively.

In the rest of this paper, susceptibility of SRD devices to radiation-induced soft errors is first discussed in Sec. II. Related work is elaborated in Sec. III. The proposed architecture is presented in Sec. IV. Experimental setup and results are reported in Sec. V. Lastly, Sec. VI concludes the paper.

II. SRDs Susceptibility to Soft Errors

A SRD device is a two-dimensional array of logic blocks, connection blocks, and switch boxes. State-of-the-art SRDs typically have similar hierarchical architecture as shown in Fig. 2. In this figure, the largest computation granularity is a tile, which includes a group of clusters. A cluster, in turn, composed of few *Logic Elements* (LEs). Lastly, a LE consists of a *Look-Up Table* (LUT), a multiplexer, and one or two FFs. Two major components of device interconnect are *Connection Blocks* (CB) and switch boxes. Connection blocks are used to route signals from/to a logic block to global interconnect while switch boxes route signals between horizontal and vertical routing wires in the global interconnect.

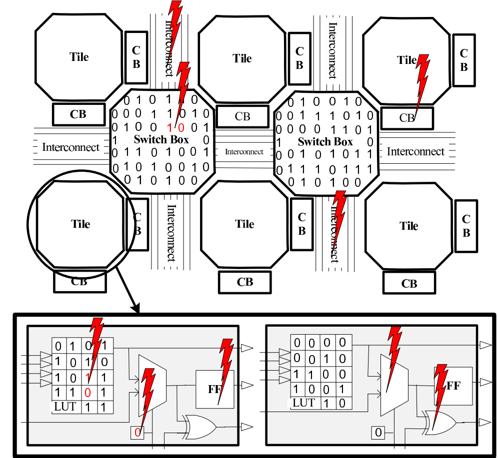


Fig. 2: Configuration and user bits in SRD devices

As shown in Fig. 2, an energetic particle can strike any part of a SRD device including FFs, LUTs, hard logic cores (e.g., logic gates), and configuration SRAM cells used in either global interconnect or multiplexer select bits. The effect of energetic particles in SRD devices can be categorized into *transient* and *permanent* errors. Particles affecting hard logic cores and FFs have transient effect as the corresponding impact can be overwritten in the subsequent clock cycles. A straightforward solution to mitigate the effect of such errors is temporal redundancy techniques. As an example, by re-executing an application running on a SRD device, the effect of such transient errors can be completely mitigated. The effect of energetic particles in SRAM configuration bits is, however, permanent and it cannot be mitigated by conventional temporal redundancy techniques. An erroneous configuration bit remains permanent in the device unless it is reconfigured by the original bitstream.

Permanent and transient effect of energetic particles can be further classified into single and multiple upsets. In the current technologies (45nm and beyond), an energetic particle can deposit charge in multiple logic gates or multiple sequential elements (FFs and SRAM cells) leading to SEMTs or SEMUs. Examples of SEMUs in a SRD device, as shown in Fig. 2, are multiple erroneous LUT bits, FFs, or routing configuration bits. Energetic particles hitting combinational logic can similarly lead to SEMTs in hard logic cores. Note either SEMUs or SEMTs can affect sequential or combinational elements, which are physically located nearby in the device physical layout. Since the probability of two simultaneous particles hitting two circuit nodes in a short time interval is extremely low, we ignore this effect in the rest of this paper.

III. Related Work

Briefly, there are two main approaches to implement robust SRD-based designs against soft errors including SEU mitigation and SEU detection/correction techniques. In SEU mitigation techniques, the *Soft Error Rate* (SER) of a SRD is alleviated using radiation hardened

by design techniques [14], or reliability oriented place and route which leads to less vulnerable configuration bits [15]–[22]. Radiation hardened by design techniques [14] are efficient, however, they cannot be applied in applications where the cost is a major concern. Using reliability oriented place and route results in area and power efficient design but it just reduces the soft error rate and cannot completely remove the effect of SEUs. Most of the reliability oriented place and route techniques such as [16], [18]–[22] do not consider protecting interconnect configuration bits. This has significantly limited the efficiency of such techniques. In addition, the efficiency of such SER mitigation techniques highly depends on the structure and functionality of target circuits. This further limits the amount of SER reduction provided by the mentioned techniques. [15], [17] have proposed a reliability oriented place and route technique to protect the interconnect configuration bits. However, the proposed technique in [15] only covers bridging faults and the other fault models such as short circuit or open fault are not studied. Similarly, [17] has focused on faults in unidirectional routing in MUX-based SRDs without taking into account sequential feedbacks and persistent errors.

There are numerous SEU detection/correction techniques in the literature [2]–[12], [23]–[31]. SEU detection/correction techniques in SRD-based design can be further categorized into two different categories including configuration scrubbing and redundancy-based techniques. In configuration scrubbing techniques, faults are detected by periodically reading configuration bitstream and rewriting faulty configuration bits with uncorrupted configuration data stored in an off-chip memory [2], [3]. A fault in configuration bits can be detected either by comparing configuration bitstream by the stored fault free one or by using error detection codes such as *Cyclic Redundancy Check* (CRC) over configuration bitstream. Most high-end SRDs support dedicated circuitry to perform high-speed scrubbing over configuration bits. Although scrubbing can effectively detect and correct single and multiple erroneous configuration bits, detection and correction latency can significantly affect both the system reliability and availability. In safety critical applications, an erroneous configuration bit can cause catastrophic effects before it is detected by the scrubber circuitry. Scrubbing is not also able to protect user bits. As such, scrubbing techniques are typically employed along with spatial redundancy techniques such as FG-TMR or CG-TMR. Additionally, scrubbing techniques are not able to distinguish faults occurring in *care bits* from those occurring in *don't care* bits. This will affect the overall system availability due to numerous false reconfigurations.

Redundancy-based techniques employ different forms of redundancies including spatial, temporal, or information redundancy techniques to detect or correct soft errors. Architectures employing spatial redundancy techniques such as *Duplication with Comparison* (DWC) [9], [27], [30] or *Triple Modular Redundancy* (TMR) [5]–[8], [10]–[12], [23]–[26], [32] have been extensively explored in the previous work. In addition to the major shortcomings of the TMR techniques discussed in Sec. I, regardless of the TMR granularity, i.e. CG-TMR or FG-TMR, some faults may not be masked by the voter due to the use of common resources such as routing as well as bridging faults in module replicas. In [26], it has been reported that about 13% of SEUs may escape from the voter module in a TMR-based architecture. Obviously, the situation gets worse for SEMU faults. [6] has reported that the failure rate due to 2-bit SEMU is 2.6 orders of magnitude greater than that of due to SEUs in a TMR protected circuits. To mitigate this problem, fault-tolerant place and route techniques such as RoRA are exploited along with TMR architectures [23], [32], [33]. These techniques further increase area, power, and performance overheads in the TMR architecture.

To reduce the overhead imposed by TMR techniques, some techniques such as *Selective TMR* (STMR) or *Partial TMR* (PTMR) have been proposed [8], [11], [24]. In these techniques, instead of applying TMR to the entire circuit, only the most critical parts of the design are protected. In particular, critical parts are defined using signal probabilities in STMR to protect the combinational logic, while in PTMR, the vulnerable parts are those having most contribution in causing persistent errors. A persistent error refers to an error occurring in configuration bits and captured in flip-flops and remains in the circuit even after the reconfiguration. The major source of such errors is the presence of feedback paths as it is detailed in [24].

Several techniques have also proposed to use *Duplication With Comparison* (DWC). As discussed earlier in Sec. I, this architecture imposes less area overhead compared to TMR techniques, however, no recovery mechanism is offered once an error is detected. Several DWC techniques such as [27], [30] have addressed this issue, but they are either applicable to some specific circuits [27] or suitable for rather small circuits [30]. [30] has proposed to extract the FSM of a design and map the original FSM as well as its duplicated version to a block of RAM available in SRDs (referred as *Dual Modular Redundancy* or DMR). The faults are detected by comparing the replicas and the faulty module is identified by using parity codes for each module. This technique is able to protect user bits (FFs), however, it is not scalable and cannot be applied to large circuits having a huge FSM. [27] has proposed an enhanced DMR technique at the system level using a smart voter to detect faulty modules. Similar to TMR, this technique suffers from faults occurring in the common resources. In addition, the smart voter significantly limits the overall reliability as it can become a source of single point of failure.

IV. Proposed Architecture

The key idea behind the proposed fault-tolerant architecture is the joint use of spatial and time redundancy techniques for detecting and correcting faults in both user and configuration bits. The block diagram of the proposed architecture has been depicted in Fig. 3. As shown in this figure, the main module is duplicated and the values of FFs in the main module are compared with the corresponding FFs in the replica module at each clock cycle. Each FF in both main and replica modules has a history FF (shown as HFF and replica HFF in Fig. 3) that holds the values of the main and replica FFs stored in the previous clock cycle. The controller in the proposed architecture compares the outputs of main FFs and replica FFs at each clock cycle and specifies whether the circuit should continue its normal operation or should roll back to its previous state stored in HFFs.

Upon detection of a fault by the controller, the whole circuit rolls back to the state stored in the previous clock cycle by copying the values of HFFs and replica HFFs into main and replica FFs, respectively. The contents of user bits in the normal operation and in the error recovery mode have been shown in Fig. 3 for few sample clock cycles. The controller can switch between the normal operation and the error recovery mode using the multiplexers shown in Fig. 3 (e.g., MUX_i and $Replica\ MUX_i$).

By rolling back the state of the circuit to its previous state, soft errors affecting hard logic or FFs will be overwritten in the subsequent clock cycle. However, soft errors in SRAM configuration bits would remain in the system and would again lead to result mismatches in the subsequent clock cycles. To cope with the effect of soft errors in configuration bits, the controller examines the number of consecutive mismatches in a timing window. When the controller detects a number of mismatches greater than a specified threshold in a given time interval, the faults are identified as permanent and as a result, a partial reconfiguration is conducted by the controller. While performing partial reconfiguration, the system state is preserved in both HFFs and replica HFFs. Once the reconfiguration is completed, the correct system state is transferred from HFFs and replica HFFs to

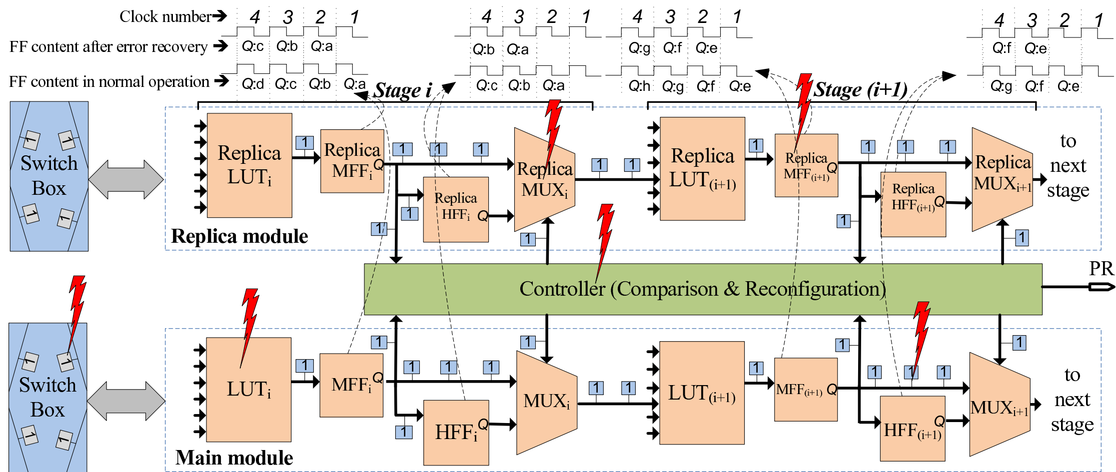


Fig. 3: Proposed architecture

main and replica FFs, respectively, and the system normal operation is then resumed.

A. Controller Logic

The controller logic, shown in Fig. 4, consists of a comparator, an OR tree, a counter, and two outputs. The comparator and the OR tree circuitry is responsible to detect possible mismatch between main and replica modules and the corresponding output signal, Sel_{mux} is routed to MUX and replica MUX selectors. The counter is used to distinguish between transient and permanent effect of energetic particles. A SEU or SEMU in main or replica FFs will cause a mismatch and as a result, the counter is incremented by one. When the comparator detects mismatches in consecutive clock cycles that is greater than a specified threshold, the fault is considered as a permanent fault. Once a fault is identified as permanent, the other controller output, PR, is activated to perform a partial reconfiguration. Upon completion of partial reconfiguration, the system normal operation is resumed and the counter is set to zero.

The comparator and the OR tree circuitry, shown in Fig. 4, can be implemented using either LUTs or hard logic cores available in SRD devices. To avoid the permanent effect of energetic particles in the controller, hard logic cores such as XOR and OR gates available in SRD devices can be used to implement the comparator and the OR tree circuitry. The effect of energetic particles in the controller will be discussed in the next subsection.

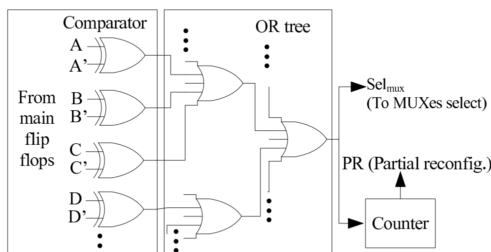


Fig. 4: Controller logic

B. Fault Tolerance Analysis

The proposed architecture has been designed to effectively mitigate the effect of SETs, SEMTs, SEUs, or SEMUs in any of SRD resources and quickly recover the system state. In the following, we discuss the effect of soft errors in user bits, configuration bits, clock signals, and controller circuitry.

1) Soft Errors in User Bits: A soft error in the main FFs or replica FFs will be immediately detected by the controller and will be overwritten in the next clock cycle by the system state stored in HFFs and replica HFFs. As shown in Fig. 3, a soft error in MFF_i or replica MFF_i will only delay the system state by one clock cycle. A soft error in HFFs is not important as it will be automatically overwritten in the next clock cycle.

A soft error affecting multiple user bits will be also lead to result mismatch and will be corrected in the next clock cycle. There is only one situation that HAFTA is able to detect but not to correct soft errors affecting multiple FFs. This is when a particle affects a FF and its corresponding HFF at the same time. In this case, a mismatch is detected by the controller and a similar mismatch will be repeated in the subsequent clock cycles. Although the probability of a particle to simultaneously affect both main FF and its corresponding HFF is very low, placement algorithms can be further enhanced to eliminate the effect of such errors. This will be detailed in Sec. IV-D.

2) Soft Errors in Configuration Bits: A particle affecting multiple LUT bits (or configuration bits in either multiplexers or global routing) will eventually lead to a soft error in one or multiple user bits. The placement of main and replica modules are done such way that no routing resources are shared between these two modules. As a result, multiple erroneous configuration bits will result in either single or multiple erroneous user bits in only one of the modules (either main module or replica module). Such erroneous user bits will be detected by the controller and identified as permanent faults and eventually will be corrected by a partial reconfiguration.

3) Soft Errors in Clock Signals: Soft errors in configuration bits used in clock circuitry is one of the issues in SRD-based fault-tolerant techniques [2]. The major effect of soft errors in clock circuitry is inhibiting a clock signal passing through a tile or a cluster. Once a clock signal is inhibited in a tile or a cluster, the state of FFs remains the same as the previous clock cycle and as a result, a mismatch is detected by the controller. As such, a partial reconfiguration is initiated which will correct erroneous configuration bits used in clock circuitry. Note soft errors typically do not affect wires used in clock routing since clock wires typically come with large capacitors which can mitigate the deposited charge caused by energetic particles.

4) Soft Errors in the Controller: Soft errors can affect the comparator, the OR tree, the counter, and routing signals within the controller. In case of SETs or SEMTs in the comparators or the OR tree, an error may be automatically masked or it may be propagated to PR and Sel_{mux} signals. In case of erroneous PR and Sel_{mux} signals, a false reconfiguration will be imposed to the system and a downtime will be incurred to the system. Since the reconfiguration time is orders of magnitude smaller than the time between two consecutive particle hits, this downtime is negligible as compared to the overall application running time. The availability results due to partial reconfiguration will be reported in Sec. V. A SEU or SEMU in the counter may also cause a false reconfiguration. Similar to discussion provided above,

the system unavailability for such false reconfiguration lasts only for very short period of time. Lastly, to further enhance the reliability of the controller, the controller circuitry can be reconfigured on a regular basis to avoid any catastrophic effect of erroneous configuration bits within the controller circuitry.

C. Availability Analysis

Here, we describe how to compute availability with respect to fault types described in Sec. II. To compute availability, we investigate different cases where a system is *up* or *down* causing the system to be *available* or *unavailable*, respectively. Such cases have been examined for both HAFTA and TMR architectures, as shown in Fig. 5. *Case A* through *Case D* investigates SEUs/SEMUs affecting a design protected by HAFTA. *Case A* in this figure demonstrates a SEU/SEMU affecting user bits. Since the recovery is achieved in one clock cycle, no downtime is incurred to the system. *Case B* depicts a situation where a SEU/SEMU affects the controller. If the error is not masked, a false reconfiguration is incurred to the system. *Case C* illustrates a SEU/SEMU affecting a *don't care* configuration bit. Here, no reconfiguration is imposed to the system and the system will continue its normal operation. *Case D* is the representative of SEUs/SEMUs affecting *care* configuration bits. Once consecutive mismatches are detected by the controller, a partial reconfiguration is initiated. While the partial reconfiguration is being performed, the system will be in the down state.

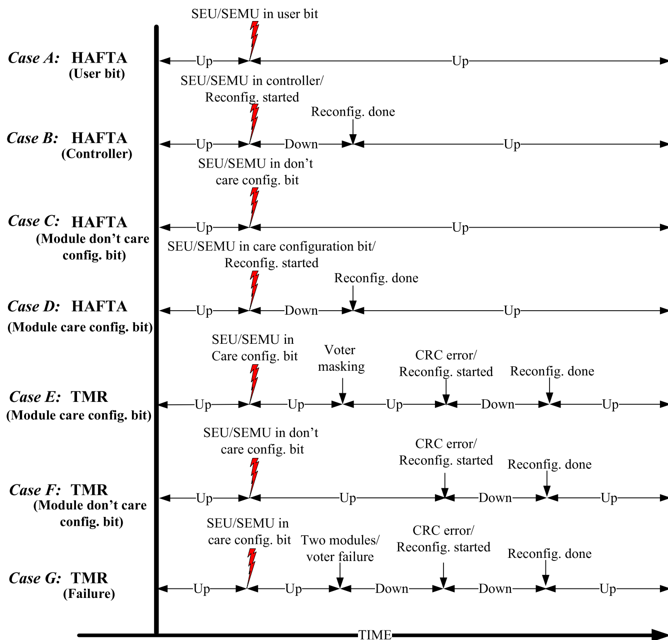


Fig. 5: System availability for different fault types/sites: HAFTA *vs.* TMR

The last three cases shown in Fig. 5 illustrate different situations where a SEU/SEMU affects a TMR system. In *Case E*, a SEU/SEMU affects care single or multiple configuration bits in one of the modules but erroneous values are masked by the voter(s). Once erroneous configuration bit(s) is detected by the CRC checker, a partial reconfiguration is conducted and the normal operation is then resumed. *Case F* depicts the situation, where a SEU/SEMU affects *don't care* configuration bits. Since there is no mechanism to distinguish between erroneous *don't care* configuration bits from others, upon detection of CRC error, the system normal operation is stalled and a partial reconfiguration is conducted. Lastly, *Case G* represents the case in which two modules or a voter is affected by a SEU/SEMU. In such situation, the system will be in the down state until an erroneous configuration bit is detected by the CRC checker and a partial reconfiguration is conducted.

D. Placement Algorithm

The main issue in HAFTA is placing history FFs in a way that they are close to the main FFs as much as possible to reduce power and area overheads incurred by long interconnect or wires. On the other hand, to remove the probability of multiple bit upsets in a main FF and its corresponding HFF, they should be placed apart from each other. Note SEMUs can only affect multiple FFs located nearby in the device physical layout. For the sake of simplicity, we will explain the overall algorithm for placing the unused FFs as history FFs using an example shown in Fig. 6. In this figure, the used resources are colored black and the unused resources are left as white. FFs used as HFFs are colored gray.

In the proposed placement algorithm, the best choice of placement for a HFF is when a FF and its corresponding HFF is placed in nearby LEs within one cluster. This has been shown in Fig. 6 by pair FFs 4 and 5. Such placement will minimize both power and area overheads while it also eliminates the probability of multiple bit upsets in a main FF and its corresponding HFF. In case, such unused FFs are not available within the cluster, unused FFs in the nearby clusters within the same tile will be the next candidate. This has been shown by pair FFs 6, 7, 8, and 9 in Fig. 6. Lastly, if unused FFs are not found in either the nearby LEs or clusters, unused FFs in the nearby tiles are selected as HFFs (as shown by pair FFs 2 and 3). In this case, the general interconnect through a switch-box is utilized.

E. Limitations of the Proposed Architecture

There are few major challenges threatening the efficiency of the proposed architecture. First, most SRD venders do not reveal layout-level device information. Instead, relative locations of logic blocks within SRDs are reported. This may limit the efficiency of the proposed placement algorithm presented in Sec. IV-D. Despite this limitation, a conservative placement of HFFs such as those denoted by pair FFs 6, 7, 8, and 9 in Fig. 6 can be used to ensure that a FF and its corresponding HFF is not located nearby in the physical layout. Several circuit-level experimental studies investigate SEMU adjacency neighborhood [34], [35]. For instance, it is observed in [34] that the cross section of SEMU in Virtex-IV devices is about 3.85×10^{-16} cm^2. Considering the size of a typical logic element in the same device, it can be concluded that the probability of having two bit-flips in two adjacent logic elements is extremely low. The area of a logic element can be roughly calculated using die size and the number of logic elements within a device. Further discussion of SEMU cross section in configuration and user bits has been provided in [34], [35].

Another issue that may limit the efficiency of the proposed architecture is when there is a feedback path from a FF input to its output through a combinational logic. Such circuit structure shown in Fig. 7a can result into persistent errors as discussed in [24], [31]. As demonstrated in Fig. 7a, the proposed architecture can detect result mismatch but before partial reconfiguration gets completed, the error can again propagate from the feedback path to the main FF. This results in another mismatch by the controller. Such error detection and correction can be continuously repeated within the circuit. To cope with such errors, we suggest to use an architecture shown in Fig. 7b for circuits with feedback paths. As shown in this figure, while partial reconfiguration is being performed, the output of HFFs is send out to both the next stage FF and the feedback path. As demonstrated by an example in Fig. 7b, this architecture can effectively cope with persistent errors in circuits with feedback paths.

Lastly, communication with external circuitry is another limitation of the proposed architecture when it is used in a larger system. In the

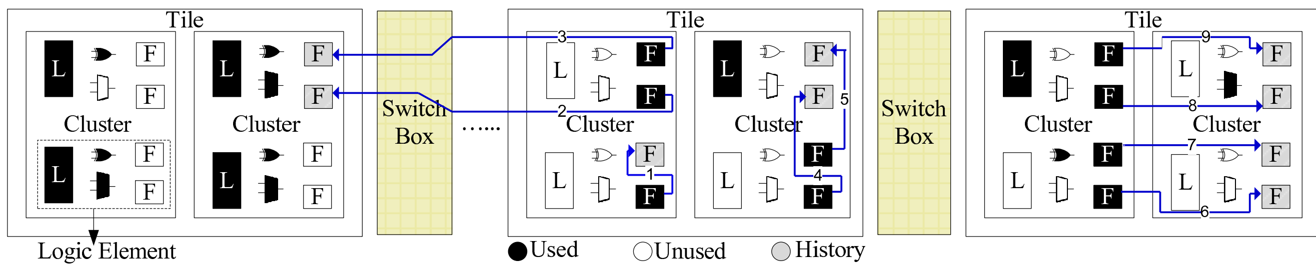
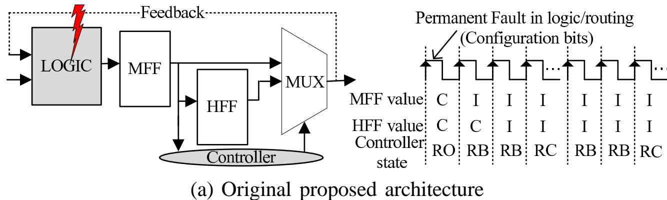
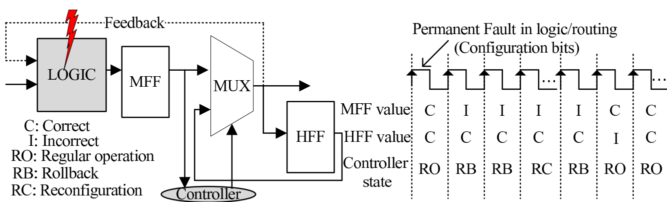


Fig. 6: Placement of HFFs (L:LUT, F:FF)



(a) Original proposed architecture



(b) Modified architecture to mitigate persistent errors

Fig. 7: Persistent errors in HAFTA

proposed architecture, the external circuitry should wait during system recovery until the correct system state is restored. While outgoing signals can be derived by the previous system state, the synchronization with incoming signals seems to be more challenging. One way to be properly synchronized with incoming signals is buffering input values and adding a *wait* signal to the design in order to halt the previous input state. Upon detection of any mismatch, the *wait* signal can be activated by the controller to stop receiving new inputs. In case, incoming signals are derived by real-time environmental sensors, this synchronization scheme cannot be utilized. Such environment sensors, however, do not have a significant variation during a short period of time while the circuit is being recovered. This issue is one of the main challenges in roll-back and roll-forward mechanism techniques and has been further addressed in the previous work [36], [37].

V. Experimental Setup and Results

To evaluate the reliability, availability, and power and area overheads of the proposed architecture, we have implemented ISCAS89 benchmarks on a Virtex6©(xc6vlx240t) Xilinx device [38] for unprotected design, FG-TMR, CG-TMR, and HAFTA. Layout-level device resources are extracted using the *Xilinx Design Language* (XDL) format of mapped designs. This information is then feed into our placement algorithm in order to properly add HFFs to the circuits. Once placement of HFFs are completed, the final routing is performed.

To examine the reliability of a design protected by HAFTA, we have employed a simulation based fault injection approach. Fault injection is performed on a simulation model of designs using ModelSim© toolset [39]. The simulation model of designs has been extracted from the XDL format such that it represents the layout-level resources available in the SRD device. We have carried out fault injection experiments on LUTs, FFs, and configuration bits used in either multiplexers or interconnects. The fault injection experiments have been performed with respect to the fault models described in Sec. II.

The results of fault injection experiments for both unprotected design and protected design by HAFTA, CG-TMR, and FG-TMR have been reported in Table I. In this table, we report the probability of system failure for different resources available in the SRD device with respect to SEU and SEMU fault models. Due to page limitation, the probability of system failure has been reported only for three circuits selected as representative of small (s349), medium (s953), and large (s5378) circuits from ISCAS89 benchmark suite. In our experiments, 80,000 faults have been injected for each circuit under study (240,000 faults in total). The fault injection experiments have been performed on eleven processing cores running over 24 hours. As the results demonstrate, the protected circuit by HAFTA can efficiently detect and correct 100% of both permanent and transient faults for both SEU and MBU fault models. The failure probabilities reported in this table have been computed with 95% confidence level. The sampling error is equal to $Z_{1-a/2}\sqrt{p(1-p)/n}$ [40]. In this equation, p and n are failure rate and the number of fault injections, respectively [40]. The other parameter ($Z_{1-a/2}$) is confidence level coefficient (in our study, $a = 95\%$ and $Z_{1-a/2} = 1.96$). As it was expected, both CG-TMR and FG-TMR can mitigate 100% of SEUs in FFs. However, both architectures are susceptible to SEUs and SEMUs affecting configuration bits.

To measure the area overhead, the percentage of additional used FFs, clusters, and tiles as compared to the unprotected design has been reported in Table II. As reported in this table, HAFTA uses more FFs but imposes less area overhead as compared to CG-TMR and FG-TMR. The results demonstrate that HAFTA, CG-TMR, and FG-TMR, on average, impose 175%, 251%, 338% area overhead as compared to the unprotected design, respectively.

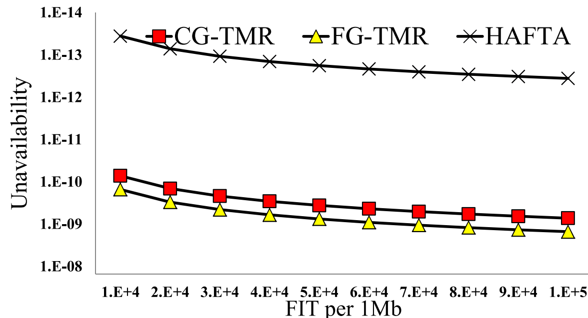
TABLE II: Percentage area overhead as compared to unprotected design (CG: CG-TMR, FG: FG-TMR, HAF: HAFTA)

	FF			TILE			CLUSTER		
Circuit	CG	FG	HAF	CG	FG	HAF	CG	FG	HAF
s400	200	200	310	350	400	225	317	367	333
s526	200	200	310	133	333	183	188	400	275
s641	200	200	312	259	188	94	291	264	109
s713	200	200	312	200	165	100	214	243	124
s838	200	200	306	185	292	254	206	371	300
s953	200	200	307	196	328	164	235	400	181
s1196	200	200	289	344	450	206	313	419	188
s1238	200	200	289	347	379	137	289	337	123
s1423	200	200	303	250	461	233	235	477	219
s5378	200	200	301	247	382	158	243	358	172
AVG.	200	200	304	**251**	**338**	**175**	253	364	202

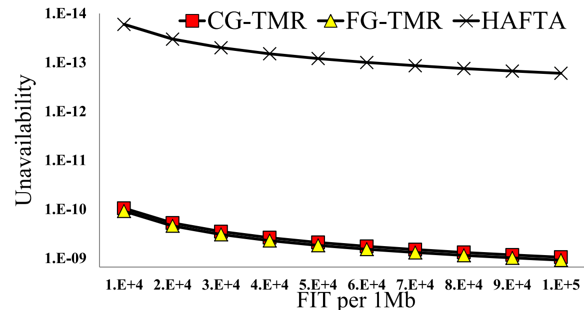
In Fig. 8, we report the availability of the proposed architecture for two sample circuits of ISCAS89 benchmark suite. The system unavailability has been reported under different FIT rates ranging from 10,000 to 100,000 FITs per Mbits. The typical FIT rates at the sea-level varies from 100 to 1000 FITs per Mbits but these numbers can be 10X (up to 100X) greater in high altitudes [41]. As shown in Fig. 8,

TABLE I: Probability of system failure: HAFTA *vs.* conventional architectures (# of fault injections for each circuit: 80,0000)

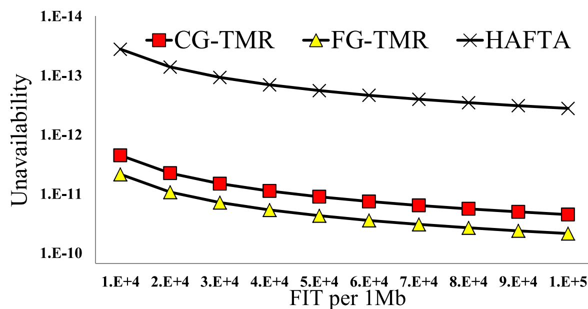
Circuit	Fault Type	Fault Site	Unprotected	CG-TMR	FG-TMR	HAFTA
s349	SEU	LUT	$0.0394_{\pm 0.0054}$	$0.0008_{\pm 0.0008}$	$0.0004_{\pm 0.0006}$	$0.0000_{\pm 0.0004}$
	SEU	NET	$0.5240_{\pm 0.0138}$	$0.0694_{\pm 0.0070}$	$0.0590_{\pm 0.0065}$	$0.0000_{\pm 0.0004}$
	SEU	FF	$0.0020_{\pm 0.0012}$	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$
	SEMU	LUT	$0.0712_{\pm 0.0071}$	$0.0014_{\pm 0.0010}$	$0.0010_{\pm 0.0009}$	$0.0000_{\pm 0.0004}$
s953	SEU	LUT	$0.0174_{\pm 0.0036}$	$0.0004_{\pm 0.0006}$	$0.0004_{\pm 0.0006}$	$0.0000_{\pm 0.0004}$
	SEU	NET	$0.3660_{\pm 0.0132}$	$0.1034_{\pm 0.0094}$	$0.0248_{\pm 0.0040}$	$0.0000_{\pm 0.0004}$
	SEU	FF	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$
	SEMU	LUT	$0.0497_{\pm 0.0060}$	$0.0005_{\pm 0.0006}$	$0.0006_{\pm 0.0007}$	$0.0000_{\pm 0.0004}$
s5378	SEU	LUT	$0.0152_{\pm 0.0034}$	$0.0002_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$
	SEU	NET	$0.3450_{\pm 0.0134}$	$0.1332_{\pm 0.0084}$	$0.0210_{\pm 0.0043}$	$0.0000_{\pm 0.0004}$
	SEU	FF	$0.0008_{\pm 0.0008}$	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$
	SEMU	LUT	$0.0190_{\pm 0.0038}$	$0.0000_{\pm 0.0004}$	$0.0002_{\pm 0.0004}$	$0.0000_{\pm 0.0004}$



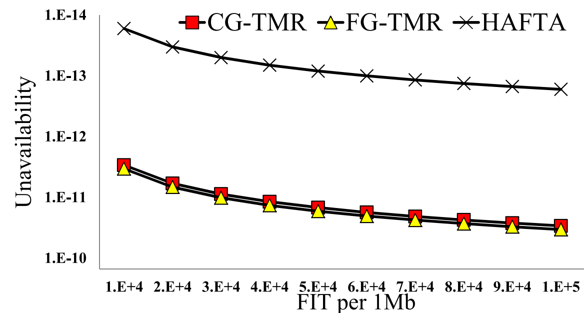
(a) s953 (CRC readback freq. = 2Mhz)



(b) s1238 (CRC readback freq. = 2Mhz)



(c) s953 (CRC readback freq. = 66Mhz)



(d) s1238 (CRC readback freq. = 66Mhz)

Fig. 8: Availability of HAFTA *vs.* FG-TMR and CG-TMR

the availability of the proposed architecture is at least two orders of magnitude higher compared to FG-TMR and CG-TMR. In the results reported in Fig. 8, we compute availability with two nominal CRC readback frequencies (2Mhz and 66Mhz). As can be seen in Fig. 8, FG-TMR and CG-TMR appear to be more sensitive to CRC frequency. This is due to configuration bits in FG-TMR and CG-TMR are mainly protected by the CRC readback scheme and, as a result, higher CRC frequencies can speed up detecting a permanent configuration bit and improve the overall availability.

To estimate the power consumption, we have used Xpower Analyzer© [42] presented by Xilinx. The dynamic power results are reported in Fig. 9. The dynamic power overhead of the proposed architecture as compared to the unprotected design is, on average, about 74% while CG-TMR and FG-TMR imposes more than 126% and 149% dynamic power overhead versus the unprotected design. The less power overhead of the proposed architecture is mainly due to HAFTA employs the wasted resources available throughout the SRD device. This significantly reduces the interconnect of clock signals that are the most active and power consumptive signals in sequential circuits.

To evaluate the performance overhead of the proposed architecture, the critical path delay of the studied circuits have been extracted for the target architectures. As shown in Fig. 10, HAFTA imposes 25% performance overhead as compared to unprotected designs while CG-TMR and FG-TMR impose 5.7% and 6.8% performance overhead as compared to unprotected designs, respectively. The higher performance overhead of HAFTA can be mainly related to the OR tree circuitry used in the controller.

VI. Conclusions

In this paper, we proposed a power-efficient fault-tolerant architecture for SRD-based designs. The proposed architecture can efficiently cope with SEMTs in hard-wired logic as well as SEMUs occurring in both configuration and user bits. The key idea behind the proposed technique is to keep the previous state of FFs using history FFs. To mitigate the effect of radiation-induced soft errors, we proposed to duplicate the entire circuit and compare the values of main FFs with their corresponding replica FFs. To minimize the area and power overheads in the proposed architecture, we also proposed an efficient algorithm to place HFFs in unused FFs. The simulation results through fault injection experiments on ISCAS89 benchmarks showed that the proposed architecture offers higher level of reliability and availability as compared to CG-TMR and FG-TMR while having less area and power overheads.

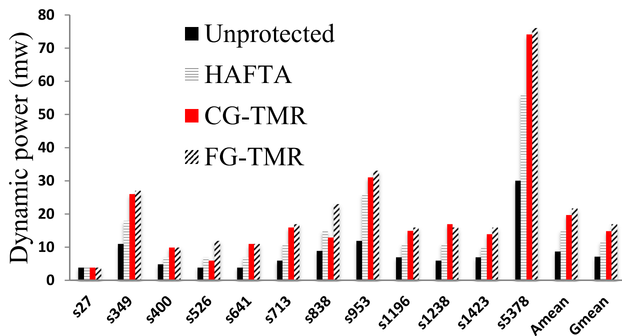


Fig. 9: Dynamic power

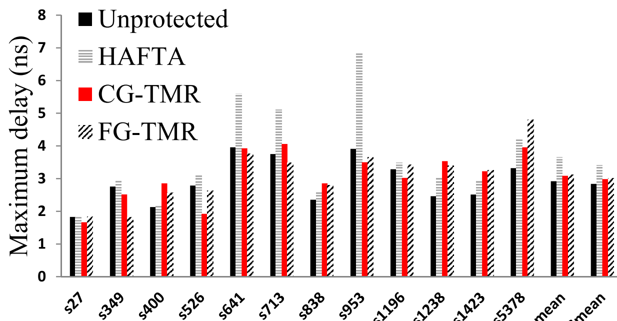


Fig. 10: Performance

REFERENCES

[1] H. Asadi, M.B. Tahoori, B. Mullins, D. Kaeli, and K. Granlund. Soft error susceptibility analysis of sram-based fpgas in high-performance information systems. *IEEE Transactions on Nuclear Science (TNS)*, 54(6):2714 – 2726, December 2007.

[2] H. Asadi and M.B. Tahoori. Analytical techniques for soft error rate modeling and mitigation of fpga-based designs. *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, 15(12):1320 – 1331, December 2007.

[3] S.P. Park, D. Lee, and K. Roy. Soft-error-resilient fpgas using built-in 2-d hamming product code. *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, 20(2):248 – 256, February 2012.

[4] X. She and K.S. McElvain. Time multiplexed triple modular redundancy for single event upset mitigation. *IEEE Transactions on Nuclear Science (TNS)*, 56(4):2443 – 2448, August 2009.

[5] F. Lahrach, A. Doumar, E. Chatelet, and A. Abdaoui. Master-slave tmr inspired technique for fault tolerance of sram-based fpga. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 58 – 62, July 2010.

[6] L. Sterpone, M. Violante, A. Panariti, A. Bocquillon, F. Miller, N. Buard, A. Manuzzato, S. Gerardin, and A. Paccagnella. Layout-aware multi-cell upsets effects analysis on tmr circuits implemented on sram-based fpgas. *IEEE Transactions on Nuclear Science (TNS)*, 58(5):2325 – 2332, October 2011.

[7] A. Manuzzato, S. Gerardin, A. Paccagnella, L. Sterpone, and M. Violante. Effectiveness of tmr-based techniques to mitigate alpha-induced seu accumulation in commercial sram-based fpgas. *IEEE Transactions on Nuclear Science (TNS)*, 55(4):1968 – 1973, August 2008.

[8] P.K. Samudrala, J. Ramos, and S. Katkoori. Selective triple modular redundancy (stmr) based single-event upset (seu) tolerant synthesis for fpgas. *IEEE Transactions on Nuclear Science (TNS)*, 51(5):2957 – 2969, October 2004.

[9] F. Lima, L. Carro, and R. Reis. Designing fault tolerant systems into sram-based fpgas. In *Design Automation Conference (DAC)*, pages 650 – 655, June 2003.

[10] J.M. Johnson and M.J. Wirthlin. Voter insertion algorithms for fpga designs using triple modular redundancy. In *Field programmable gate arrays (FPGA)*, pages 249 – 258, February 2010.

[11] O. Ruano, J.A. Maestro, and P. Reviriego. A methodology for automatic insertion of selective tmr in digital circuits affected by seus. *IEEE Transactions on Nuclear Science (TNS)*, 56(4):2091 – 2102, August 2009.

[12] M. Niknahad, O. Sander, and J. Becker. Fgtmr - fine grain redundancy method for reconfigurable architectures under high failure rates. In *North-East Asia Symposium on Nano, Information Technology and Reliability (NASNIT)*, pages 186 – 191, December 2011.

[13] F. Brglez, D. Bryan, and K. Koiminski. Combinational profiles of sequential benchmark circuits. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1929 – 1934, May 1989.

[14] B.S. Gill, C. Papachristou, and F.G. Wolff. A new asymmetric sram cell to reduce soft errors and leakage power in fpga. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1460 – 1465, April 2007.

[15] S. Golshan and E. Bozorgzadeh. Single-event-upset (seu) awareness in fpga routing. In *Design Automation Conference (DAC)*, pages 330 – 333, June 2007.

[16] J.Y. Lee, Y. Hu, R. Majumdar, L. He, and M. Li. Fault-tolerant resynthesis with dual-output luts. In *Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 325 – 330, January 2010.

[17] N. Jing, J. Lee, W. He, Z. Mao, and L. He. Mitigating fpga interconnect soft errors by in-place lut inversion. In *International Conference on Computer-Aided Design (ICCAD)*, pages 582 – 586, November 2011.

[18] M. Jose, Y. Hu, R. Majumdar, and L. He. Rewiring for robustness. In *Design Automation Conference (DAC)*, pages 469 – 474, June 2010.

[19] C. Peng, C. Dong, and D. Chen. Setmap: A soft error tolerant mapping algorithm for fpga designs with low power. In *Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 388 – 393, January 2011.

[20] Z. Feng, Y. Hu, L. He, and R. Majumdar. Ipr: In-place reconfiguration for fpga fault tolerance. In *International Conference on Computer-Aided Design (ICCAD)*, pages 105 – 108, November 2009.

[21] J. Lee, Z. Feng, and L. He. In-place decomposition for robustness in fpga. In *International Conference on Computer-Aided Design (ICCAD)*, pages 143 – 148, November 2010.

[22] Y. Hu, Z. Feng, L. He, and R. Majumdar. Robust fpga resynthesis based on fault-tolerant boolean matching. In *International Conference on Computer-Aided Design (ICCAD)*, pages 706 – 713, November 2008.

[23] L. Sterpone and M. Violante. A new reliability-oriented place and route algorithm for sram-based fpgas. *IEEE Transactions on Computers (TC)*, 55(6):732 – 744, June 2006.

[24] B. Pratt, M. Caffrey, J.F. Carroll, P. Graham, K. Morgan, and M. Wirthlin. Fine-grain seu mitigation for fpgas using partial tmr. *IEEE Transactions on Nuclear Science (TNS)*, 55(4):2274 – 2280, August 2008.

[25] R.L. Shuler, B.L. Bhuva, P.M. ONeill, J.W. Gambles, and S. Rezgui. Comparison of dual-rail and tmr logic cost effectiveness and suitability for fpgas with reconfigurable seu tolerance. *IEEE Transactions on Nuclear Science (TNS)*, 56(1):214 – 219, February 2009.

[26] L. Sterpone and M. Violante. Analysis of the robustness of the tmr architecture in sram-based fpgas. *IEEE Transactions on Nuclear Science (TNS)*, 52(5):1545 – 1549, December 2005.

[27] S. Liu, G. Sorrenti, P. Reviriego, F. Casini, J.A. Maestro, and M. Alderighi. Increasing reliability of fpga-based adaptive equalizers in the presence of single event upsets. *IEEE Transactions on Nuclear Science (TNS)*, 58(3):1072 – 1077, June 2011.

[28] F. Abate, L. Sterpone, C.A. Lisboa, L. Carro, and M. Violante. New techniques for improving the performance of the lockstep architecture for sees mitigation in fpga embedded processors. *IEEE Transactions on Nuclear Science (TNS)*, 56(4):1999 – 2000, August 2009.

[29] Q. Zhao, Y. Ichinomiya, M. Amagasaki, M. Iida, and T. Sueyoshi. A novel soft error detection and correction circuit for embedded reconfigurable systems. *IEEE Embedded Systems Letters (ESL)*, 3(3):89 – 92, September 2011.

[30] A. Tiwari and K.A. Tomko. Enhanced reliability of finite-state machines in fpga through efficient fault detection and correction. *IEEE Transactions on Reliability (TR)*, 54(3):459 – 467, September 2005.

[31] D.E. Johnson, K.S. Morgan, M.J. Wirthlin, M.P. Caffrey, and P.S. Graham. Detection of configuration memory upsets causing persistent errors in sram-based fpgas. In *Military Aerospace Programmable Logic Devices (MAPLD)*, September 2004.

[32] L. Sterpone and N. Battezzati. A new placement algorithm for the mitigation of multiple cell upsets in sram-based fpgas. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1231 – 1236, march 2010.

[33] S. Golshan and E. Bozorgzadeh. Seu-aware resource binding for modular redundancy based design on fpgas. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1124 – 1129, April 2009.

[34] 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 (TNS)*, 52(6):2455 – 2461, December 2005.

[35] H. Quinn and P. Graham. Terrestrial-based radiation upsets: a cautionary tale. In *Field-Programmable Custom Computing Machines (FCCM)*, pages 193 – 202, April 2005.

[36] Y. Tamir and M. Tremblay. High-performance fault-tolerant vlsi systems using micro rollback. *IEEE Transactions on Computers (TC)*, 39(4):548 – 554, April 1990.
[37] W.J. Huang and E.J. McCluskey. Transient errors and rollback recovery in lz compression. In *Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 128 – 135, December 2000.
[38] Xilinx. Virtex-6 family overview. http://www.xilinx.com/support/documentation/virtex-6.htm.
[39] Mentor Graphics. Modelsim. http://www.mentor.com.
[40] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert. Statistical fault injection: Quantified error and confidence. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 502 – 506, April 2009.
[41] J.F. Ziegler, H.P. Muhlfeld, C.J. Montrose, H.W. Curtis, T.J. O'Gorman, and J.M. Ross. Accelerated testing for cosmic soft-errors rate. *IBM Journal of Research and Development (JRD)*, 40(1):51 – 74, January 1996.
[42] Xilinx. Xpower analyzer. http://www.xilinx.com/products/design_tools/logic_design/verification/xpower_an.htm.