Adaptive Mitigation of Radiation-Induced Errors and TDDB in Reconfigurable Logic Fabrics

Rawad Al-Haddad, Rashad S. Oreifej, Ramtin Zand, Abdel Ejnioui, and Ronald F. DeMara
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL, 32816-2362
demara@mail.ucf.edu

Abstract— Self-reliance capabilities of mission-critical systems gain importance as technology scaling and logic capacity of SRAM-based reconfigurable devices increase. The Sustainable Modular Adaptive Redundancy Technique (SMART) is evaluated to optimize the reliability, availability, and energy efficiency of reconfigurable logic devices with a given area footprint. A Monte Carlo driven Continuous Markov Time Chain (CMTC) simulation is conducted to assess availability using runtime adaptation with SMART in comparison to conventional design-time static Triple Modular Redundancy (TMR) techniques. In harsh environments, adaptive redundancy is shown to improve system availability under lengthy repair times, and to a more significant degree under rapid recovery times. When compared to TMR, adaptive redundancy achieves power savings ranging from 22% to 29%, at a reduced area cost ranging from 17% to 24%, while maintaining comparable levels of availability.

***Index Terms*— SRAM-based Field Programmable Gate Arrays (FPGAs), Radiation effects, Single event effects, Soft errors, Hard faults, Failure analysis, IC reliability, Triple Modular Redundancy (TMR), Harsh radiation environment, Scrubbing, Adaptive computing architectures.**

I. Introduction

Transient faults, which commonly occur as a *Single Event Upset (SEU)* are a primary source of concern when deploying SRAM-based FPGA devices in space applications. Whereas SEU is non-destructive, *Single Event Latch-up (SEL)* can be destructive as it occurs when a charged particle causes excessive supply power to destruct permanently the memory cell [1]. *Triple Modular Redundancy (TMR)* is capable of masking the transient effects [2] while scrubbing techniques overwrite SEUs in the configuration logic [3]. Numerous capable FPGA fault-handling techniques have been developed, some of which are covered in [4]. Focusing on dynamic techniques, [5] presents TMR coupled with the partial dynamic reconfiguration to mitigate the effects of soft errors, whereas the approach herein provides two-levels of hard-fault refurbishment. Bounded error recovery time in FPGA-based TMR circuits using dynamic partial reconfiguration is presented in [6, 7] that focuses on circuits that recover from SEUs within a specified period as an alternative to scrubbing. It attempts to minimize area by reducing scrubbing, whereas the approach herein adapts the redundancy level based on fault occurrence and mission requirements.

In this paper, we show how adaptive recovery techniques can outperform design-time deterministic TMR for the above failure modes. To manage the tradeoff between reliability and overhead, SMART exploits reconfigurability to realize adaptive levels of redundancy. As such, SMART can improve system availability over static TMR while optimizing area usage and energy consumption. In addition to exploiting FPGA reconfigurability, SMART refurbishes functionality due to multiple hard faults, which is a capability beyond TMR, using genetic algorithms optimized for refurbishment. We relied on a standard triplication tool called BL-TMR to generate 28 benchmarks to be used in evaluating SMART's overhead [8].

II. Autonomous Fault-Tolerance in SMART

SMART consists of a hardware layer realizing adaptive redundancy and a software layer which is not on the critical throughput path to control availability, energy consumption, and area in dynamic environments which is presented in detail in [9] and overviewed here. The hardware layer realizes self-repair using a Reconfigurable Adaptive Redundancy System (RARS) arrangement shown in Figure 1 [9]. It consists of one Autonomous Element (AE) and three identical Functional Elements (FEs) which reside in the reconfigurable fabric. The software layer realizes self-healing capabilities through an application-independent, intrinsic, evolutionary repair technique of *Organic Genetic Algorithm (OGA)*, which leverages the benefits of dynamic partial reconfiguration. Figure 2 depicts the high-level view of SMART's refurbishment operations and events that trigger their execution.

With reports from RARS, SMART varies dynamically the number of redundant modules by reorganizing components to provide the appropriate level of redundancy to match mission requirements in terms of a system-level performance metric, e.g. SNR in a signal processing application. This approach is different from the one commonly found in other fixed redundancy techniques such as TMR. Furthermore, SMART minimizes power consumption by operating in a high power mode only when multiple instances of the user application are needed to identify, mask, or repair faults. For a detailed description of SMART components and their internal operation, the reader is referred to [9] due to page-limitations here.

We step through the use case of a typical edge detection algorithm to demonstrate organic system operation [9]. Three scenarios involving hard faults are evaluated. The first scenario involves injection faults with a disabled AE where the system operates in duplex mode with two FEs executing the edge-detection algorithm, while the third FE is in 'cold standby' inactive mode. While fault injection is enabled, the inactive AE does not monitor the faults in the other FEs. In this scenario, the

This research was supported in part by NASA Intelligent Systems NRA Contract NNA04CL07A and *DARPA* contract #W31P4Q-08-C-0168.

image displays some degradation as the edge detected in the image starts to show faulty pixels as indicated in Figure 3(b).

The second scenario involves injecting faults with an enabled AE. In this scenario, the system operates in duplex mode with two FEs running the edge-detection algorithm, while the third FE is in ‘cold standby’ inactive mode. The distinction is that the AE is enabled to monitor faults in the FEs since fault injection is activated. Image quality remains constant as follows. The intervention steps of the AE start by detecting a discrepancy in FE1 to deactivate it by changing its status to faulty. Next, the AE activates FE3 by switching its status to online. Finally, the AE enables its voter, which detects the discrepant FE1. As the output is read from the majority vote report, the system does not show any performance degradation as indicated in Figure 3(a).

In recovery, fault injection is disabled by using a triplex system in which FE1 is faulty similar to the preceding scenario. In this case, the report from the voter of the AE changes from a discrepant FE1 to no discrepancies. After a discrepancy-free window duration, the AE determines that the fault has been recovered and thus no longer needs to operate in triplex mode. In this case, the AE disables the voter and FE3.

Visually, Figure 3(a) shows a fault-free satellite image displaying urban buildings with industrial fans on their roofs as a result of processing by Sobel edge detection. Figure 3(b) depicts the same image after a single-fault injected in FE1, FE2 or FE3. Upon the detection of the discrepancy caused by the fault, RARS switches to triplex mode, thereby allowing the system to maintain 100% of its fault-free throughput. Hence there is no degradation in quality as compared to the fault-free scenario. Figure 3(c) depicts the impact of two faulty FEs where system performance clearly drops, as can be seen from the degraded edge-detected image. When the software monitoring layer of SMART initiates the refurbishment of one of the faulty FEs through PR, the system regains 100% performance, as shown in Figure 3(d). Thus, the application throughput is restored using hardware identification of resource capabilities and autonomous refurbishment realized by the OGA.

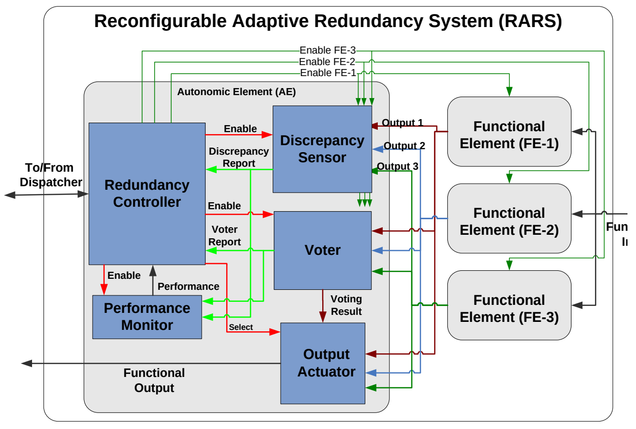


Figure 1. Reconfigurable adaptive redundancy arrangement [9].

III. Markov Model of Online Refurbishment

SMART has two main advantages, namely the capability of handling hard faults and adapting redundancy based on the mission reliability and resource requirements. Both advantages need to be properly evaluated using standard metrics. Calculating *Mean Time To Repair (MTTR)* of hard faults and

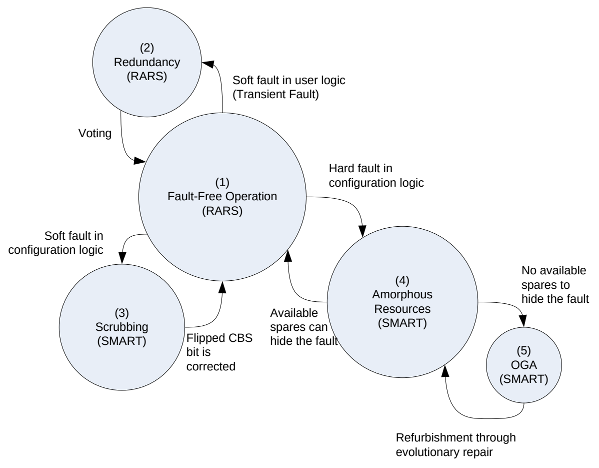

(2) Redundancy (RARS)
Soft fault in user logic (Transient Fault)
Voting
(1) Fault-Free Operation (RARS)
Hard fault in configuration logic
Soft fault in configuration logic
(3) Scrubbing (SMART)
Flipped CBS bit is corrected
Available spares can hide the fault
(4) Amorphous Resources (SMART)
No available spares to hide the fault
(5) OGA (SMART)
Refurbishment through evolutionary repair

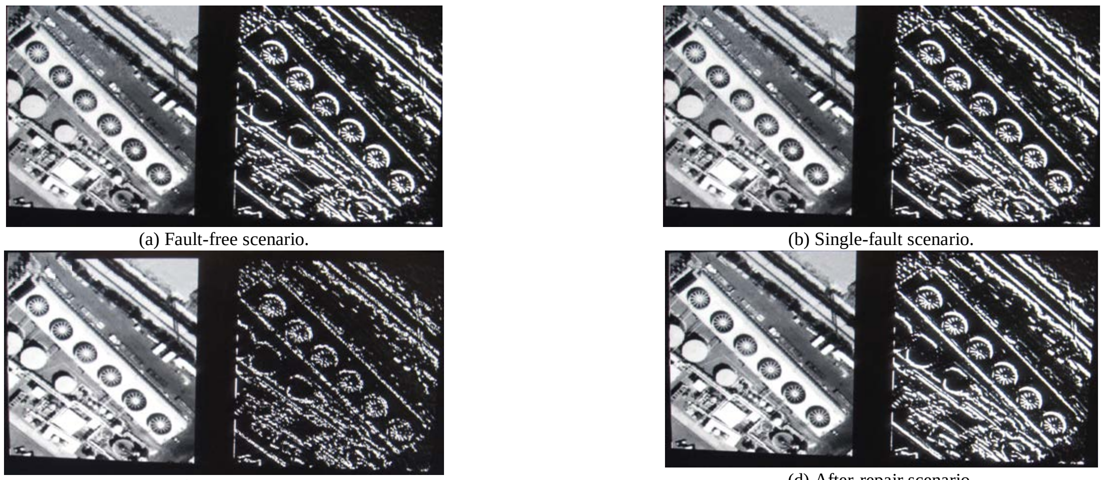
Figure 2. High-level operational view of SMART repair methods [9].

soft faults is not enough to judge SMART’s abilities to sustain realistic missions. Thus, based on published data and experimental measurements of SMART’s prototype, we formulated nine space missions to use in evaluation. Then, CMTC was employed to simulate SMART’s behavior in these missions to calculate the overall availability of the system and the time it spends in each repair stage. The common convention in evaluation resource overhead against the industry-standard TMR approach is to analytically assume that TMR requires three times the FEs’ overhead plus the voter’s overhead. This however is not always the case due to the abundance of triplication optimization algorithms that can do better. Thus, we relied on a standard triplication tool called BL-TMR to generate 28 benchmarks to be used in evaluating SMART’s overhead [8].

The power and area results for the edge detection application that we developed were extracted experimentally from the Xilinx Power Analyzer (XPA) and the Place And Route (PAR) reports. XPA reports power consumption while PAR reports physical resource usage (LUTs, Input-Output Block (IOBs), BRAM, etc.). As for the TMR benchmark results we intended to use for comparison, we have employed BL-TMR for triplication. This tool takes the Electronic Data Interchange Format (EDIF) netlist of a design and produces a triplicated EDIF netlist of the same design. EDIF generation is embedded in the Xilinx flow using the NGD2EDIF tool that can generate EDIF representation of the design from the Native Generic Database (NGD) file. The resulting EDIF file can undergo the triplication process of BL-TMR to generate the triplicated EDIF, which can be translated back to the Xilinx process file formats using the EDIF2NGD tool. This custom triplication flow starts after the standard Xilinx flow generates the NGD file in order to apply the triplication using the BL-TMR redundancy generation flow.

A. Mission Use Cases

Combining the three FEs inside the Reconfigurable Adaptive Redundancy System (RARS) on a Virtex XC4VSX35 occupies around 1,800 LUTs out of 34,560 LUTs available in this device [9]. Based on values reported in published work and



(a) Fault-free scenario.

(b) Single-fault scenario.

(c) Two faulty FEs scenario.

(d) After-repair scenario.

Figure 3. Original and edge-detection images under different RARS configurations [9].

experimental results attained by SMART prototype, and to produce a set of use cases in examining SMART's performance, we report the following fault error rates and repair rates for soft and hard faults.

CREME-96 simulator was used in [10] to calculate predicted *Soft-Fault Rate (FS)* per day for different 90nm device types. Assuming low earth orbit with altitude of 800 Km and inclination of 22.0 degrees, the reported SEU rate per day is 7.56 for Xilinx XQR4VSX55 90nm FPGA. This device has 24,567 slices [11], each slice has two LUTs (G and F). Thus, the final rate for SEU/LUT in hours is $(7.56 \div 49{,}134) \div 24 = 6.411 \times 10^{-6}$. This value will be multiplied by the number of LUTs in each FE to determine the soft fault rate in each FE per one simulation hour.

We calculated the *Soft-Fault Repair Rate (RS)* based on scrubbing speed in the SMART JTAG-based prototype to account for the worst case scenario, which takes around 39.56 seconds to initialize the boundary scan chain, download the bitstream, readback/verify the bitstream, and evaluate the FE for a wide window of functional input to ensure the SEU is corrected in the CBS. Even if Internal Configuration Access Port (ICAP) is used to expedite the scrubbing process, the assumed value is still valid as one can always expand the evaluation window for the repaired FE to gain higher statistical confidence that the fault is indeed repaired.

Hard-Fault Rate (FH) is the rate of TDDB failures. Many claim that Virtex devices are SEU immune to specific levels radiation. However, based on Xilinx published data, [12] predicts 10% of the LUTs in a circuit to be affected by TDDB per year under the most stressful conditions of t_{ox}=1.2 nm, oxide area = 0.25 mm^2, at 125 °C and 3.0 V. The static signal probability is assumed to be 1 because the LUT is a lookup table that has all gates turned on all the time. For the sake of proper factorial experimental design, we assumed three levels of FS based on the environmental conditions, where the fault rate under *demanding* conditions is assumed to be 10%, under *moderate* conditions is 5%, and under *favorable* condition is 1%. These values are relative to the adjusted FE size that takes into consideration the resource decomposition rate such that the FE ends up with 600 LUT/FE at the end of the mission time. The same aging rate dictated the length of simulation time, as the system cannot function for more than 10 years given a hard fault rate of 10% of the LUTs per year.

Similar to the FH case, we calculate three levels of *Hard-Fault Repair Rate (RH)* to establish full 3x3 factorial experiment. The three levels of RH were calculated based on simulation results for OGA with different level of hard fault impact on the LUTs, where a hard fault can impact one, two, or four bit(s) of the LUTs. The associated repair rates correspond to *rapid*, *intermediate*, and *lengthy* repairs. The number of generations and the repair time/rate are listed in Table I.

TABLE I. OGA Results for Various Numbers of Hard Faults.

Number of Faults	1	2	4
Generations	3962	31352	63307
MTTR (hours)	0.704415	5.573703111	11.25462
RH (hours)	1.4196177	0.17941393	0.0888524

Conventional TMR and scrubbing techniques commonly found in the literature do not have hard-fault repair, thus will have RH converging to infinity. Plugging finite large numbers for RH in the CTMC model, which will be demonstrated in the next section, caused the system to stay in faulty states from the point it is hit with a hard-fault until the end of the mission. The nine use cases and the simulation parameters are summarized in Table II. The resulting 3x3 experiments represent nine mission scenarios for different operating conditions. RS and FS values are fixed for all nine use cases.

The variation that is seen in the table is due to the different number of LUTs required for different experiments to accommodate the decomposition rate of the LUTs. Only FH and RH are varied across the experiments as they represent the main focus of this work and demonstrate the true contribution of SMART over conventional repair techniques. The simulation time assumes each year has 10,000 hours.

B. Markov Models

The first evaluation metric that we present to qualify SMART's benefit over conventional TMR is reliability modeling using CTMC. CTMC relies on a state-transition diagram that depicts a state space of the chain, which is defined by all the states that the system can traverse during its operation,

TABLE II. FAULT AND REPAIR VALUES OF THE NINE USE CASES.

UC #	Description	FH (per hour)	FS (per hour)	RS (per hour)	RH (per hour)	Simulation Time (hours)
1	Demanding conditions Rapid repair	0.065753425	0.038466235	90.91	1.4196177	10,000
2	Moderate conditions Rapid repair	0.006027397	0.007052143	90.91	1.4196177	20,000
3	Favorable Conditions Rapid Repair	0.000723288	0.004231286	90.91	1.4196177	60,000
4	Demanding conditions Lengthy repair	0.065753425	0.038466235	90.91	0.0888524	10,000
5	Moderate conditions Lengthy repair	0.006027397	0.007052143	90.91	0.0888524	20,000
6	Favorable Conditions Lengthy Repair	0.000723288	0.004231286	90.91	0.0888524	60,000
7	Demanding conditions Intermediate Repair	0.065753425	0.038466235	90.91	0.17941393	10,000
8	Moderate conditions Intermediate Repair	0.006027397	0.007052143	90.91	0.17941393	20,000
9	Favorable Conditions Intermediate Repair	0.000723288	0.004231286	90.91	0.17941393	60,000

along with the possible transitions between the states with each transition being characterized by a transition probability. Based on these states and transitions, the model can be solved analytically or simulated experimentally to calculate the probability of being in a certain state based on the previous state [13]. Moreover, Monte Carlo simulation of the CTMC can predict the expected transitions that the system is likely to undergo with time [14]. We perform a comparative study using Markov tools to quantify the effect of having hard-fault repair in mission critical applications. The system that we model is RARS, which has three instances of the user applications and is capable of switching from duplex to triplex configuration [9] [15][16]. The resulting Markov state transition diagram is shown in Figure 4.

For instance, state S8 is said to be faulty because it has all FE's faulty, two of which have soft faults and one has hard fault. The states belong to vertical lanes that denote the total number of faulty FEs in RARS. The possible transitions between the states are characterized by one of the following rates: FS: Soft fault rate, denoting the SEU rate in the system. RS: Soft repair rate, which is the time needed to scrub the CBS to restore the correct value of faulty LUTs. FH: Hard fault rate, which

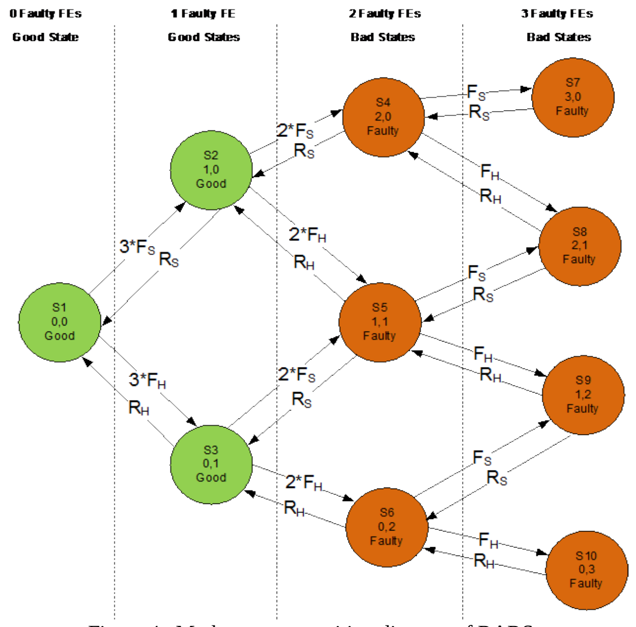


Figure 4. Markov state transition diagram of RARS.

represents the TDDB fault rate. RH: Hard repair rate, which is the time that the OGA needs to repair faulty FEs.

The system starts from an initial good state S1, which has 0 faulty FEs. A soft fault can occur with a rate of FS to put the system in S2 (1, 0), or a hard fault can occur with a rate of FH to put the system in state S3 (0, 1). RARS is expected to stay error-free even with the existence of one faulty FE, at the expense of switching from the low power and area duplex mode to the high power and area triplex mode. Thus, S1 is not different from S2 and S3 in term of availability, but does consume less area and power. For all Sn (n>3), RARS will be unavailable and will also consume high area and power similar to S2 and S3 because the triplex configuration is needed during repair. RS or RH repairs will move the system from faulty states to healthier ones.

C. *Availability Evaluation Metric Results*

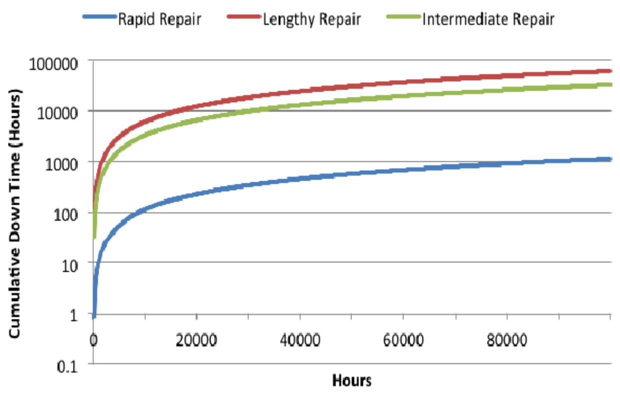
Table III can be of great importance in pre-deployment preparations as it can tell the system designers where to focus in order to handle the common case scenarios. For instance, none of the use cases has entered S7 (all 3 modules hit by SEU) due to the very low MTTR compared to the high MTTF in the soft fault case. This analysis can impact design decisions such as the interfacing between the scrubber and the FEs or the number of ports in the reconfiguration ROM, as the system is highly unlikely to scrub three FEs at the same time. Similar conclusions can be drawn about S9 and S10 for use case 3. The availability of the nine use cases are reported in the last row of Table III. Availability is calculated according to Mean Time To Failure (MTTF) and Mean Time To Repair (MTTR) values of the system, $Availability(A) = \frac{MTTF}{MTTF+MTTR}$. The reported availability makes clear that the demanding conditions can greatly impact the system availability to levels below the accepted standards (use case 4: A=36%, use case 7: A=65.6%). A mission operating in such harsh conditions must be equipped with quick repair mechanisms to be able to process the rapid arrival rate, and thus be able to produce relatively higher availability rates such as use case 1 where A=98.8%. The impact of lengthy repair is also demonstrated in Table III. A rapid repair will move the system from 98.8% availability under the demanding conditions (UC1), to three nines under moderate conditions (UC2), to six nines under favorable conditions (UC3). Similarly, the impact of mission conditions on the performance of a particular fault-tolerance approach is great; such impact can be demonstrated by scrutinizing the results of

TABLE III. AVERAGE CUMULATIVE TIME IN STATE FOR THE NINE USE CASES (UCs).

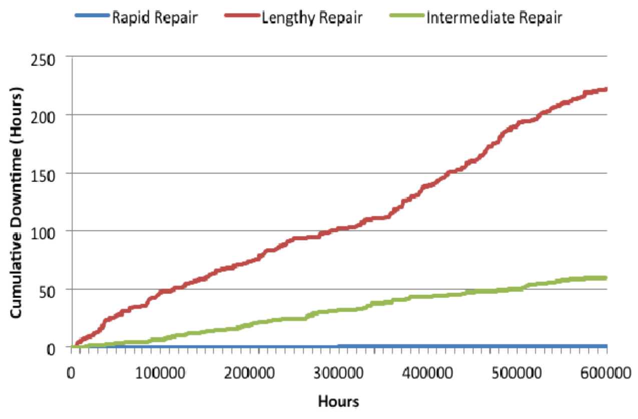
S	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9
1	86617.1	197416	598992	11119	162040	585007	31157	180556	592712
2	110.433	45.2201	83.5574	14.3668	37.6925	81.7908	39.7021	42.2014	83.0391
3	12015.3	2516.95	923.3729	24848.4	33194.18	14687.45	34308.25	18107.62	7144.62
4	0.09638	0.01509	0.0041015	0.01409	0.003113	0.0075195	0.0494751	0.0003906	0.01094
5	10.3220	0.41594	0.090234	21.1690	5.169533	1.313137	29.26737	2.807783	0.63295
6	1096.85	20.7712	0.783642	36652.5	4428.781	221.4332	25121.46	1251.339	58.3688
7	0	0	0	0	0	0	0	0	0
8	0.09406	0.01069	0.0039	0.02598	0.00119	0.00468	0.04846	0.00312	0.00996
9	0.44219	0.00098	0	15.3885	0.323547	0.012548	10.60557	0.105731	0.00234
10	49.3238	0.09766	0	27229.0	293.4965	0.722656	9233.248	39.06087	1.36357
Availability(A)	0.98842	0.999893	0.999998	0.36018	0.976361	0.999627	0.655709	0.993533	0.999899

use cases 4, 5, and 6 which all utilize lengthy repair mechanisms. The mission conditions can elevate the system availability from 36% to 99.9%, making a huge impact on the mission success rate. Table III only shows the cumulative time at the end of the simulation. To further explain the behavior of the system, we plot the cumulative downtime of each use case versus the mission time. The use cases need to be grouped by FH because the hard fault rate will impact the maximum mission time. So, under the most demanding conditions of 10% of the LUT impacted by hard faults each year, the system can live for 10 years maximum, after which all LUTs will be impacted by faults. Figure 5 shows the cumulative downtime of the system with time. The first figure, corresponding to the demanding conditions, is plotted on logarithmic Y-axis due to the huge divergence in cumulative downtime of rapid and lengthy repairs. For instance, Figure 5(a) shows that the mission that is equipped with rapid repair mechanism resulted in 1,000 hours of system downtime, whereas a system with lengthy repair resulted in more than 60,000 hours of downtime, confirming the importance of efficient hard-fault repairs in SMART. On the other hand, Figure 5(b) depicts the favorable mission conditions, whereby it was plotted on a linear scale because of the relatively marginal difference between the use cases with the rapid and lengthy repairs. Even after running for 60 years, the system with the lengthy repair only accumulated approximately 225 hours of downtime. One can argue that in such favorable conditions a hard-fault repair mechanism would not be required, but this really depends on the mission type. If this is an imaging application aiming to capture explorative images then we might agree. However, if the FE is designed for a more critical application, such as a power controller or security-critical encryption circuit, then 225 hours, a little more than 9 days, can represent a significant period of time that may jeopardize the mission success.

Figure 6 shows the availability of the nine use cases throughout the mission lifetime. The impact of the hard-fault rate on the system availability is demonstrated under a GA-based repair process which can exhibit the spikes observed during refurbishment due to its stochastic behavior. The system with lengthy repair shows Availability<0.4 under demanding conditions in Figure 6(a) and 0.9996 under favorable conditions in Figure 6(b). The availability of the use cases is also affected by the repair time as shown in the two figures, especially when the fault rate is high to push RARS toward faulty states without a repair mechanism and an MTTR that is low enough to bring it back to the healthy states. Figure 7 shows the percentage of time spent in each of the 10 states under each of the nine use cases. It is clear that use cases 4 and 7, with demanding conditions and lengthy as well as intermediate repair respectively, are the ones that register less presence in S1 and spend more time in S6 and S10. A practical way of studying Figure 7 is to combine the states based on their overall impact on the mission status, meaning that S1 by itself is a distinguished state which guarantees that the system is available and is running in reduced power and area modes through the exploitation of the reconfiguration property of the FPGA.



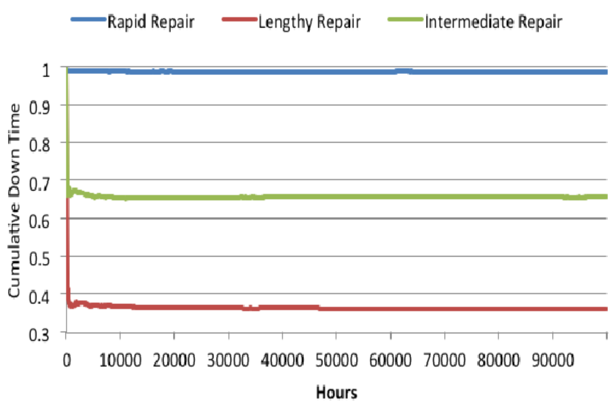
(a) Demanding condition.



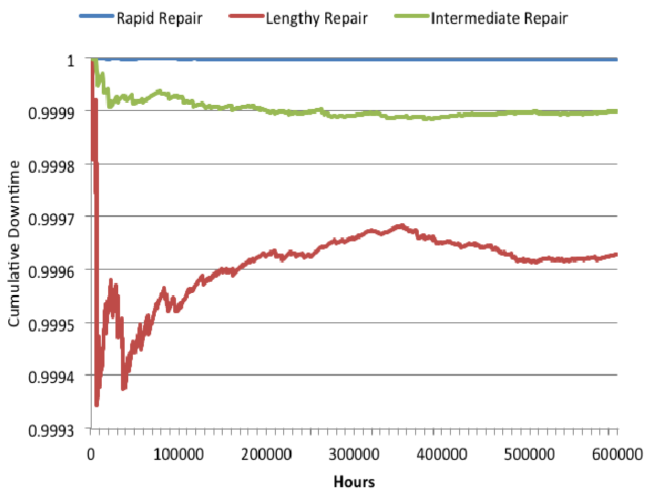
(b) Favorable condition.

Figure 5. Cumulative downtime under the nine use cases.

A less desirable outcome of the system is seen in S2 and S3, where the system is still available via triplex configuration of RARS, yet consume more power and area than S1. The availability in these two states is exactly equal to S1's availability, but the system is less reliable as it cannot handle any further faulty FEs. The remaining states from S4 to S10 represent the least desirable system conditions where it expends the triplex power and area, yet is not sufficiently available. A design goal of SMART is to minimize the time spent in S4 to S10. Another important conclusion that can be drawn from Figure 7 is that the faulty states that are actually traversed throughout the mission lifetime are the ones that comprise hard faults, which are S3, S6, and S10. This can be attributed to the high MTTR for hard faults compared to soft faults.



(a) Demanding condition.



(b) Favorable condition.

Figure 6. Availability under the nine use cases.

The states that feature soft faults are visibly negligible, because SMART is able to exit them in very short time by applying partial reconfiguration scrubbing. In fact, Table III shows that S7 which represents three soft-faulty FEs was never visited even with very long simulation times (60 years), a clear indication that conventional repair techniques can efficiently handle soft faults, steering the attention to hard-fault repair as a vital requirement for autonomous fault-handling in mission critical systems running in harsh environments. Finally, to quantify the aggregation of the states of RARS, Figure 8 depicts the percentage of time spent on each of the operation phases under the nine use cases. (A) with lower power and area represents S1, (A) with high power and area combines S2 and S3, whereas (1-A) corresponds to states S4-S10.

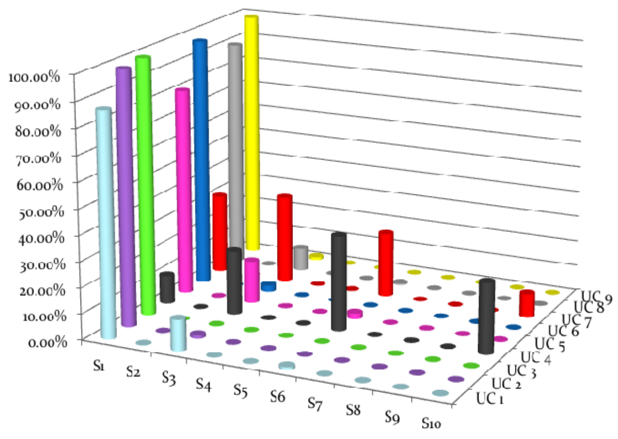
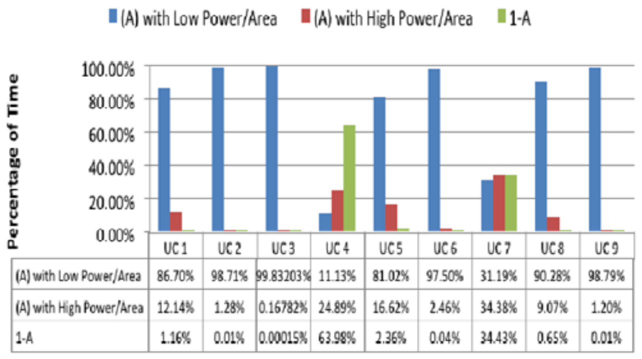


Figure 7. Percentage of time in each state under the nine use cases.



	UC 1	UC 2	UC 3	UC 4	UC 5	UC 6	UC 7	UC 8	UC 9
(A) with Low Power/Area	86.70%	98.71%	99.83203%	11.13%	81.02%	97.50%	31.19%	90.28%	98.79%
(A) with High Power/Area	12.14%	1.28%	0.16782%	24.89%	16.62%	2.46%	34.38%	9.07%	1.20%
1-A	1.16%	0.01%	0.00015%	63.98%	2.36%	0.04%	34.43%	0.65%	0.01%

Figure 8. Operational phases distribution under the nine use cases.

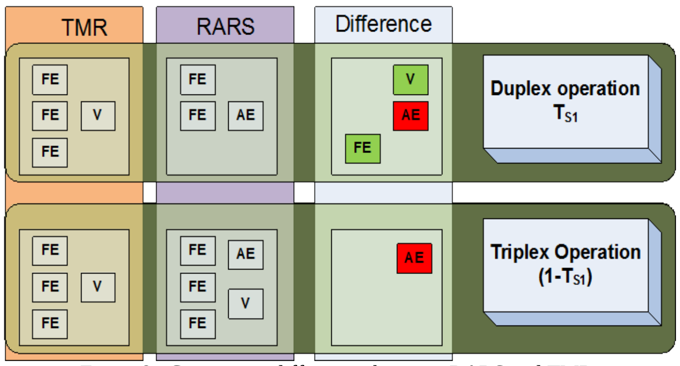


Figure 9. Component differences between RARS and TMR.

TABLE IV. PARAMETER DEFINITION IN OVERHEAD ANALYSIS.

Term	Definition
O_{FE}	Overhead of one FE
O_{AE}	Overhead of AE (without the voter component)
O_{V}	Overhead of the voter
O_{TMR}	Overhead of the TMR
O_{RARS}	Overhead of RARS
O_{DX}	Overhead of RARS when it runs the duplex mode
O_{TX}	Overhead of RARS when it runs the triplex mode
O_{S}	Overhead saving by using RARS over conventional TMR

Use case 3 with favorable conditions and rapid repair has almost negligible (1-A) component. Use cases 4 and 7 with demanding conditions and lengthy and intermediate repair, respectively, are the ones that spend more time in (1-A) than in (A). Other use cases show mixed behaviors that correlate to the reaction time to faults and their arrival rates. Such figure can be constructed based on the expected mission conditions and the fault-tolerance overhead.

D. Area and Power Comparison to Industry-Standard Techniques

Referring to the CTMC experiment in the previous section, we define *Time in State 1 (T_{S1})* as the period of time in which RARS is in S1 and thus offers power and area saving over TMR while providing the same level of availability. The component difference between RARS and TMR is shown in Figure 9. Let the parameters of interest be denoted as shown in Table IV. The Overhead of RARS is a weighted average controlled by T_{S1}. The duplex overhead is two times the FE overhead plus the AE overhead, whereas the triplex overhead is three times the FE overhead plus the AE and the voters overhead:

$$O_{RARS} = T_{S1} \times O_{DX} + (1 - T_{S1}) \times O_{TX} \quad (1)$$

$$O_{DX} = 2 \times O_{FE} + O_{AE} \quad (2)$$

$$O_{TX} = 3 \times O_{FE} + O_{AE} + O_{V} \quad (3)$$

The goal is to calculate the overhead savings of RARS compared to TMR as follows:

$$O_{S} = \frac{O_{TMR} - O_{RARS}}{O_{TMR}} \quad (4)$$

E. Experimental Setup

The BL-TMR tool was run 28 times to generate triplicated designs of the FEs with the specification listed in Table V. The following list of options depicts the TMR configurations that were considered to synthesize the circuits for the experiment:

- Voter Insertion Location:
  1) *Triplicate Logic (TL)*: Only internal logic, including clock signals, will be triplicated, without triplication of the IOs.
  2) *Triplicate Logic and Input ports (TLI)*: The logic and the input ports will be triplicated.
  3) *Triplicate Logic and Output ports (TLO)*: the logic and the output ports will be triplicated.
  4) *Triplicate Logic, Input, and Output ports (TLIO)*: Triplicates all logic, input, and output signals.
- Voter Insertion Algorithm:
  1) *Voters before Every Flip-Flop (FF)* Algorithm: This algorithm will place a voter before the data input of every FF. The algorithm is very simple and does not require heavy analysis of the design, it guarantees that only one voter will be inserted in any timing path, reducing the negative timing impact of the triplication.
  2) *Voters after Every FF* Algorithm: Similar to the previous algorithm, but inserts the voter after the FF. This has produced the best timing results.
  3) Basic *Strongly Connected Components (SCC) Decomposition* Algorithm: applies Kosaraju algorithm to remove all feedbacks from the SCC. It runs quickly but produces unsatisfactory timing results compared to the other algorithms because it allows more than one voter in the timing path.
  4) *Highest Fan-out SCC Decomposition* Algorithm: Reduces the number of voters using a heuristic search to find nets with high Fan-out as candidate places to insert voters.
  5) *Highest FF Fan-out SCC Decomposition* Algorithm: Combines 4 and 2, it guarantees that only one highest Fan-out voter is inserted per timing path, by inserting it after the FF outputs, resulting in cutting more voters and thus protecting the timing paths and saving more area.
  6) *Highest FF Fan-in Input*: This option finds the highest fan-in FF in the SCC that is a legal voter location.
  7) *Highest FF Fan-in Output*: This option is identical to option 6, but inserts the voter after the identified FF.

TABLE V.BL-TMR TRIPLICATED EDGE DETECTOR BENCHMARKS.

Bench-mark	Triplication Location	Voter Insertion Algorithm
1	Logic only (TL)	Before Every Flip-Flop
2		After Every Flip-Flop
3		Basic Strongly Connected Components (SCC) Decomposition
4		Highest Fan-out SCC Decomposition
5		Highest Flip-Flop Fan-out SCC Decomposition
6		Highest Flip-Flop Fan-in Input
7		Highest Flip-Flop Fan-in Output
8	Logic and Input ports (TLI)	Before Every Flip-Flop
9		After Every Flip-Flop
10		Basic Strongly Connected Components (SCC) Decomposition
11		Highest Fan-out SCC Decomposition
12		Highest Flip-Flop Fan-out SCC Decomposition
13		Highest Flip-Flop Fan-in Input
14		Highest Flip-Flop Fan-in Output
15	Logic and Output ports (TLO)	Before Every Flip-Flop
16		After Every Flip-Flop
17		Basic Strongly Connected Components (SCC) Decomposition
18		Highest Fan-out SCC Decomposition
19		Highest Flip-Flop Fan-out SCC Decomposition
20		Highest Flip-Flop Fan-in Input
21		Highest Flip-Flop Fan-in Output
22	Logic, Input, and Output ports (TLIO)	Before Every Flip-Flop
23		After Every Flip-Flop
24		Basic Strongly Connected Components (SCC) Decomposition
25		Highest Fan-out SCC Decomposition
26		Highest Flip-Flop Fan-out SCC Decomposition
27		Highest Flip-Flop Fan-in Input
28		Highest Flip-Flop Fan-in Output

This has resulted in 4 × 7 = 28 triplicated benchmark designs to be used in the comparison against RARS. The triplication was accomplished using the specialized BL-TMR tool on a post-synthesis EDIF netlist to optimize the design for area and speed. The resulting designs were first analyzed using the Xilinx Place And Route (PAR) reporting tools to calculate the area overhead of each benchmark. The full results are shown in Table VI. We rely on the “Total equivalent gate count” as generated by the Xilinx tool to be the area overhead metric in this experiment. Benchmark 5 (Highest Flip-Flop Fan-out SCC Decomposition, Logic Only) resulted in the least number of gates out of the 28 benchmarks.

But first, O_{DX} and O_{TX} must be calculated by synthesizing the sub-modules of RARS independently and generating the Xilinx

PAR reports accordingly. The results of the FE, AE, and Voter areas are shown in Table VII. Substituting the values in Equations (2) and (3), O_{DX}=15,115 gates and O_{TX}=22,793 gates. O_{RARS} can be calculated for any given T_{S1}. Measured

TABLE VI. AREA RESULTS OF THE 28 BENCHMARKS.

Benchmark	Slices	4 input LUTs	Total equivalent gate count
1	1,294	2,148	20,629
2	1,320	2,144	21,307
3	1319	2148	20,953
4	1237	2006	20,083
5	1182	1925	19,975
6	1260	2079	20,215
7	1185	1928	19,993
8	1297	2173	20,779
9	1323	2145	21,313
10	1323	2149	20,959
11	1240	2007	20,089
12	1185	1926	19,981
13	1264	2080	20,221
14	1188	1929	19,999
15	1343	2229	21,771
16	1,416	2,289	22,833
17	1,357	2,109	21,375
18	1,256	1,980	20,583
19	1,200	1,899	20,475
20	1,304	2,037	20,619
21	1,203	1,902	20,493
22	1,370	2,253	21,915
23	1,434	2,289	22,833
24	1,388	2,109	21,375
25	1,275	1,980	20,583
26	1,218	1,899	20,475
27	1,339	2,037	20,619
28	1,221	1,902	20,493

values indicate that the T_{S1} threshold after which RARS becomes beneficial in term of area is 37%. To generalize the area saving potential over a spectrum of T_{S1} values, we depict the relation between the total equivalent gate count of RARS and T_{S1}. On top of this, we overlay the 28 triplication benchmarks area results on a secondary x-axis as shown in Figure 10. The results show that RARS will become more beneficial than all the TMR benchmarks when T_{S1} is approximately greater than 40%. Next, the XPA tool was used to analyze the dynamic power consumption of the same 28 benchmark designs. Dynamic power consumption is greatly affected by the presence of triplicated IO's, and thus the results of TLIO sets were only considered to select the power winner design to be fair to RARS, which triplicates the input and output ports. Therefore, the best benchmark in the power category is benchmark 22 with 166.32 mWatts. Again, the same values were calculated for RARS sub-modules by synthesizing them independently and applying the XPA analysis to the resulting designs. The results shown in Table VIII indicate that the majority of the dynamic power is consumed by the FE elements

TABLE VII. AREA RESULTS OF RARS SUB-MODULES.

Module	Slices	4 input LUTs	Total equivalent gate count
One FE	348	616	6,495
AE (without Voter)	86	151	2,125
Voter	71	107	1,183

due to the amount of logic used compared to the AE and the Voter. The Voter and the AE consumed relatively equal amounts of dynamic power. Applying Equations (2) and (3), we calculate P_{DX}=117.22 mWatts and P_{TX}=177.98 mWatts. P_{RARS} can be calculated for any given T_{S1}.

TABLE VIII. POWER RESULTS OF RARS SUB-MODULES.

Module	Clock	Input	Output	Logic	Signals	Total
FE	6.2	15.59	31.15	1.11	1.69	55.74
Voter	4.38	0	0	0.52	0.12	5.02
AE (without Voter)	4.77	0	0	0.6	0.37	5.74

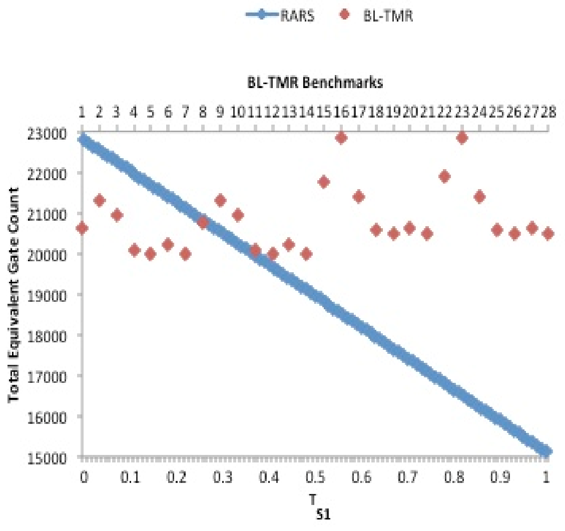


Figure 10. RARS area overhead relative to 28 benchmarks.

Plotting the power in mWatts versus T_{S1} will show linear savings with increased duplex time as shown in Figure 11. In comparison with the 28 benchmarks, RARS can still be beneficial for power savings unless TLI or TL are used, but this would decrease the reliability of the design because not all IOBs are triplicated, introducing many failure points to the system. Figure 12 depicts the percentage of power and area savings of RARS over the top two benchmarks, 5 and 22, except for the power of design 5, which does not include IOBs and thus produced very low power consumption at the expense of less

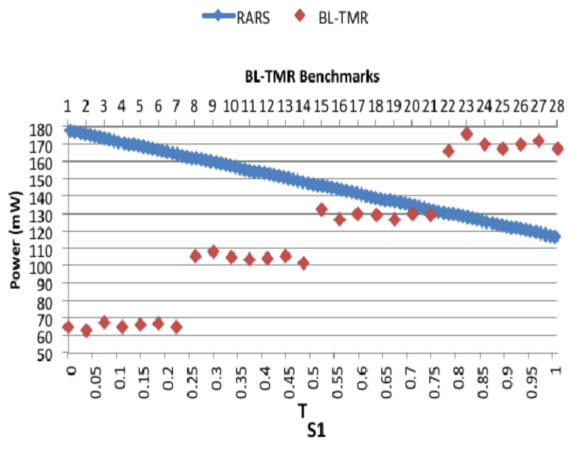


Figure 11. RARS power overhead relative to 28 benchmarks

TABLE IX. COMBINING AVAILABILITY, AREA, AND POWER RESULTS.

UC	S1 (A, Low Power, low Area)	S2, S3 (A, High Power, High Area)	S4-S10 1-A, High Power, High Area)	A (%)	Avg. Power	Avg. Area	Power Savings over Design 22	Area Savings over Design 5	Recommended Method
1	86.704%	12.1379%	1.15828%	98.8417	125.3	15319.13	24.66%	19.22%	SMART
2	98.707%	1.28331%	0.01014%	99.9899	118	14284.5	29.05%	23.83%	SMART
3	99.833%	0.16663%	0.00012%	99.9999	117.3	14187.37	29.46%	24.27%	SMART
4	11.130%	24.8876%	63.9821%	36.0179	171.2	21833.57	-2.94%	-9.83%	TMR
5	81.162%	16.5161%	2.32216%	97.6778	128.7	15796.86	22.64%	17.09%	SMART
6	97.522%	2.43057%	0.04736%	99.9526	118.7	14386.6	28.62%	23.38%	SMART
7	31.189%	34.3823%	34.4291%	65.5709	159	20104.55	4.38%	-2.12%	TMR
8	90.189%	9.15584%	0.65490%	99.3451	123.2	15018.69	25.94%	20.56%	SMART
9	98.798%	1.19488%	0.00749%	99.9925	118	14276.64	29.08%	23.87%	SMART

reliability. All the three lines enter the positive region of the Y-axis at $T_{S1}>37\%$. If the power is the main concern of the mission, then any $T_{S1}>20\%$ will mean that RARS will be more beneficial than any BL-TMR generated designs. Note that the previous power analysis ignores the impact of the power consumption of the reconfiguration process.

We experimentally calculated the T_{S1} values of the nine use cases, and used these realistic values as an input to the weighted average in Equation (1) to calculate the area and power overhead of RARS under the nine use cases. The RARS expected values were compared against benchmarks 5 and 22 as the top designs in term of area and power, respectively. Table IX shows the experiment results, where TMR was the recommended approach over SMART only in use cases 4 and 7. For the remaining use cases, SMART consistently showed better power and area requirements. The power savings ranged from 22% to 29%, whereas the area savings ranged from 17% to 24%. To clarify the power saving, it should be considered that a conventional TMR configuration utilizes three times the required logic and also a voter to assure that system can tolerate faults in a specific period of a mission. However, RARS employs the minimal level of redundancy to meet the mission fault tolerance requirements by only enabling redundant resources when they are actually needed. Thus, this design results in significant power saving during the fault free operation of system which is the most of its lifetime.

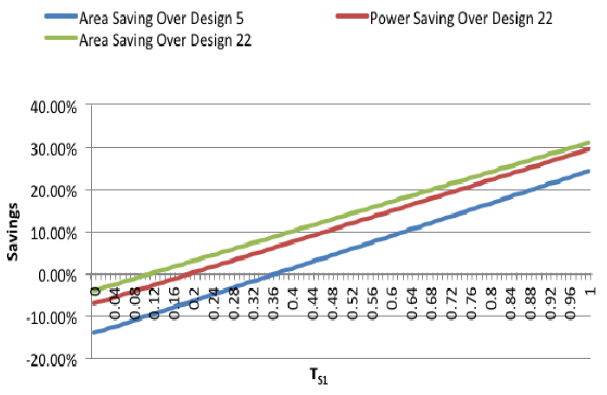


Figure 12. RARS area and power savings relative to the top two

IV. Conclusion

This paper presents a Monte Carlo driven Continuous Markov Time Chain simulation of an adaptive resilience hierarchy for refurbishment of hard and soft faults in FPGA devices. It provides a sustainable realization for long missions under a graded model of failure. The dilemma of choosing a fixed redundancy degree is avoided by deferring a commitment to a particular fault handling configuration until run-time with the decision process elevated above the critical throughput path. In the age of power-aware design which is also a long-standing considerations of many embedded systems, an adaptive technique is indicated to conserve up to 30% of power used by TMR while providing improved protection for suitable applications including future technology scaling and switching elements.

References

[1] D. C. Sharp, et al. “Challenges and solutions for embedded and networked aerospace software systems,” *Proceedings of the IEEE*, 2010.

[2] M. Caffrey, et al. “Assuring robust triple modular redundancy protected ciruits in SRAM-based FPGAs,” *Radiation Effects in Semiconductors*, CRC Press, 2011.

[3] J. Heiner, et al. “FPGA partial reconfiguration via configuration scrubbing,” *International Conference on Field Programmable Logic and Applications*, 2009.

[4] M. G. Parris, et al. “Progress in autonomous fault recovery of field programmable gate arays,” *ACM Computing Surveys*, 2011.

[5] C. Bolchini, et al. “TMR and partial dynamic reconfiguration to mitigate SEU faults in FPGAs,” *IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, 2007.

[6] E. Cetin, et al. “Towards bounded error receovery time in FPGA-based TMR circuits using dynamic partial reconfiguration,” *23rd International Conference on Field Programmable Logic and Applications*, 2013.

[7] H. Quinn, et al. “An automated approach to estimating hardness assurance issues in triple-modular redundancy circuits in Xilinx FPGAs,” *IEEE Transactions on Nuclear Science*, 2008.

[8] Brigham Young University, “BYU-LANL triple modular redundancy usage guide (Version 0.5.2),” *Brigham Young University*, 2009.

[9] R. Al-Haddad, et al. “Sustainable modular adaptive redundancy technique emphasizing partial reconfiguration for reduced power consumption,” *International Journal of Reconfigurable Computing*, 2011.

[10] C. W. Tseng, et al. “Static upset characteristics of the 90nm Virtex-4QV FPGAs,” *IEEE Radiation Effects Data Workshop*, 2008.

[11] Xilinx, “Xinlix Virtex-4 family overview, DS112(v1.1),” 2004.

[12] S. Srinivasan, et al. “Toward increasing FPGA lifetime,” *IEEE Transactions on Dependable and Secure Computing*, 2008.

[13] M. K. Ng, et al. “Markov Chains: Models, Algorithms and Applications,” *Springer*, ISBN 978-0-387-29337-0, 2006.

[14] R. Y. Rubinstein, et al. “Simulation and the Monte Carlo method,” *John Wiley & Sons*, 2011.

[15] J. Kok, et al. “FPGA implementation of an evolutionary algorithm for autonomous unmanned aerial vehicle on-board path planning,” *IEEE Transactions on Evolutionary Computation*, 2013.

[16] R. F. DeMara, et al. “Autonomic Fault-Handling and Refurbishment Using Throughput-Driven Assessment,” *Applied Soft Computing*, vol. 11, pp. 1588-1599, 2011.