TMR with More Frequent Voting for Improved FPGA Reliability

Brian Pratt,[1] Michael Caffrey,[2] Derrick Gibelyou,[1] Paul Graham,[2] Keith Morgan,[2] and Michael Wirthlin[1]

[1] Department of Electrical and Computer Engineering
Brigham Young University, Provo, UT 84604 USA
[2] Los Alamos National Laboratory, Los Alamos, NM 87545 USA

***Abstract*—Triple modular redundancy (TMR) is a popular technique for mitigating single-event upsets (SEUs) in FPGAs. Traditional TMR, however, is only designed to protect against a single fault at a time. TMR with more frequent voting (also called *partitioned TMR*) can provide improved reliability by giving protection against more than a single upset at a time. This paper implements partitioned TMR within an FPGA and demonstrates the improvements in reliability provided by this technique through fault injection. The results of these experiments demonstrate significant improvements in reliability when large numbers of upsets occur. Arbitrarily increasing the number of partitions, however, provides diminishing returns as the reliability of the voters become dominant with large numbers of partitions.**

***Index Terms*—FPGA, reliability, SEU, MIU, TMR, Markov**

I. Introduction

SRAM-based field-programmable gate arrays (FPGAs) are becoming increasingly popular for space-based applications due to their high-throughput capabilities and relatively low cost. These SRAM-based devices, however, are susceptible to radiation-induced single-event upsets (SEUs). Many methods have been proposed to mitigate the effects of SEUs on SRAM FPGAs. The most popular technique used is triple modular redundancy (TMR).

TMR involves creating three redundant copies of a circuit and adding majority voters to choose the correct circuit output from the three copies. With this mitigation approach, a single-module failure will not cause an error in the circuit output, since the other two modules continue to operate correctly and will override the faulty module. TMR is thus often combined with configuration scrubbing in attempt to prevent multiple upsets from affecting the circuit at one time.

Configuration scrubbing, or simply *scrubbing*, is the periodic refresh of the contents of the FPGA configuration memory. Depending on the scrubbing rate and the rate of upsets in a particular environment, multiple upsets in the configuration can be almost completely avoided [1].

In the case of mismatched scrubbing and upset rates, however, multiple independent upsets (MIUs) may affect the circuit at one time. In this case, TMR is sometimes insufficient for protecting the circuit. In harsh radiation environments (or during temporary harsh conditions such as solar flares) where upsets may occur too frequently to adequately protect with scrubbing, a different mitigation technique may be necessary.

This work was supported by the Department of Energy NA-22 Office of Nonproliferation Research and Development at Los Alamos National Laboratory. Approved for public release under LA-UR-08-XXXX; distribution is unlimited.

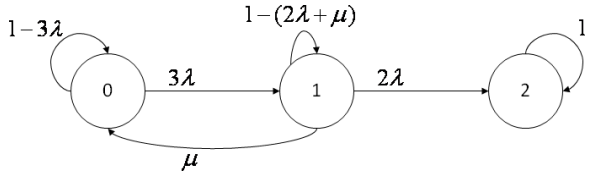


Fig. 1. Markov model showing TMR accompanied by configuration scrubbing

Figure 1 shows a Markov model of a system using TMR with scrubbing. Each node in a Markov model represents a particular state the circuit is in from a reliability point of view. In this case, State 0 represents the circuit operating correctly. State 1 represents the state in which one of the three TMR *domains* is operating incorrectly. State 2 represents the *failure* state in which two or more of the three TMR domains are operating incorrectly.

In the Markov model, arcs leaving a particular state are labeled with a value or expression. This value is the conditional probability of transitioning from the state at the beginning of the arc to the state at the end of the arc, assuming having started in this first state. The variables λ and μ represent the failure rate of the unmitigated system and the repair rate of the FPGA configuration (the scrubbing rate), respectively.

In this model, an SEU can cause a fault in one of the three TMR domains of the circuit and move the circuit from State 0 to State 1. The probability of failure of one of these domains is equal to the failure rate of the original circuit, λ. The probability of this transition, then, is equal to 3λ since TMR requires three copies of the original circuit. Configuration scrubbing repairs any faults (at the standard repair rate, μ) that exist in the FPGA and restores the circuit back to State 0. Alternately, a second SEU could affect another domain and cause a transition from State 1 to State 2. This transition probability is 2λ since only two of the redundant modules need be considered. Since the circuit has failed at that point and its output was incorrect, the model represents no exit from State 2. Thus the probability of transitioning from State 2 to State 2 is one.

Note that the transition from State 1 to the failure state requires a second SEU in a domain distinct from that which was affected in the transition from State 0 to State 1. This means that some MIUs are correctly masked by TMR. As long as only one TMR domain is affected, the TMR circuit will continue to operate correctly, albeit in State 1 of the Markov model.

This paper will demonstrate a method of modifying TMR such that more protection is provided for MIUs. By adding more frequent voting to a TMR circuit, we can partition the circuit into more than just the three TMR domains. This makes it less likely that a second (or third, etc.) upset will cause the triplicated circuit to fail, making this version of TMR more robust than the traditional view.

II. Related Work

TMR with more frequent voting has been examined before. Gurzi calculated the maximum reliability gains achievable by using extra voters to partition a circuit with TMR [2]. This paper also showed that the maximum reliability gains would be achieved with equally-sized partitions, or in other words, that the reliability of each partition should be equal.

Kastensmidt, et al. examined using voters to partition a circuit with TMR in FPGAs [3]. The paper focused on providing protection against domain-crossing events (DCEs), where a single upset could affect two different TMR domains. In an FPGA certain single-bit upsets in the routing configuration can affect multiple redundant copies of a circuit, in effect breaking the assumption that TMR can fully protect a circuit against single faults. The paper showed that by partitioning a TMR circuit with voters, many of these DCEs could be avoided.

Bolchini, et al. partitioned a circuit with voters and error detectors in order to increase the performance of configuration scrubbing [4]. The circuit is divided into sections and TMR applied to each section individually. The outputs of each section are examined for errors and only the configuration of the section in error is repaired using partial reconfiguration. This can result in more efficient repair of faults in the FPGA memory due to increased fault-detection rate and reduced configuration time.

This paper will focus on the benefits of TMR with more frequent voting for the purpose of mitigating the effects of *multiple independent upsets* (MIUs). These are separate, statistically independent single-event upsets which affect the target FPGA. When multiple upsets affect a circuit before configuration scrubbing is able to repair the circuit, multiple faults exist in the circuit. As TMR is not designed to handle multiple failures, the circuit may not operate correctly in this circumstance. This paper will show the benefits of partitioning the circuit into multiple sections in order to protect against MIUs.

III. More Frequent Voting for MIU Resilience

Traditional TMR is designed to protect against a single fault in the protected circuit at a time. Though some MIUs will be masked by TMR, many can break TMR by causing faults in multiple redundant circuit modules. TMR with more frequent voting can protect against many more MIUs by creating partitions in the triplicated circuit.

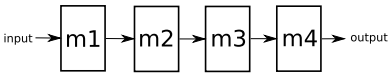


Fig. 2. A simple, unmitigated sample circuit. Each box represents a module in the circuit.

To illustrate, a sample circuit is shown in Figure 2. When applying traditional TMR, the circuit components are triplicated and a majority voter placed at the output of the circuit in order to provide a correct output even if one of the branches (or domains) of TMR is operating incorrectly. The circuit with standard TMR applied is shown in Figure 3.

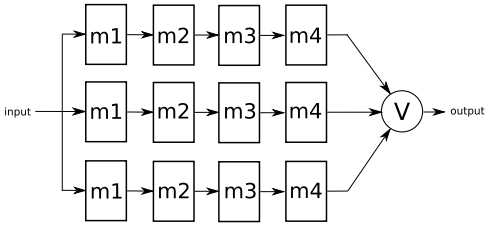


Fig. 3. The sample circuit with TMR applied

The circuit in Figure 4 shows the same circuit with more frequent voting. Voters are added periodically throughout the design, breaking up the triplicated circuit into partitions. Each partition is, in a sense, a separate entity from the other partitions in terms of reliability. The next section illustrates this property.

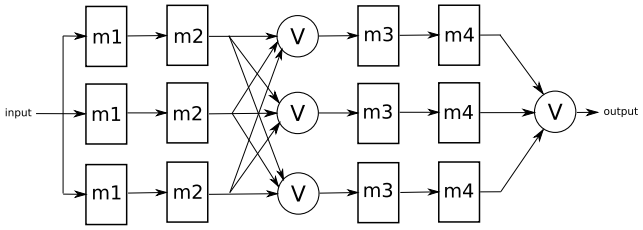


Fig. 4. The sample circuit with TMR using two partitions

Figures 5 and 6 demonstrate the major advantage of TMR with more frequent voting. In Figure 5, a set of upsets affect the sample circuit with traditional TMR. The upsets, which affect multiple TMR domains, overwhelm TMR and the triplicated circuit fails. In Figure 6, upsets affect the same portions of the circuit as in the first example, but the partitioning successfully protects the circuit.

Note how the partitions created by the voters isolate each section of the circuit. Errors are contained in that section due to the data correctly provided by the voters. For this circuit to fail, two different TMR domains in a single partition must be affected by one or more upsets. Depending on the size of the partitions, this can be made much more unlikely than upsets affecting two full TMR domains.

It is also worth mentioning that standard TMR is able to handle some sets of multiple upsets. In Figure 5, notice that the upsets affecting modules $m3$ and $m4$ alone are not enough

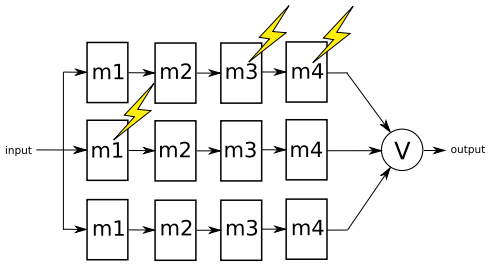


Fig. 5. Multiple errors in the sample circuit with TMR

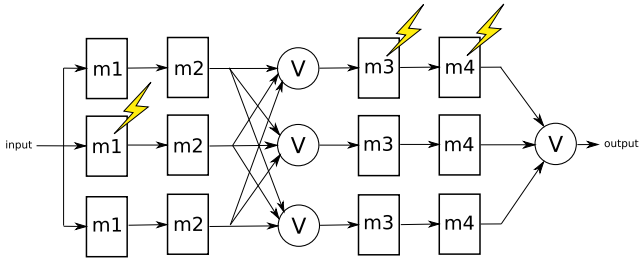


Fig. 6. Multiple errors in the sample circuit with TMR with more frequent voting

to overwhelm TMR. Standard TMR can essentially handle any number of upsets as long as they are contained in a single TMR domain. TMR with more frequent voting works in a similar manner, but allows more sets of multiple upsets due to the partitioning of the circuit.

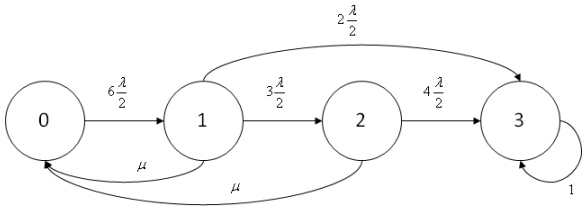


Fig. 7. Markov model showing TMR with more frequent voting - two partitions

Figure 7 shows a Markov model for a TMR system with two partitions. State 0, again, is the correct operation state and State 3 is the failure state. The transition from State 0 to State 1 occurs with probability $6\frac{\lambda}{2}$ since there are now essentially six parts to the circuit, each with a failure rate of $\frac{\lambda}{2}$. The original circuit is partitioned in two, hence $\frac{\lambda}{2}$, and triplicated.

Similar to the standard TMR case, State 1 indicates that a single module in a single partition is operating incorrectly. A transition from this state to the failure state is possible in the event that a second SEU causes another TMR domain *in the same partition* to fail. This would result in that partition operating incorrectly and thus the failure of the circuit to maintain correct outputs. Since only two of the six pieces of the circuit matter in this case, this transition probability is $2\frac{\lambda}{2}$.

State 2 represents the state in which the circuit has a module in error in each *partition*. In this case, the redundant circuit will still operate correctly since each partition will produce correct outputs. Scrubbing can repair the circuit and return the system to State 0. An additional upset in either of the partitions may cause the circuit to transition to the failure state if it results in a second module failure in one of the partitions. Thus, the two modules of the circuit that already have errors are ignored, and the transition probability to the failure state is $4\frac{\lambda}{2}$.

This same model may be extended to any number of partitions. It is clear that as more partitions are created, the probability of independent upsets causing system failure decreases. With more partitions, there is more isolation between the different areas of the circuit and less chance that MIUs would cause a particular partition to fail.

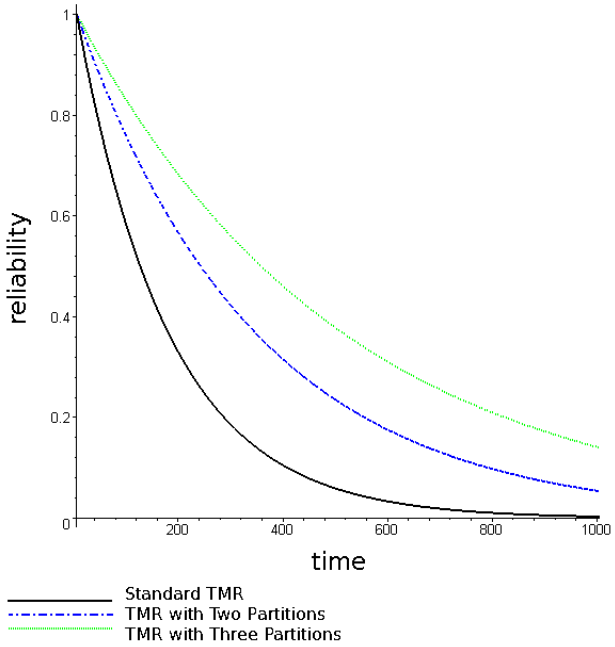


Fig. 8. Reliability comparison between three different versions of mitigation: standard TMR, TMR with two partitions, and TMR with three partitions.

Figure 8 illustrates the reliability advantages of using more frequent voting in a TMR system. The reliability of standard TMR (a single partition) is compared against that of two-partition and three-partition TMR systems. In these examples the system was divided into equal partitions, as suggested by Gurzi. The failure rate of the original circuit is set at $\lambda = 0.001$. This failure rate is divided equally to calculate the failure rate of smaller partitions. The repair rate (the scrubbing rate) is defined as μ = 0.1.

IV. Test Methodology

In order to test the effectiveness of TMR with more frequent voting on SRAM FPGAs, we used a fault injection tool previously developed and described in detail in [5]. This fault injection tool has been shown to have very high accuracy in predicting the sensitivity of an FPGA design [6].

The fault injection tool used is based on the SLAAC-1V FPGA board, which contains three identical Virtex 1000 FPGAs on a PCI card. The first FPGA is loaded with the configuration of the design under test (DUT). The second FPGA is loaded with the original, unmitigated design, called the *golden*. The third FPGA contains a pseudo random input generator (a linear-feedback shift register) which provides

identical inputs to the DUT and the golden. This third FPGA also contains a comparator which receives and compares the outputs of the first two FPGAs. The fault injection tool, running on the host PC, injects faults into the configuration memory of the DUT FPGA. When the third FPGA sees that the outputs of the DUT and golden do not match, the error is recorded.

Previous experiments with this tool tested every configuration cell in the DUT FPGA to obtain a complete sensitivity map of the particular design [5]. For these experiments, the tool was modified such that any number of faults could be injected into the configuration memory at one time. Using multiple upsets each test iteration, similar to a scrubbing cycle, the fault injection tool can estimate the sensitivity of the mitigated design in environments with higher upset rates.

The modified fault injection tool was used to measure the sensitivity of each test design using different numbers of upsets in each test iteration. The timeline for each test iteration was as follows:

1) Run the DUT and golden FPGAs with identical random inputs to insure matching functionality.
2) Using a Poisson random number generator with a fixed mean, choose a number of faults, N, to inject into the DUT.
3) Inject a set of faults in the DUT FPGA by choosing N random configuration bits (independently chosen) and inverting the contents of each.
4) Run the DUT and golden FPGAs with identical random inputs and observe the outputs.
5) If the outputs of the DUT and golden do not match at any point:
   a) Record the number of iterations since the last failure.
   b) Reset the `iterations_since_failure` count.
6) Repair the configuration of the DUT by inverting the contents of each of the N randomly-chosen configuration bits.

These steps were repeated until 10,000 failures had been recorded. Note that the choice of each configuration bit is done independently of any other choice so that the set of upsets models MIUs that would be encountered in a real radiation environment. Also note these upsets are chosen out of the entire set of DUT configuration bits, not just those that implement active logic.

Before a test is run on a particular design, an average number of upsets per iteration is chosen. This number is used as the mean for a Poisson random number generator to chose the number of faults to inject during each iteration. A new random number is selected for each test iteration. The Poisson distribution can be used to model the number of events occurring withing a given time interval if these events occur with a known average rate and independently of the time since the last event. **TODO: Insert Poisson reference here? (xapp987?)**

After injecting a number of faults, the designs on the DUT and golden FPGAs were allowed to run for a time (at least 1 ms at a clock rate of 10 MHz) while the third FPGA watched for mismatches in the outputs. When a mismatch occurred, the number of iterations that passed before the error was recorded. The sensitivity of the design to that particular average number of upsets is simply the number of failures recorded divided by the total number of test iterations.

This test methodology also gives the benefit of knowing how long it took for a design to fail in terms of scrub cycles. The `iterations_since_failure` counts recorded by the fault injection tool can then be used to create a histogram of the number of failures observed by the number of scrub cycles passed before each failure. This histogram can then be used to estimate the reliability function for a design, as will be shown in Section VI.

V. Experimental Results

The test design used for the results presented here is a simple shift register. The shift register is 1 bit wide by 3,000 deep. This simple structure of this design made it easy to insert voters anywhere desired into the triplicated version of the design. Thus it was possible to experiment with many different numbers of partitions.

The plot in Figure 9 shows a subset of the data obtained through our fault injection experiments. It compares the reliability (the y axis) of three different versions of the mitigated test design: standard TMR (one partition), TMR with two partitions, and TMR with four partitions. The y axis measures the reliability of the design in terms of the percentage of test iterations (scrub cycles) that resulted in system failures. Thus lower numbers represent higher reliability. The x axis shows different runs of the fault injection tool, each using a different number of upsets injected per test iteration. The exact numbers tested are shown in Table I.

This subset of the results illustrates that the reliability gains obtained by using TMR with more frequent voting depends heavily on the number of upsets per scrub cycle. Increasing the number of upsets injected per test iteration is similar to placing the test device in a harsher radiation environment. In both cases the rate of upsets relative to the scrubbing rate increases. The plot in Figure 9 shows that adding partitions to a TMR circuit increases the reliability of the circuit and that the increase in reliability is more pronounced for higher upset rates.

Figure 10 shows the full set of results in a different view. The y axis again measures the reliability of each design point. In this plot, the x axis shows the different mitigated designs by the number of partitions in each. Again, the designs tested and the numerical results are shown in Table I. The trend lines in the plot correspond to the upset rates of the fault injection experiments.

This plot illustrates the difference in reliability gains as a function of the upset rate of the environment. For higher upset rates, the difference in the reliability of the standard TMR circuit and the circuit with 25 partitions is much greater than for lower upset rates. This plot indicates that the gains of this technique are likely to be significant in environments with high upset rates, but that the gains are modest in those with

TABLE I

TABLE SHOWING THE RESULTS FROM THE FAULT INJECTION EXPERIMENTS. THE NUMBER OF UPSETS SHOWN IS THE AVERAGE NUMBER OF CONFIGURATION UPSETS INJECTED PER ITERATION.

† DUE TO A LIMITATION IN THE TEST FIXTURE, THESE EXPERIMENTS DO NOT INCLUDE TRIPLICATION OF THE CLOCK SIGNAL.

Partitions	Slices	FFs	LUTs	Failure Rate - 1 upset	Failure Rate - 2 upsets	Failure Rate - 4 upsets	Failure Rate - 8 upsets	Failure Rate - 16 upsets	Failure Rate - 24 upsets
1	9729	9768	39	0.0882%	0.2877%	1.0086%	3.4059%	10.793%	20.430%
2	9729	9768	42	0.0555%	0.1683%	0.5694%	1.9447%	6.7048%	13.243%
3	9729	9768	45		0.1222%	0.4077%	1.3970%	4.9213%	10.016%
5	9729	9768	51	0.0300%	0.0850%	0.2666%	0.8967%	3.1483%	6.5836%
10	9729	9768	66	0.0210%	0.0516%	0.1522%	0.4911%	1.7470%	3.6683%
25	9729	9768	111		0.0363%	0.0939%	0.2656%	0.8433%	1.7536%
50	9729	9768	186	0.0156%	0.0343%	0.0786%	0.1968%	0.5607%	1.0959%
100	9729	9768	339	0.0197%	0.0377%	0.0799%	0.1790%	0.4484%	0.8016%
1000	9729	9768	3039	0.0189%	0.0346%	0.0699%	0.1420%	0.2939%	0.4769%
1500	9731	9768	4539	0.0207%	0.0371%	0.0755%	0.1549%	0.3124%	0.5027%
3000	9737	9768	9042	0.0174%		0.0716%	0.1434%	0.2866%	0.4669%

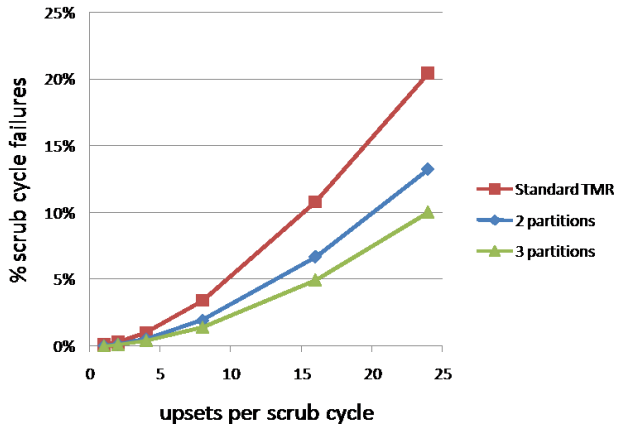


Fig. 9. Data obtained from fault injection showing the reliability gains of using TMR with more frequent voting by the number of upsets injected per scrub cycle.

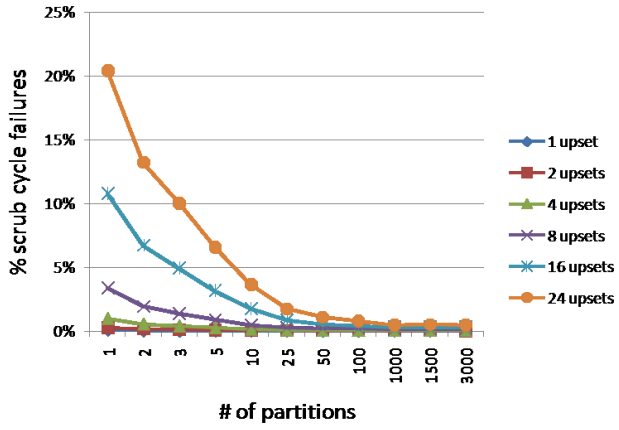


Fig. 10. Data obtained from fault injection showing the reliability gains of using TMR with more frequent voting by the number of partitions created.

low upset rates. In fact, for the single upset case, there are no reliability gains for adding more frequent voting to the TMR design. This makes sense, however, since standard TMR is designed to handle any single-bit upset. The advantages of TMR with more frequent voting lie in resiliency against multiple upsets in a single scrub cycle.

The plot in Figure 10 also suggests diminishing returns in reliability gains when adding more partitions. For each increase in the number of partitions, the increase in reliability is not as significant. Thus it may be most useful to add a relatively small number of partitions to a TMR design for the optimal cost/benefit trade-off. This, of course, will most likely be design-dependent.

TODO: Update this with the new numbers As an example of the reliability gains achieved in these tests, examine the 4 upsets per scrub cycle case. The design with standard TMR (one partition) shows a failure rate of 0.0788%. By adding 6 partitions to the design, that failure rate is reduced by about 35% to 0.0508%. Adding the same number of partitions in the 8 upsets per scrub cycle case results in an improvement in the failure rate by nearly 50%.

VI. EXPERIMENTAL VS. THEORETICAL RESULTS

The experimental results presented in the previous section can be directly compared theoretical foundation explained in Section I. As mentioned in Section IV, the data from the fault injection tool can be used to create a histogram of the failures over time. Figure 11 shows this histogram for the standard TMR case with an average of 8 upsets per iteration. Each bin in the histogram represents the number of times a failure was recorded after that number of scrub cycles in the fault injection experiment. The histogram consists of 10,000 entries, which was the number of failures recorded for each experiment.

The histogram of failures can then be converted into an estimate of the probability mass function (PMF) of failures simply by dividing the entries for each bin by the number of entries, in this case 10,000. The PMF describes the probability that a failure will occur after x scrub cycles for a particular design and upset rate. This estimated PMF can then be used to compute the estimated cumulative distribution function (CDF). The CDF, $F(x)$, is related to the PMF, $f(x)$, by the following equation:

$$F(x) = \sum_{k=0}^{x} f(x). \tag{1}$$

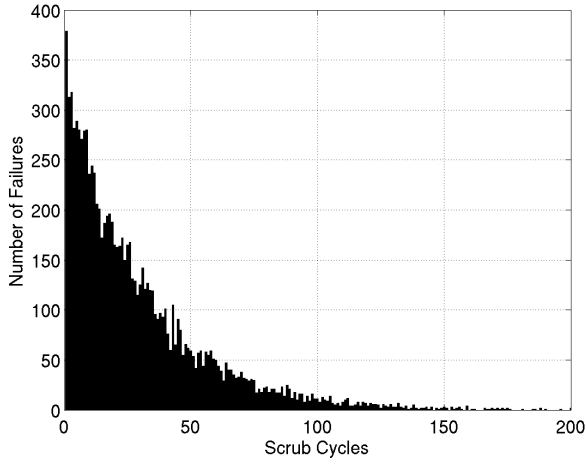


Fig. 11. Histogram showing the number of failures occurring after a certain number of scrub cycles for standard TMR (one partition) with an average of 8 upsets per iteration.

In turn, the reliability function, $R(x)$, can be directly computed from the CDF:

$$R(x) = 1 - F(x). \tag{2}$$

Figure 12 shows the estimated reliability function for three levels of TMR partitioning with the theoretical reliability function computed using the Markov models described in Section I. Both plots were created assuming an average of 8 upsets per iteration. For the theoretical models, the parameters μ and λ were chosen to match the experimental parameters. The repair rate, μ, was set to 1 to produce a time scale in terms of scrub (repair) cycles. The failure rate, λ, was calculated as eight times the sensitivity (as measured by the fault injection tool) of a single TMR domain in the standard TMR design (8×0.01789). This sets the upset rate at 8 upsets per scrub cycle and takes the sensitivity (the dynamic cross section) of the design into account since a single upset only has a 1.789% chance of causing an error in this design.

Figure VI shows the difference between the theoretical and experimental reliability curves (the experimental minus the theoretical) for the standard TMR (one partition) case. The reliability curve for the theoretical model is almost always lower than the experimental curve. The experiments thus measured a higher reliability than the model predicted. It is unclear at this point why the model is overly pessimistic. There are, of course, properties that are unaccounted for in the model. For example, due to limitations in the fault injection test fixture, the clock lines for these designs could not be triplicated. This deviation is not accounted for in the general theoretical model presented in Section I.

Figures VI and VI show similar differences for the cases with two and three partitions, respectively. Overall, the experimental data show trends similar to the theoretical results. Future research may reveal the precise reasons for the differences between the two sets of results.

VII. Conclusion

This paper has presented a method for increasing the reliability of FPGA designs in the presence of radiation-induced single-event upsets. It has evaluated the use of TMR with more frequent voting to increase design resilience against multiple independent upsets (MIUs). This technique effectively breaks up the triplicated circuit into self-contained partitions. Each partition is protected with TMR and is isolated from the rest of the partitions in the design. SEUs in other partitions do not affect the ability of a partition to operate correctly. This increases the reliability of the design by allowing more faults to exist in the circuit than possible with standard TMR.

The experimental results presented demonstrated that adding partitions to a TMR design does, indeed, increase the reliability of a sample FPGA design. The experiments showed that adding more partitions gives higher reliability gains in the presence of high upset rates rather than low rates. The results presented also show that the incremental reliability improvements diminish as more partitions are added. The optimal number of partitions will most likely depend on both the design and the radiation environment.

This approach does not come without limitations. The major downsides to TMR with more frequent voting are the area and delay added by the voting circuitry. Each partition created requires a set of voters, which use FPGA resources, as well as routing for the voter signals, which may add routing congestion to the design. In addition, the timing of the design is altered when adding these voters and extra routing. The extra delay through these components may also reduce the maximum clock speed of circuit.

To further study this technique and its possible benefits to FPGA reliability, we would like to perform a number of follow-on experiments. We would like to test this technique on newer architectures such as the Xilinx Virtex 4. Additional, more complex test designs could give more insight into the flexibility of this method. A detailed timing analysis of the test circuits would give insight into the timing consequences of more frequent voting. Also, we would like to do radiation testing in order to validate the fault injection experiments.

References

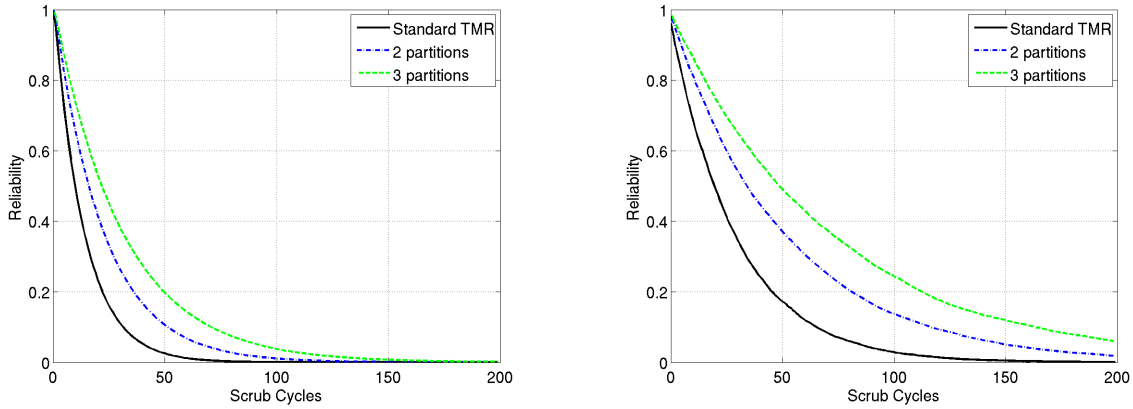
[1] C. Carmichael, M. Caffrey, and A. Salazar, "Correcting single-event upsets through Virtex partial configuration", Tech. Rep., Xilinx Corporation, June 1, 2000, XAPP216 (v1.0).

[2] K. Gurzi, "Estimates for best placement of voters in a triplicated logic network", *Electronic Computers, IEEE Transactions on*, vol. EC-14, no. 5, pp. 711–717, Oct. 1965.

[3] F. L. Kastensmidt, L. Sterpone, L. Carro, and M. S. Reorda, "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs", in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, Washington, DC, USA, 2005, pp. 1290–1295, IEEE Computer Society.

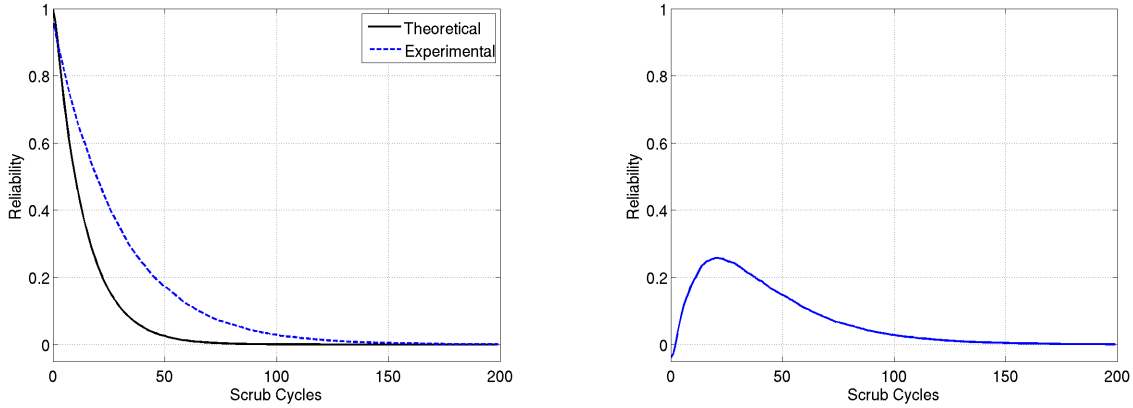
[4] C. Bolchini, A. Miele, and M. Santambrogio, "TMR and partial dynamic reconfiguration to mitigate SEU faults in FPGAs", *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT '07. 22nd IEEE International Symposium on*, pp. 87–95, 26-28 Sept. 2007.

[5] E. Johnson, M. J. Wirthlin, and M. Caffrey, "Single-event upset simulation on an FPGA", in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, T. P. Plaks and P. M. Athanas, Eds. June 2002, pp. 68–73, CSREA Press.



(a) Theoretical (b) Experimental

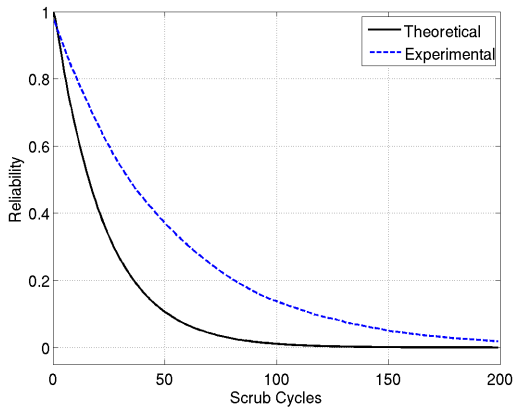
Fig. 12. Theoretical and experimental reliability curves for three levels of TMR partitioning with an average of 8 upsets per iteration.



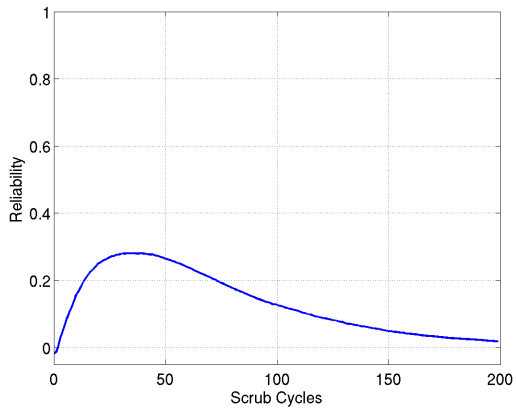
(a) Reliability curves (b) Difference between curves

Fig. 13. Comparison of the calculated theoretical reliability with the measured experimental reliability for standard TMR (one partition) with an average of 8 upsets per iteration.

[6] M. Wirthlin, E. Johnson, P. Graham, and M. Caffrey, "Validation of a fault simulator for field programmable gate arrays", in *Proceedings of the IEEE 2003 Nuclear and Space Radiation Effects Conference*, Monterey, CA, July 2003, IEEE, p. TBA, IEEE, Accepted.

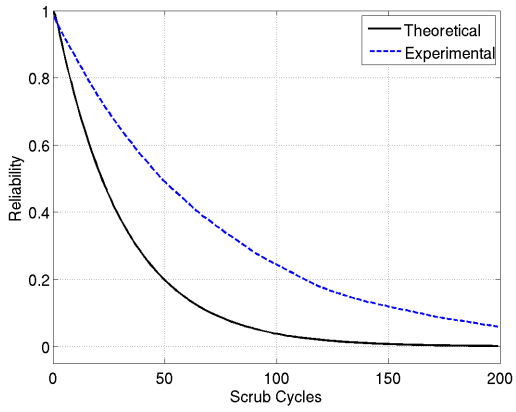


(a) Reliability curves

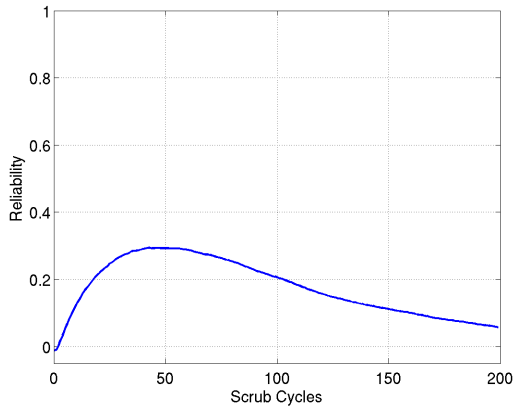


(b) Difference between curves

Fig. 14. Comparison of the calculated theoretical reliability with the measured experimental reliability for TMR with two partitions with an average of 8 upsets per iteration.



(a) Reliability curves



(b) Difference between curves

Fig. 15. Comparison of the calculated theoretical reliability with the measured experimental reliability for TMR with three partitions with an average of 8 upsets per iteration.