2008-12-11

Estimating TMR Reliability on FPGAs Using Markov Models

Daniel McMurtrey
dmcmurt@sandia.gov

Keith S. Morgan
keith.morgan@byu.net

Brian Pratt

Michael J. Wirthlin
wirthlin@ee.byu.edu

Estimating TMR Reliability on FPGAs Using Markov Models

Daniel McMurtrey, Keith Morgan, Brian Pratt, Michael Wirthlin

I. Background

A. Markov

A fundamental problem in fault-tolerant computing is predicting a system's reliability. Several methods exist to model system reliability. Markov modeling, named for the Russian mathematician Andrei Markov, is one such method.

The underlying assumption of Markov models is that the probability of a state transition depends only on the current state. These are called first-order Markov models. Other higher-order Markov models exist, but these are mathematically much more difficult to deal with. The systems analyzed in this paper can all be modeled as a first-order Markov process.

Since the conditional probability of being in any state j at time t in a first-order Markov process only depends on the previous state i, the conditional probability of transitioning from one state to the next can be assembled in a square matrix called a transition matrix $\mathbf{T}$. Each entry in the transition matrix at row m and column n represents the probability of a transition from state m to state n (where m can equal n). As such, all entries in a transition matrix are non-negative and the entries in each row must sum to unity. Multiplying the vector of marginal probabilities of the different states at time t will give the marginal vector of probabilities for the next time period [1].

The reliability of a computing system can be modeled as a very simple Markov process with two states—functional and failed. A graphical representation of this Markov model is shown in Figure 1. State 0 represents the functional state and state 1 represents the failed state. The system transitions from functional to failed at a rate of λ, as labeled on the arc from sate 0 to state 1.

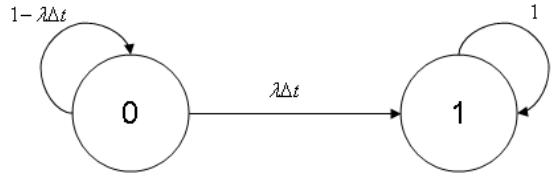

Fig. 1. A simple two-state Markov model.

The reliability function of the Markov model depicted in Figure 1 can be derived by first converting the process to a continuous-time model and then solving a set of differential equations for the probability the system is in state 1, the failed state, at time t.

To convert to a continuous-time Markov model, a transition matrix $\mathbf{T}$ is needed. By inspection, the transition matrix for Figure 1 is

$$\mathbf{T} = \begin{bmatrix} 1-\lambda\Delta t & \lambda\Delta t \\ 0 & 1 \end{bmatrix}, \tag{1}$$

where the entry in row m, column n represents the probability of transitioning from state m to state n. Notice that entry $1,1$ is one, meaning once the system is in the failed state it will remain there indefinitely.

From the transition matrix a set of equations can be defined that represent the probability of being in state 0 or 1 at time $t+\Delta t$. The distribution of probabilities at time $t+\Delta t$ is equal to the product of the distribution of the probabilities at time t multiplied by the transition matrix. Stated mathematically,

$$[p_0(t+\Delta t), p_1(t+\Delta t)] = [p_0(t), p_1(t)] \begin{bmatrix} 1-\lambda\Delta t & \lambda\Delta t \\ 0 & 1 \end{bmatrix}. \tag{2}$$

Multiplying out yields the system of equations

$$p_0(t+\Delta t) = (1-\lambda\Delta t)p_0(t) \tag{3}$$
$$p_1(t+\Delta t) = \lambda\Delta t p_0(t) + p_1(t). \tag{4}$$

Subtracting $p_0(t)$ from both sides of Equation 3, $p_1(t)$ from both sides of Equation 4, and dividing both sides of both equations by Δt gives

$$\frac{p_0(t+\Delta t) - p_0(t)}{\Delta t} = -\lambda p_0(t) \tag{5}$$
$$\frac{p_1(t+\Delta t) - p_1(t)}{\Delta t} = \lambda p_0(t). \tag{6}$$

Notice that the left hand side of Equations 5 and 6 are the definition of a derivative when $\Delta t \to 0$. Thus taking $\lim_{\Delta t \to 0}$ of Equations 5 and 6 generates

$$p_0'(t) = -\lambda p_0(t) \tag{7}$$
$$p_1'(t) = \lambda p_0(t). \tag{8}$$

Solving this set of equations is more easily accomplished if they are first converted to the LaPlace domain. Taking the LaPlace transform of Equations 7 and 8 yields

$$sP_0(s) - p_0(0) = -\lambda P_0(s) \tag{9}$$
$$sP_1(s) - p_1(0) = \lambda P_0(s) \tag{10}$$

where $p_i(0) = p_i(t)$ at $t = 0$. In matrix form, Equations 9 and 10 can be written as

$$[p_0(0), p_1(0)] = [P_0(s), P_1(s)] \begin{bmatrix} \lambda+s & -\lambda \\ 0 & s \end{bmatrix}. \tag{11}$$

Solving for $[P_0(s), P_1(s)]$ gives

$$[P_0(s), P_1(s)] = [p_0(0), p_1(0)] \begin{bmatrix} \lambda + s & -\lambda \\ 0 & s \end{bmatrix}^{-1}. \quad (12)$$

Computing the inverse of the matrix yields

$$[P_0(s), P_1(s)] = [p_0(0), p_1(0)] \begin{bmatrix} \frac{1}{\lambda+s} & \frac{\lambda}{(\lambda+s)s} \\ 0 & \frac{1}{s} \end{bmatrix}. \quad (13)$$

If the system starts out operational (in state 0), then the initial probability distribution is

$$[p_0(0), p_1(0)] = [1, 0]. \quad (14)$$

Substituting the values from Equation 14 into Equation 13 gives

$$[P_0(s), P_1(s)] = [1, 0] \begin{bmatrix} \frac{1}{\lambda+s} & \frac{\lambda}{(\lambda+s)s} \\ 0 & \frac{1}{s} \end{bmatrix}, \quad (15)$$

or after multiplying through,

$$P_0(s) = \frac{1}{\lambda + s} \quad (16)$$

$$P_1(s) = \frac{\lambda}{(\lambda + s)s}. \quad (17)$$

Taking the inverse LaPlace transform of Equations 16 and 17 results in

$$p_0(t) = e^{(-\lambda t)} \quad (18)$$

$$p_1(t) = 1 - e^{(-\lambda t)}. \quad (19)$$

In words, Equations 18 and 19 are the probability $p_i(t)$ that the system is in state i at time t. Since the system reliability $R(t)$ as a function of time is the probability that the system has survived up to time t, the reliability is simply the probability the system is in state 0 at time t, or

$$R(t) = p_0(t) = e^{(-\lambda t)}. \quad (20)$$

In more complex systems it is typically easier to compute the probability the system is in the failed state and subtract that from unity (1) to get the reliability function. Computing $R(t)$ in this manner yields

$$R(t) = 1 - p_1(t) = 1 - (1 - e^{(-\lambda t)}) = e^{(-\lambda t)}, \quad (21)$$

which is the same result.

B. Triple Modular Redundancy

Triple modular redundancy (TMR) is a reliability technique in which each module in a circuit is triplicated and a majority vote (two out of three) is taken on the outputs to determine the final module output. This technique is represented in Figure 2. In this figure, the three rows of rectangular blocks represent three "domains" of identical circuit modules. The three circles represent the 2-out-of-3 majority voters. The output from each domain is compared against the others and the final result determined. The output of the circuit is also triplicated.

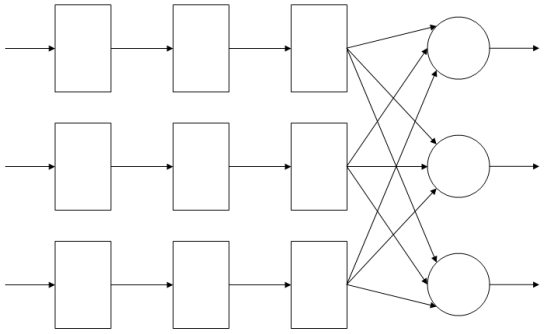

Fig. 2. A simplified view of triple modular redundancy (TMR). Rectangles represent logic blocks, circles represent voters.

TMR is able to mask any single fault (affecting only one module) in the circuit. In fact, TMR masks any number of faults that affect only one domain (only the modules in the top row of Figure 2, for instance). As long as only one domain has failed, the matching outputs of the other two will determine the output of the voter.

If, however, two faults occur in separate domains, the output of the voters may be incorrect. To prevent the system from failing, then, a failed module must be replaced or repaired before another fault occurs. In FPGAs, a method called "configuration scrubbing," or simply "scrubbing," is used to repair faulty modules. Scrubbing involves the periodic refresh of the contents of the FPGA configuration memory. This effectively replaces the modules that have failed by reinitializing the contents of the SRAM cells that define their operation. The scrubbing rate can be adjusted according to the rate at which faults are expected to occur.

II. TMR WITH REPAIR

Applying the Markov model to a system that has been implemented with TMR yields the graph shown in Figure 3. State 0 represents the state where all the TMR modules are functioning properly. State 1 represents the case where a fault has occurred in any of the three replicated modules. State 2 represents the state where more than one module has been affected by a fault. In state 2 the output is no longer dependable and the system is unreliable at this point. There is no recovery once multiple modules have failed.

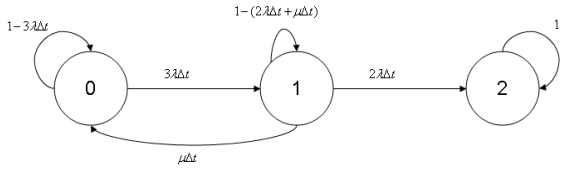

Fig. 3. Markov model representing TMR with repair through scrubbing.

For the system modeled in Figure 3, the failure rate λ is assumed to be identical for each of the three modules. Also, it is assumed that scrubbing can repair any module at rate μ. Since the modules all have an equal failure rate λ, the rate at

which the system can transition from state 0 to state 1 is 3λ, or the sum total of the failure rates of each of the individual modules. Once a single module has failed, the system is in state 1. From state 1, the system can transition to state 2 with rate 2λ since one of the other non-affected modules must have a fault for the total system to fail. However, if scrubbing occurs before a second module fails, the system would return to the fully functioning state 0. Scrubbing occurs at rate μ.

The transition matrix can be represented as

$$\mathbf{T} = \begin{bmatrix} 1-3\lambda\Delta t & 3\lambda\Delta t & 0 \\ \mu\Delta t & 1-2\lambda\Delta t-\mu\Delta t & 2\lambda\Delta t \\ 0 & 0 & 1 \end{bmatrix}. \quad (22)$$

Using the transition matrix, the transition equation is

$$[p_0(t+\Delta t), p_1(t+\Delta t), p_2(t+\Delta t)] = [p_0(t), p_1(t), p_2(t)]\mathbf{T}. \quad (23)$$

Expanding this equation produces

$$\begin{aligned} p_0(t+\Delta t) &= (1-3\lambda\Delta t)p_0(t) + \mu\Delta t p_1(t) && (24)\\ p_1(t+\Delta t) &= 3\lambda\Delta t p_0(t) + (1-(2\lambda+\mu)\Delta t)p_1(t) && (25)\\ p_2(t+\Delta t) &= 2\lambda\Delta t p_1(t) + p_2(t). && (26) \end{aligned}$$

Manipulating these equations produces

$$\begin{aligned} \frac{p_0(t+\Delta t)-p_0(t)}{\Delta t} &= -3\lambda p_0(t) + \mu p_1(t) && (27)\\ \frac{p_1(t+\Delta t)-p_1(t)}{\Delta t} &= 3\lambda p_0(t) - (2\lambda+\mu)p_1(t) && (28)\\ \frac{p_2(t+\Delta t)-p_2(t)}{\Delta t} &= 2\lambda p_1(t), && (29) \end{aligned}$$

with the familiar definition of a derivative on the left hand side of each equation. Taking the limit as $\Delta t \to 0$ yields the continuous-time domain equations

$$\begin{aligned} p_0' &= -3\lambda p_0(t) + \mu p_1(t) && (30)\\ p_1' &= 3\lambda p_0(t) - (2\lambda+\mu)p_1(t) && (31)\\ p_2' &= 2\lambda p_1(t). && (32) \end{aligned}$$

Applying the LaPlace transform to each equation produces

$$\begin{aligned} sP_0(s) - p_0(0) &= -3\lambda P_0(s) + \mu P_1(s) && (33)\\ sP_1(s) - p_1(0) &= 3\lambda P_0(s) - (2\lambda+\mu)P_1(s) && (34)\\ sP_2(s) - p_2(0) &= 2\lambda P_1(s). && (35) \end{aligned}$$

Combining equations 33, 34, and 35 yields

$$[p_0(0), p_1(0), p_2(0)] = [P_0(s), P_1(s), P_2(s)]\mathbf{A}, \quad (36)$$

where,

$$\mathbf{A} = \begin{bmatrix} s+3\lambda & -\mu & 0 \\ -3\lambda & \mu+s+2\lambda & 0 \\ 0 & -2\lambda & s \end{bmatrix}. \quad (37)$$

$$\mathbf{A} = \begin{bmatrix} s+3\lambda & -3\lambda & 0 \\ -\mu & \mu+s+2\lambda & -2\lambda \\ 0 & 0 & s \end{bmatrix}. \quad (38)$$

Solving for $[P_0(s), P_1(s), P_2(s)]$ by obtaining the inverse of $\mathbf{A}$ gives

$$[P_0(s), P_1(s), P_2(s)] = [p_0(0), p_1(0), p_2(0)]\mathbf{A}^{-1}, \quad (39)$$

where,

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{\mu+s+2\lambda}{\mu s+s^2+5\lambda s+6\lambda^2} & \frac{3\lambda}{\mu s+s^2+5\lambda s+6\lambda^2} & \frac{6\lambda^2}{s(\mu s+s^2+5\lambda s+6\lambda^2)} \\ \frac{\mu}{\mu s+s^2+5\lambda s+6\lambda^2} & \frac{s+3\lambda}{\mu s+s^2+5\lambda s+6\lambda^2} & \frac{2\lambda(s+3\lambda)}{s(\mu s+s^2+5\lambda s+6\lambda^2)} \\ 0 & 0 & \frac{1}{s} \end{bmatrix}. \quad (40)$$

Assuming the initial distribution of probabilities is $[p_0(0), p_1(0), p_2(0)] = [1, 0, 0]$ (meaning the system starts in the functional state with all modules fault-free), Equation 39 becomes

$$[P_0(s), P_1(s), P_2(s)] = [1, 0, 0]\mathbf{A}^{-1}, \quad (41)$$

which can also be represented as

$$\begin{aligned} P_0(s) &= \frac{\mu+s+2\lambda}{\mu s+s^2+5\lambda s+6\lambda^2} && (42)\\ P_1(s) &= \frac{3\lambda}{\mu s+s^2+5\lambda s+6\lambda^2} && (43)\\ P_2(s) &= \frac{6\lambda^2}{s\left(\mu s+s^2+5\lambda s+6\lambda^2\right)}. && (44) \end{aligned}$$

Solving for the reliability of the system requires determining the probability of being in states 0 or 1. However, it reduces the math to subtract the probability of being in state 2 from unity (1). Solving for the reliability $R(t)$, using the inverse LaPlace transform gives

$$\begin{aligned} R(t) &= 1 - p_2(t) && (45)\\ &= \frac{(\mu+5\lambda)sinh(\frac{1}{2}t\sqrt{\mu^2+10\lambda\mu+\lambda^2})e^{-\frac{1}{2}(\mu+5\lambda)t}}{\sqrt{\mu^2+10\lambda\mu+\lambda^2}} \\ &\quad + \cosh(\frac{1}{2}t\sqrt{\mu^2+10\lambda\mu+\lambda^2})e^{-\frac{1}{2}(\mu+5\lambda)t} && (46) \end{aligned}$$

These results were verified against the results obtained in [2]. Using the values of $\lambda = 0.001$ and $\mu = 0.1$ we were able to demonstrate the improvement in reliability of a system implemented with TMR. Figure 4 is a plot of the reliability of a non-redundant system and a TMR system as a function of time. The reliability of the non-redundant system drops immediately and quickly approaches 0. The system with TMR has a much higher reliability than the non-redundant system.

III. Persistence

While many FPGA applications cannot tolerate faults nor the corresponding interruption of service, there are applications which can tolerate a temporary interruption of service. By tolerating temporary interruptions of service, an application can trade availability for improvements in reliability.

In [3], [4], Morgan *et al.* showed that an FPGA application that implements scrubbing will experience both permanent and temporary fault-induced service interruptions. They called the permanent interruptions persistent errors and the temporary interruptions non-persistent errors. When a fault induces non-persistent errors the application becomes temporarily unavailable. Once scrubbing repairs the fault, the functional errors will eventually exit and the system will return to normal operation. When a fault induces persistent errors the application becomes permanently unavailable.

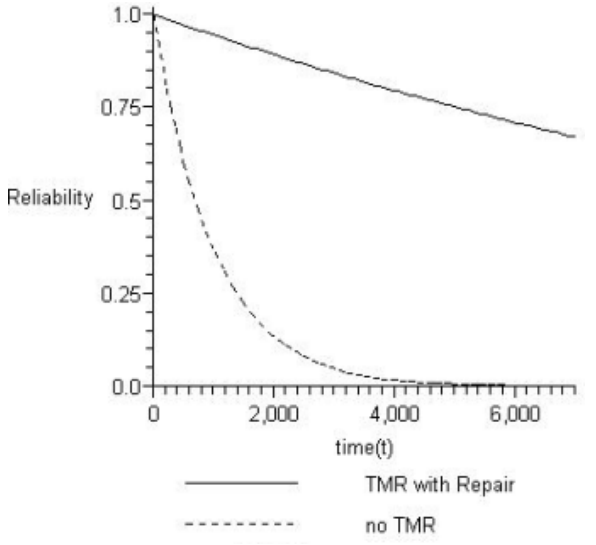

Fig. 4. Comparison of the Reliability of TMR vs. no TMR on a design. λ = 0.001 and μ = 0.1

Traditionally, FPGA application failure occurs after any interruption of service. By tolerating temporary service interruptions, an application will only fail after permanent service interruptions. To measure the improvement in reliability achieved by tolerating temporary service interruptions, a Markov model of a system that tolerates non-persistent errors was developed. The model is shown in Figure 5. State 0 is the functional state. State 1 is the unavailable state. State 2 is the failed state. In state 1, one or more non-persistent errors have occurred. In state 3 a persistent error has occurred. All state transition arcs are governed by the normal failure rate λ, the fraction p of the FPGA cross section susceptible to persistent errors and the scrubbing rate μ.

In state 0, the system will transition to state 2 when a persistent error occurs. On this arc the normal failure rate is scaled by the persistent cross section fraction p. On the other hand, the system will transition from state 0 to state 1 when a non-persistent error occurs. On this arc the normal failure rate is scaled by the non-persistent cross section fraction. Since the non-persistent and persistent cross sections are mutually exclusive, the non-persistent cross section fraction is simply $1 - p$.

In state 1, the system will transition back to state 0 when scrubbing repairs the fault that caused a non-persistent error. The rate on this arc is simply the scrubbing rate μ. On the other hand, the state can still transition to state 2, the failed state, if another fault induces a persistent error before scrubbing returns the system to state 0. Again this arc is governed by the failure rate, scaled by the persistent fraction of the sensitive cross section.

The transition matrix T for the Markov model in Figure 5 is

$$\mathbf{T} = \begin{bmatrix} 1-\lambda\Delta t & \lambda(1-p)\Delta t & \lambda p\Delta t \\ \mu\Delta t & 1-(\lambda p+\mu)\Delta t & \lambda p\Delta t \\ 0 & 0 & 1 \end{bmatrix}. \quad (47)$$

Using the process described in Section I-A, the reliability as a function of time was found to be

$$R(t) = e^{(-\lambda p t)}. \quad (48)$$

Notice the similarity to Equation 21, the reliability of a simple system without repair (repair of failures). Adding the concept of a persistent cross section to the Markov model simply scales the failure rate in the reliability equation by the persistent cross section fraction.

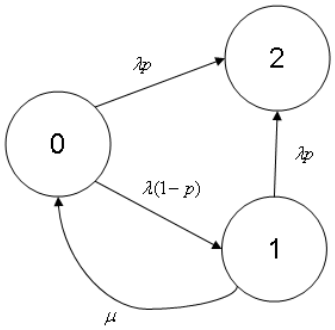

Fig. 5. Markov model of a system that only fails after a persistent error. State 0 is the functional state. State 1 is the unavailable state. In state 1, one or more non-persistent errors have occurred. State 2 is the failed state. In state 2 a persistent error has occurred.

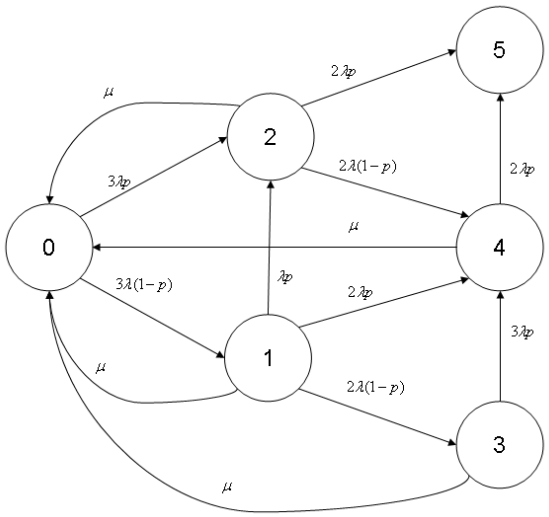

Fig. 6. Markov model of a system with TMR that only fails after a persistent error.

Applying the notion of persistence to TMR, produces the Markov model shown in Figure 6. As in previous discussions, λ is the rate at which faults occur in an individual module and μ is the scrubbing rate that allows the system to recover from faulty modules.

The states in Figure 6 are:

- State 0 - the system has no errors
- State 1 - one module has failed with a non-persistent error
- State 2 - one module has failed with a persistent error (the module could also have non-persistent errors as well)

- State 3 - two modules have failed with non-persistent errors
- State 4 - two modules have failed one with persistent error and at least one other with a non-persistent error (again the module with a persistent error could have other non-persistent errors)
- State 5 - two modules have failed with persistent errors, all the modules could have any number of non-persistent errors

Note that only in state 5 is the system failed to the point where scrubbing cannot correct the errors. Thus all states but 5 and 0 will transition back to state 0 if scrubbing occurs. These transition arcs are marked with the scrubbing rate μ. In state 0, the system will transition to state 1 if a non-persistent error has occurred in any of the modules. This occurs at a the sum of the error rates of each module scaled by the non-persistent fraction $(1-p)$. The system will transition to state 2 if a persistent error occurs. The probability of this happening is the sum of the error rates of the individual modules scaled by the persistent fraction p. From state 1, the system can move to state 2 if a persistent error occurs in the already failed module, or move to state 3 if a non-persistent error occurs in a functioning module. State 2 will transition to state 4 if a non-persistent error occurs in a functioning module or state 5 if the error is persistent. State 3 will transition to state 4 if a persistent error occurs in any of the three modules. State 4 will transition to state 5 if a persistent error occurs in any of the two modules that do not have persistent errors.

By inspection, the transition matrix for this Markov model is shown in Equation 49.

Applying the techniques demonstrated in Sections I-A and II we can solve for the reliability of this model. The technique used yielded multiple solutions. Figure 7 is a plot of a particular solution with $\lambda = 0.001, \mu = 0.1$ and $p = 0.1$. The plot shows that TMR does improve an FPGA application's reliability, specifically one that tolerates non-persistent errors. Comparing Figure 7 with Figure 4 shows designs which can tolerate non-persistent errors increase the reliability benefits gained by TMR.

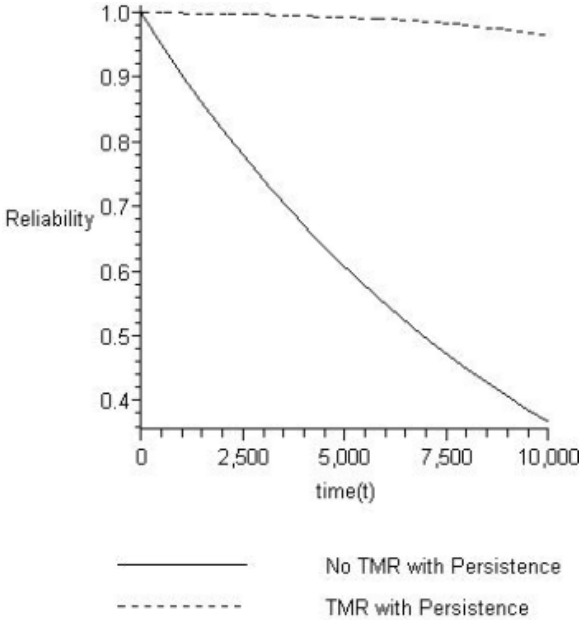

Fig. 7. Reliability of a TMR design vs. non-redundant design taking into account persistence.
$\lambda = 0.001$, $\mu = 0.1$, and $p = 0.1$

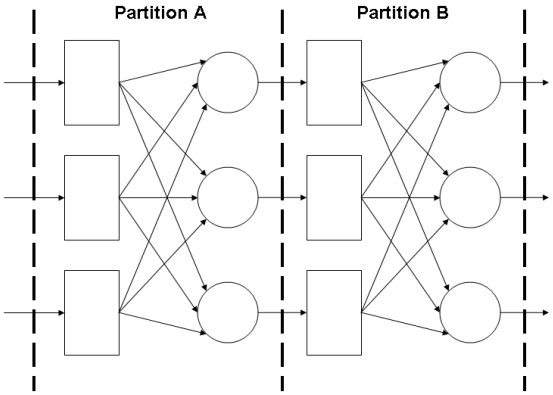

Fig. 8. A TMR system with multiple TMR partitions.

IV. TMR WITH PARTITIONS

It is sometimes desirable to improve an FPGA application's reliability by partitioning the circuit into smaller sections and voting on the outputs of each partition. Figure 8 shows how this is done. In this arrangement, the correct state of the circuit is restored from partition to partition, rather than allowing an error propagate through the entire circuit.

Adding partitions to TMR gives the advantage of higher reliability by masking more faults in the circuit than standard TMR. For example, if a fault occurred in module 1 of partition A and another fault occurred in module 2 of partition B, both faults would be masked and the outputs of both partition A and partition B would be correct. If a similar set of faults were to occur in the circuit of Figure 2, TMR would be defeated and the circuit output would be incorrect.

To estimate the reliability of the partitioned TMR circuit, we created the model illustrated in Figure 9 which corresponds to a circuit with two TMR partitions (as in Figure 8). State 0 represents the system when there are no faults present. State 1 represents the system after a fault occurs in a single module. State 2 represents the system when two modules in *separate* TMR partitions contain faults. State 3 is the failure state in which two faults are present in two different modules in the *same* TMR partition. λ represents the failure rate of both partitions of a single module. Note that states 0, 1, and 2 are all functional states and all faults are successfully masked.

Using the model of Figure 9, we construct the transition matrix T as

$$\mathbf{T} =$$

$$\mathbf{T} = \begin{bmatrix} 1-3\lambda\Delta t & 3\lambda(1-p)\Delta t & 3\lambda p\Delta t & 0 & 0 & 0 \\ \mu\Delta t & 1-(2\lambda(1-p)+3\lambda p+\mu)\Delta t & \lambda p\Delta t & 2\lambda(1-p)\Delta t & 2\lambda p\Delta t & 0 \\ \mu\Delta t & 0 & 1-(2\lambda p+\mu+2\lambda(1-p))\Delta t & 0 & 2\lambda(1-p)\Delta t & 2\lambda p\Delta t \\ \mu\Delta t & 0 & 0 & 1-(3\lambda p+\mu)\Delta t & 3\lambda p\Delta t & 0 \\ \mu\Delta t & 0 & 0 & 0 & 1-(2\lambda p+\mu)\Delta t & 2\lambda p\Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (49)$$

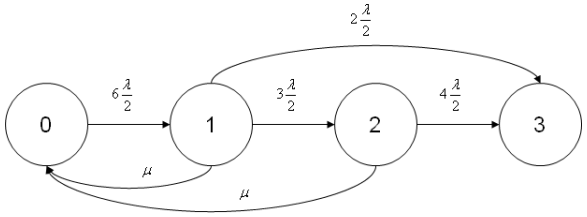

Fig. 9. Markov model representing a TMR system with two TMR partitions.

$$\begin{bmatrix} 1-3\lambda\Delta t & 3\lambda\Delta t & 0 & 0 \\ \mu\Delta t & 1-(\mu+\frac{5}{2}\lambda)\Delta t & \frac{3}{2}\lambda\Delta t & \lambda\Delta t \\ \mu\Delta t & 0 & 1-(\mu+2\lambda)\Delta t & 2\lambda\Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (50)$$

which results in the reliability function plotted in Figure 12.

For this example we are assuming the partitions are exactly equal in size.

The model in Figure 9 can be extended to include any number of TMR partitions in order to increase reliability further. Figure 10 shows the model as we have extended it to three TMR partitions. Figure 11 illustrates how the model could be extended to N partitions. The difference in reliability between systems with two and three TMR partitions in shown in Figure 12.

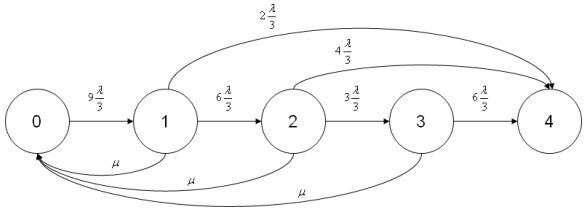

Fig. 10. Markov model representing a TMR system with three TMR partitions.

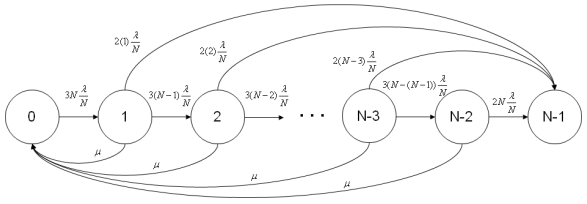

Fig. 11. Markov model representing a TMR system with N TMR partitions.

V. Conclusions

Although TMR, partitioned TMR, and non-persistent error tolerance empirically improve a system's reliability, it is good to derive a closed-form estimate of the reliability in order to accurately assess a particular system. In this paper, solutions for system reliability were derived for a simple computing system, a system with TMR and repair, a system which could tolerate non-persistent errors, a system with TMR and repair that could tolerate non-persistent errors and a system with partitioned TMR and repair. The results for the first four solutions are plotted together in Figure 13. Again, $\lambda = 0.001, \mu = 0.1$ and $p = 0.1$. The results indicate that significant improvements in reliability can be made with the techniques referenced throughout this paper.

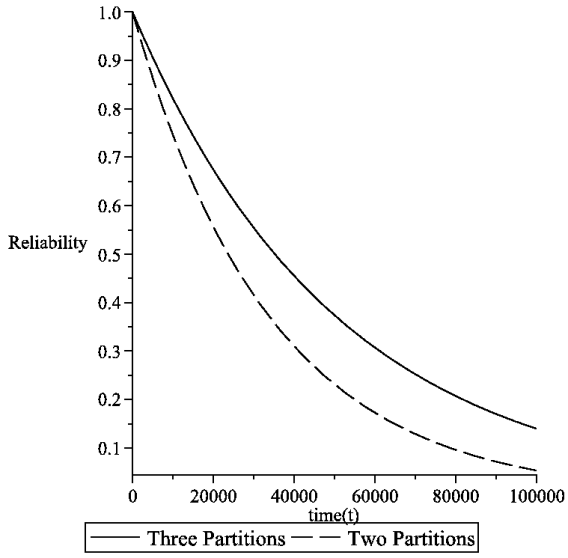

Fig. 12. Comparison of the Reliability of two TMR partitions vs. three TMR partitions on a design.
λ = 0.001 and μ = 0.1

References

[1] J. Lindsey, *Statistical Analysis of Stochastic Processes in Time*. Cambridge University Press, 2004.

[2] D. P. Siewiorek and R. S. Swarz, *"Reliable Computer Systems"*. A K Peters, 1998.

[3] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "Seu-induced persistent error propagation in FPGAs," *IEEE Transactions on Nuclear Science*, no. 6, DEC 2005.

[4] K. S. Morgan, "Seu-induced persistent error propagation in FPGAs," Master's thesis, Brigham Young University, August 2006.

[5] D. K. Pradhan, *"Fault-Tolerant Computer System Design"*. Prentice Hall, 1998.

[6] K. S. Trivedi, *"Probability and Statistics with Reliability, Queuing and Computer Science Applications"*. John Wiley & Sons, Inc., 2002.

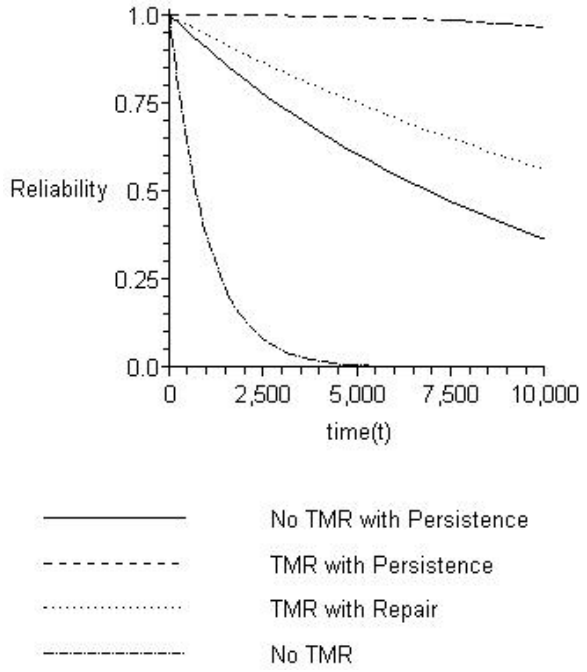

Fig. 13. Comparison of the Reliability of systems with different levels of mitigation and different levels of non-persistent error tolerance. $\lambda = 0.001$ and $\mu = 0.1$