Reliability Models for SEC/DED Memory with Scrubbing in FPGA-based Designs

Yubo Li, Brent Nelson, and Michael Wirthlin
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering
Brigham Young University, Provo, UT 84602
Email: liyubobuaa@gmail.com, nelson@ee.byu.edu, wirthlin@ee.byu.edu

Abstract—**Block RAMs on FPGAs are susceptible to radiation-induced SEUs, and are usually protected by fault-tolerant techniques such as SEC/DED code and/or scrubbing. Scrubbing can be divided into two categories based on the mechanism which controls the scrubbing interval. With deterministic scrubbing, each memory location is scrubbed (repaired) on a regular basis and the interval is fixed. With probabilistic scrubbing, a word can be checked and corrected any time it is accessed (read or written). In this case the scrubbing interval is exponentially distributed.**

This paper presents two MTTF models for SEC/DED memory with scrubbing. The first one considers only probabilistic write scrubbing, but takes into account *non-uniform* write rates for different memory locations. The second model combines both deterministic scrubbing *and* probabilistic write scrubbing into a single model. The proposed models provide more accurate MTTF estimates compared to prior models, and allow for insights into the impact of write rate and memory access distributions on memory reliability.

I. Introduction

Field Programmable Gate Arrays (FPGAs) are an attractive technology for space applications due to their reprogrammability and low NRE cost. However, SRAM-based FPGAs are susceptible to radiation-induced single-event upsets (SEUs) because they employ volatile memory for configuration data and user data storage.

Block RAMs (BRAMs) are widely used in FPGAs to store user data. BRAMs are smaller and more distributed than traditional memories. In addition, the depth and width of BRAMs in Xilinx FPGAs are configurable [1], [2], making them more flexible to fit the functionality of various modules or designs. For example, in digital signal processing (DSP) applications, different modules usually perform different operations on the data. This would result in different BRAM sizes and non-uniform write access patterns to the BRAMs. The impact of non-uniform write distributions on memory reliability will be discussed in more detail in Section II.

The reliability of BRAMs can be jeopardized by radiation-induced SEUs. Single error correction/double error detection (SEC/DED) codes are a common fault tolerant technique for memory systems. SEC/DED can detect any single-bit error in a memory word and correct it immediately, as well as detect but not correct double-bit errors. However, when a word is corrupted by a single-bit error, the data is only corrected on the output port, while the corrupted word remains erroneous in the memory storage. As errors accumulate in the memory, multiple faults in a word will eventually defeat SEC/DED. Thus scrubbing, another fault tolerant technique, is often employed along with SEC/DED to prevent a memory from accumulating a second error in a single word.

During *deterministic scrubbing*, each word of the memory is regularly read and checked for correctness. If a single-bit error is detected, it is corrected by SEC/DED and written back to the memory. An appropriate scrub interval can effectively prevent a memory from gathering a second error and therefore considerably improves the memory's reliability.

However, another kind of scrubbing exists and is called *probabilistic scrubbing* [3]. With probabilistic scrubbing, whenever a word is accessed by the circuit using the BRAM, the data is checked and corrected. Probabilistic scrubbing can be done any time a word is read, written, or both. For example, with probabilistic read scrubbing, the word would be read and, if an error was detected by the SEC/DED circuitry, another memory cycle would be used to write the corrected value back to memory. This obviously has performance ramifications by requiring additional memory write cycles from time to time, something that may interfere with normal circuit operation. However, on all memory write operations, new (and therefore correct) data is written into the memory. Hence write operations can be seen as a kind of probabilistic write scrubbing which requires no extra circuitry or clock cycles — that is, it comes for free.

Others have investigated the problem of memory reliability over the past decades [3], [4], [5], [6]. Both [3] and [6] have provided reliability/MTTF models for SEC/DED memory with deterministic scrubbing only. Saleh et al. [3] have provided a reliability/MTTF model for SEC/DED memory with probabilistic scrubbing only but do not differentiate between read and write scrubbing. Additionally, this model is limited because it assumes that all memory locations

This work was supported in part by the I/UCRC Program of the National Science Foundation within the NSF Center for High-Performance Reconfigurable Computing (CHREC), Grant No. 0801876.

have a uniform access (probabilistic scrub) rate, which is unrealistic for FPGA applications. In addition, none of the prior work has proposed models which combine both deterministic and probabilistic scrubbing in the same model.

This paper will investigate the reliability and MTTF of memories in FPGA-based designs by proposing two new MTTF models which improve on the prior work. The first one considers only probabilistic write scrubbing, but takes into account *non-uniform* write rates for different memory locations. The second model combines both deterministic scrubbing *and* probabilistic write scrubbing into a single model. The proposed models provide more accurate MTTF estimates compared to existing models, and will allow for insights into the impact of write rate and memory access patterns on memory reliability.

II. Reliability Models for SEC/DED Memory with Scrubbing

In this section, several previous models for SEC/DED memory with scrubbing are introduced and compared, and the motivations of the proposed models are discussed.

A. Saleh's Deterministic model

In [3], Saleh et al. proposed a reliability model for SEC/DED memories with deterministic scrubbing. This model was developed for caches in desktop computing devices rather than for embedded FPGA memories. Thus it is a different environment than FPGA applications, but the modeling process and some of their assumptions still apply.

Saleh's deterministic model made four assumptions:

- Transient faults occur with the Poisson distribution.
- All the bit flips are statistically independent.
- A second bit flip in a single word does not correct the first one.
- Every word is considered as an entity with error rate λN, where λ is the bit failure rate, and N is the number of bits in a single word.

The MTTF estimate of Saleh's deterministic model is given in Table I. In the MTTF equation, ν is the deterministic scrub rate enforced by the scrubber, and M is the number of words in the memory.

TABLE I: Saleh's Deterministic Model

Memory System	MTTF
SEC/DED with deterministic scrubbing	$\frac{2\nu}{M\lambda^2 N^2}$

B. Edmonds' Deterministic Model

Edmonds et al. [6] proposed another reliability model for SEC/DED memories with deterministic scrubbing. This model was created for BRAMs on a Xilinx FPGA, XQR5VFX130 [2], and therefore had a similar context to the proposed models of this paper.

The assumptions of Edmonds' deterministic model are the same as Saleh's probabilistic model, except for the fourth one. Edmonds' deterministic model assumes that every bit has an error rate of λ, thus the error rate of a word depends on the number of existing errors. Table II gives the equation of Edmonds' deterministic model.

TABLE II: Edmonds' Deterministic Model

Memory System	MTTF
SEC/DED with deterministic scrubbing	$\frac{2\nu}{M\lambda^2 N(N-1)}$

It is worth noting that although Saleh's deterministic model and Edmonds' deterministic model take different derivation approaches, they give almost the same results. Specifically, the ratio of the result given by Saleh's model to that of Edmonds' model is $N/(N-1)$. Since N is the number of bits per word, usually being 36 or 72, this difference is very small.

C. Saleh's Probabilistic Model

Saleh et al. also presented a model for SEC/DED memories with probabilistic scrubbing [3], following the same assumptions for Saleh's deterministic scrubbing. This model assumes that a word is read and checked whenever it is addressed (written or read) by the program in execution. It also assumes that the cache memory locations are somewhat evenly accessed due to the least recently used (LRU) approach used to manage lines in cache memories.

In Saleh's probabilistic model, μ represents the probabilistic scrub rate. Unlike the scrub interval in deterministic scrubbing, which is fixed, scrub interval in probabilistic scrubbing distributes exponentially with a rate of $1/\mu$. The mathematical expression of Saleh's probabilistic model is presented by Table III.

TABLE III: Saleh's Probabilistic Model

Memory System	MTTF
SEC/DED with probabilistic scrubbing	$\frac{\mu}{M\lambda^2 N^2}$

D. Motivations of the Proposed Models

In Section I, it has been pointed out that probabilistic scrubbing can occur when a word is being written or read. Although probabilistic read scrubbing has performance and area overhead by requiring additional circuitry and clock cycle to complete, probabilistic write scrubbing comes for free as new (and therefore correct) data will be written into the word on write operations. Saleh's probabilistic model does not differentiate between read and write scrubbing. Additionally, because of the context of cache memory in which it was developed, Saleh's probabilistic model assumes a uniform scrub rate for all the memory locations. However, this is not always true for FPGA applications. For example, in digital signal processing (DSP) applications, different modules usually perform different operations on the data. This would result in different BRAM sizes and non-uniform write access patterns to BRAMs. In order to study the reliability of the memory on FPGAs, a new reliability model will be proposed by this paper which considers only probabilistic write scrubbing, and which takes into account *non-uniform* write rates for different memory locations.

Furthermore, although Saleh's deterministic model and Edmonds' deterministic model both aim at memories with deterministic scrubbing, none of the prior work has proposed models which combine both deterministic scrubbing and probabilistic write scrubbing in the same model. As discussed above, probabilistic write scrubbing comes for free whenever a word is being overwritten and therefore should be included. The second model proposed by this paper, named the model for mixed scrubbing, will combine both probabilistic write scrubbing and deterministic scrubbing. Probabilistic read scrubbing is not considered because it requires performance and area overhead.

The main contribution of the proposed models is that they will allow for a more accurate and thorough understanding of memory reliability by providing MTTF estimates that are applicable to FPGAs. If a memory involves frequent write operations, it is likely to survive with a lower deterministic scrub rate or with no deterministic scrubbing at all. For example, deterministic scrubbing is not necessary for a FIFO because it already involves sufficient writes (probabilistic scrubbing). In this case, the proposed model for probabilistic write scrubbing can be used. Otherwise, if deterministic scrubbing is necessary as a supplement to probabilistic write scrubbing, the proposed model for mixed scrubbing can be applied. The two proposed models will suggest that the deterministic scrub rate may be lower than otherwise predicted because probabilistic write scrubbing can compensate for the difference. This suggests that designers may have previously over-engineered their memories because existing models gave them estimates that were way too conservative. The problem of over-scrubbing could lead to unnecessary waste of power and design effort.

Furthermore, different memory locations usually have different write rates. For example, rarely modified memory locations such as look-up tables may be written only once while locations in a RAM may be overwritten frequently. The difference in write rates among memory locations may have considerable impact on the reliability of a memory system. Consequently, it is important for the new model to reveal the relation between memory write distribution and reliability. By using per-word write rates, the proposed models will be able to give more accurate results than other models.

III. Proposed Model for Probabilistic Write Scrubbing

In this section, the proposed model for probabilistic write scrubbing will be derived and analyzed. Several insights revealed by this model will be discussed.

A. Mathematical Expression

The proposed models will be built on the following assumptions:

- Transient faults occur with a Poisson distribution.
- All the bit flips are statistically independent.
- A second bit flip in a word does not correct the first one.
- Every bit has an independent error rate λ.
- Every word i has its own write rate μ_i.

The first three assumptions are the same as those of Saleh's and Edmonds' deterministic model. The fourth assumption differs from that of Saleh's deterministic model, which considers each N-bit word as an independent entity. This will result in a more accurate model than Saleh's model. The fifth assumption is a new one that is introduced by the proposed model, and again will allow for a more accurate model and novel insights of the relation between write rates and reliability. This assumption, however, requires more parameters and a more detailed understanding on how the memory is used.

This subsection presents the deriving process of the proposed model for non-uniform probabilistic scrubbing. This memory system has non-uniform write distribution, meaning that the write rate is different for each word. When a memory location is overwritten, the new word can be considered as correct. Thus write operations can be considered as equivalent to repair processes. The memory fails whenever two errors accumulate in one word to defeat the capability of the SEC/DED protection. For a single word, both the failure and the repair processes occur continuously and independently, therefore their occurrences follow the Poisson distribution. The time interval between two writes or two bit errors in the same memory location is not fixed and can be modeled by exponential distribution. Thus, this random process can be well modeled using a Markov model.

1) Markov Model: The continuous-time Markov model of a single word i is given in Figure 1. Each state represents the number of errors in the word, and the directed arcs between states indicate the transition rate from one state to another. For example, if the total number of bits in the word is N, and each bit has the same failure rate λ, then the failure rate of the entire word is λN, which is labeled on the arrow pointing from state 0 to state 1. If the word is already in state 1, then a write operation will bring it back to state 0 because the single-bit error is corrected. Therefore the transition rate from state 1 to state 0 is the writing rate μ_i. State 2 is the failing state, or the absorbing state, of a word when SEC/DED is used, so there is no outgoing arc from state 2.

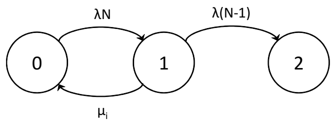

Fig. 1: State Transition Rate Diagram for Word i with write rate μ_i

From Figure 1 we can write the following transition matrix:

$$\left[\begin{array}{ccc} -\lambda N & \lambda N & 0 \\ \mu_i & -\lambda(N-1)-\mu_i & \lambda(N-1) \\ 0 & 0 & 0 \end{array}\right].$$

2) MTTF of a Single Word: The solution to this Markov chain can be obtained by defining $P_i(t)$ as being the probability of reaching state i in the time interval $[0, t]$. From the transition matrix, we can get the following set of differential equations (Chapman-Kolmogorov equations) which represents the dynamics of the probability system [3]:

$$\begin{aligned} \frac{dp_0(t)}{dt} &= -\lambda N p_0(t) + \mu_i p_1(t) \\ \frac{dp_1(t)}{dt} &= \lambda N p_0(t) - [\lambda(N-1)+\mu_i]p_1(t) \qquad (1) \\ \frac{dp_0(t)}{dt} &= \lambda(N-1)p_1(t). \end{aligned}$$

Taking the Laplace transform for the set of equations in Equation 1 gives

$$\begin{aligned} sp_0(s) - p_0(0) &= -\lambda N p_0(s) + \mu_i p_1(s) \\ sp_1(s) - p_1(0) &= \lambda N p_0(s) - [\lambda(N-1)+\mu_i]p_1(s) \quad (2) \\ sp_2(s) - p_2(0) &= \lambda(N-1)p_1(s). \end{aligned}$$

Then, by adjusting the set of equations in Equation 2, we get

$$\begin{aligned} p_0(0) &= (s+\lambda N)p_0(s) - \mu_i p_1(s) \\ p_1(0) &= -\lambda N p_0(s) + [s+\lambda(N-1)+\mu_i]p_1(s) \quad (3) \\ p_2(0) &= -\lambda(N-1)p_1(s) + sp_2(s). \end{aligned}$$

Therefore the vector $\vec{P(0)}$ can be expressed as

$$\vec{P(0)} = \vec{P(s)} \times \mathbf{A},$$

where

$$\mathbf{A} = \left[\begin{array}{ccc} s+\lambda N & -\lambda N & 0 \\ -\mu_i & s+\lambda(N-1)+\mu_i & -\lambda(N-1) \\ 0 & 0 & s \end{array}\right].$$

It is assumed that the word has no error at time 0. In other words, the probability that the word is in state 0 at time 0 is 1. Therefore we get

$$\begin{aligned} \vec{P(s)} &= \vec{P(0)} \times \mathbf{A}^{-1} \\ &= \left[\begin{array}{ccc} 1 & 0 & 0 \end{array}\right] \times \mathbf{A}^{-1}. \end{aligned}$$

Then $p_2(s)$ can be calculated as follows:

$$\begin{aligned} p_2(s) &= \frac{(-1)^4 \times det \left[\begin{array}{cc} -\lambda N & 0 \\ s+\lambda(N-1)+\mu_i & -\lambda(N-1) \end{array}\right]}{det\mathbf{A}} \\ &= \frac{\lambda^2 N^2 - \lambda^2 N}{(s+\lambda N)(s^2+\lambda s(N-1)+\mu_i) - \lambda\mu_i N s} \\ &= \frac{\lambda^2 N(N-1)}{s[s^2+(2\lambda N-\lambda+\mu_i)s+\lambda^2 N(N-1)]} \\ &= \frac{a}{s} + \frac{b}{s+\alpha_1} + \frac{c}{s+\alpha_2}, \qquad (4) \end{aligned}$$

where $\alpha_1 = \frac{\lambda(2N-1)+\mu_i-\sqrt{\lambda^2+2\lambda\mu_i(2N-1)+\mu_i^2}}{2}$ and $\alpha_2 = \frac{\lambda(2N-1)+\mu_i+\sqrt{\lambda^2+2\lambda\mu_i(2N-1)+\mu_i^2}}{2}$.

By solving Equation 4 for a, b, and c, we get

$$p_2(s) = \frac{\frac{\lambda^2 N(N-1)}{\alpha_1\alpha_2}}{s} + \frac{\frac{\lambda^2 N(N-1)}{\alpha_1(\alpha_1-\alpha_2)}}{s+\alpha_1} + \frac{\frac{\lambda^2 N(N-1)}{\alpha_2(\alpha_2-\alpha_1)}}{s+\alpha_2}. \qquad (5)$$

By taking the inverse Laplace transform for Equation 5, it gives

$$\begin{aligned} p_2(t) &= 1 + \frac{\lambda^2 N(N-1)}{\alpha_1(\alpha_1-\alpha_2)}e^{-\alpha_1 t} + \frac{\lambda^2 N(N-1)}{\alpha_2(\alpha_2-\alpha_1)}e^{-\alpha_2 t} \\ &= 1 + \frac{\alpha_2}{\alpha_1-\alpha_2}e^{-\alpha_1 t} + \frac{\alpha_1}{\alpha_2-\alpha_1}e^{-\alpha_2 t}. \qquad (6) \end{aligned}$$

By definition, $p_2(t)$ is the probability that the word is in state 2 after time interval $[0, t]$, and the reliability of the word is the probability that the word is in state 0 or state 1 at time instant t. Therefore we have

$$\begin{aligned} r(t) &= 1 - p_2(t) \\ &= \frac{\alpha_2}{\alpha_2-\alpha_1}e^{-\alpha_1 t} + \frac{\alpha_1}{\alpha_1-\alpha_2}e^{-\alpha_2 t}. \qquad (7) \end{aligned}$$

Then $MTTF_i$, the MTTF of the word is given by

$$\begin{aligned} MTTF_i &= \int_0^\infty r(t)\,dt \\ &= \frac{\alpha_2}{\alpha_2-\alpha_1}\int_0^\infty e^{-\alpha_1 t}\,dt + \frac{\alpha_1}{\alpha_1-\alpha_2}\int_0^\infty e^{-\alpha_2 t}\,dt \\ &= \frac{\alpha_1+\alpha_2}{\alpha_1\alpha_2} \\ &= \frac{\lambda(2N-1)+\mu_i}{\lambda^2 N(N-1)}. \qquad (8) \end{aligned}$$

3) MTTF of the Memory: According to definition, the MTTF of the entire memory with M words can be calculated by

$$MTTF = \int_0^\infty R(t)\,dt = \int_0^\infty \prod_{i=1}^{M} r(t)\,dt.$$

However, this equation leads to an extremely complex result which cannot be analytically derived. Therefore we go back to Equation 7 and seek for possible simplifications.

Note that when μ_i is many orders of magnitude greater than λ (which is true for FPGA applications, as long as the BRAM is being regularly written), α_1 will approach 0. In this case, we can make an approximation for $r_i(t)$, and we call the approximated value $ar_i(t)$:

$$ar_i(t) \approx e^{-\alpha_1 t}. \tag{9}$$

Calculations show that if μ is five orders of magnitude higher than λ, then $ar_i(t)$ can be viewed as a good approximation for $r_i(t)$. In this case, the reliability of word i follows an exponential distribution, and thus word i can be seen as an entity with a constant failure rate γ_i:

$$\begin{aligned}\gamma_i &= \frac{1}{MTTF_i} \\ &= \frac{\lambda^2 N(N-1)}{\lambda(2N-1)+\mu_i}.\end{aligned}$$

Consequently, the MTTF of the entire memory can be calculated by

$$\begin{aligned}MTTF &= \frac{1}{\sum_{i=1}^{M} \frac{1}{MTTF_i}} \\ &= \frac{1}{\lambda^2 N(N-1)\sum_{i=1}^{M}\frac{1}{\lambda(2N-1)+\mu_i}}.\end{aligned} \tag{10}$$

Note that if μ_i is not orders of magnitude higher than λ (in extreme cases like a ROM, μ_i could be zero), then Equation 10 is not accurate and should not be used.

Because the bit failure rate λ is usually orders of magnitude lower than the write rate μ_i, the term $\lambda(2N-1)$ in Equation 10 can be discarded. The simplified equation for MTTF can be expressed as

$$MTTF = \frac{1}{\lambda^2 N(N-1)\sum_{i=1}^{M}\frac{1}{\mu_i}}. \tag{11}$$

This approximation only has a tiny effect to the result, and makes the result more conservative.

B. Simplifying by Dividing Memory into Groups

Equation 11 requires *per-word* write rates. However, it may be feasible to categorize the words with similar write rates into one group, and then the words in the same group are considered to have the same write rate instead of individual ones. This would simplify the model and may help us better understand the characteristics of probabilistic write scrubbing. In order to investigate this feasibility, a special example is considered where a memory consists of only two words and each word has its own write rate (μ_1 and μ_2, respectively). Thus Equation 11 becomes

$$MTTF = \frac{1}{\lambda^2 N(N-1)} \cdot \frac{\mu_1\mu_2}{\mu_1+\mu_2}. \tag{12}$$

If the two words can be combined into one group with write rate of μ_1, then Equation 12 becomes

$$MTTF = \frac{1}{\lambda^2 N(N-1)} \cdot \frac{\mu_1}{2} \tag{13}$$

In Figure 2, the curve shows the reliability of the combination of the two words (Equation 12), and the horizontal line ($MTTF_{\mu_1}$) shows the MTTF when both words are considered to have the same write rate of μ_1 (Equation 13). The parameters used for this experiment are: $\lambda = 1.97 \times 10^{-11}$ upsets per bit-second, $N = 72$ bits, and $\mu_1 = 1$ per second.

In Figure 2, $\mu_1 = 1$ and μ_2 ranges from 0.8 to 1.2. When $\mu_1 = \mu_2 = 1$, Equation 12 and Equation 13 are equivalent and give the same result. When μ_2 is within the range of $\mu_1 \pm 10\%$ (between the two vertical dotted lines), the error introduced by using Equation 12 is around 5% (between the two horizontal dotted lines). This indicates that multiple words with close write rates can be combined into one group without introducing significant error to the result.

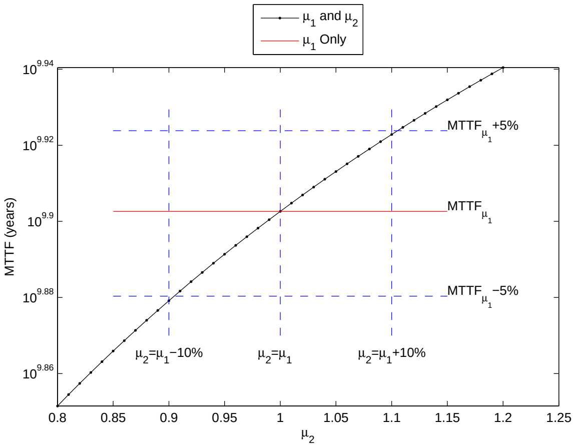

Fig. 2: Combining Two Words with Similar Write Rates into One Group.

With the knowledge from the previous example, we would like to expand the case to a larger number of words. Here is an example: there are in total 101 words in a memory, and the write rate of word 1, μ_1, is 1. The write rates of 50 words distribute randomly in the open interval $(0.9, 1)$, and the other 50 words in $(1, 1.1)$. Thus μ_1 is the median of all the write rates. Two types of MTTF are calculated using Equation 11. The first one is calculated with the median write rate, μ_1, called $MTTF\ median$. The second one is calculated using the write rate of each individual word, called $MTTF\ accurate$. The errors introduced by $MTTF\ median$ over $MTTF\ accurate$ were gathered from 1,000 sets of random write rates and shown in Figure 3.

Figure 3 shows that when the write rates distribute in the range of the median write rate $\pm 10\%$, the errors are always less than 2% - in most cases, 1%. The error level is lower than the first example where there are only 2 words, because the effects of fast write rates and slow write rates can cancel each other out to some extent.

C. Impact of Write Rate on Reliability

With the knowledge from the previous subsection, it becomes possible to investigate the reliability of a memory with multiple groups. In this subsection, the situation

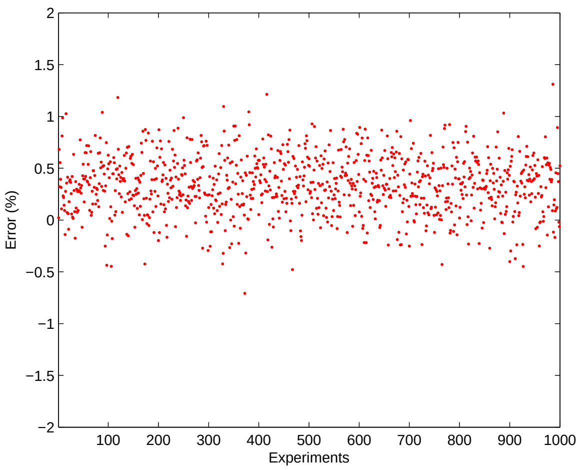

Fig. 3: Error Introduced by Using the Median Write Rate for the Entire Memory.

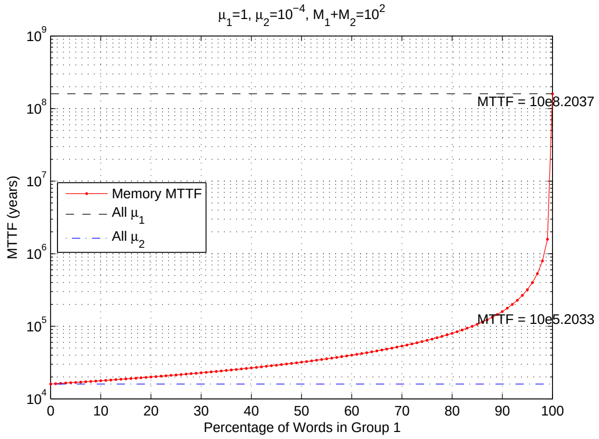

Fig. 4: Reliability of a Memory with Two Groups

where two groups with significantly different write rates is considered. In this case, each group has its own write rate, being μ_1 and μ_2, respectively, and μ_1 is orders of magnitude greater than μ_2. Group 1 has M_1 words, and group 2 has M_2 words. Thus Equation 10 becomes

$$MTTF = \frac{1}{\lambda^2 N(N-1)} \cdot \frac{\mu_1\mu_2}{M_1\mu_2 + M_2\mu_1}. \qquad (14)$$

Figure 4 shows Equation 14 with M_1 increasing and $M_1 + M_2$ being fixed at 100. In addition, μ_1 is four orders of magnitude greater than μ_2. The memory can reach its highest possible MTTF when $M_1 = 100$, and its lowest possible MTTF when $M_1 = 0$. These two bounds are marked by the two dotted lines in Figure 4. It is notable that as M_1 increases, the improvement of memory reliability shows a non-linear characteristic. When 10% of the words belong to group 2, the MTTF of the memory is $10^{5.2}$ years. When all the words are in group 1, the MTTF skyrockets to $10^{8.2}$ years. This huge difference indicates that having a small fraction of words with slow write rates can drastically hurt the reliability of the memory.

Figure 4 shows that the overall reliability of a memory is mainly determined by the memory locations that have low write rates. Therefore it is crucial to improve the reliability of these locations in order to protect the entire memory.

IV. Proposed Model for Mixed Scrubbing

In this section, the proposed model for mixed scrubbing will be derived and analyzed.

A. *Mathematical Expression*

In the previous subsection, the reliability of a single word i with probabilistic scrub rate μ_i was given:

$$r_i(t) = \frac{\alpha_2}{\alpha_2 - \alpha_1} e^{-\alpha_1 t} + \frac{\alpha_1}{\alpha_1 - \alpha_2} e^{-\alpha_2 t}, \qquad (15)$$

where $\alpha_1 = \frac{\lambda(2N-1)+\mu_i-\sqrt{\lambda^2+2\lambda\mu_i(2N-1)+\mu_i^2}}{2}$ and $\alpha_2 = \frac{\lambda(2N-1)+\mu_i+\sqrt{\lambda^2+2\lambda\mu_i(2N-1)+\mu_i^2}}{2}$. Then the reliability of an M-word memory is

$$R(t) = \prod_{i=1}^{M} r_i(t). \qquad (16)$$

When deterministic scrubbing is also taken into account, this reliability is valid before the first scrub. We assume that the deterministic scrub rate is T. After the memory is scrubbed at time T, it is considered error-free. Therefore, the reliability of the memory at time nT (n is a positive integer) is the probability that the memory has survived all n previous scrub intervals, which can be expressed as

$$Q(nT) = R^n(T). \qquad (17)$$

Equation 17 has staircase characteristics which help the evaluation of the MTTF. Figure 5 shows such characteristics. Since MTTF is the integral of the reliability function over time, the area covered by the horizontal dotted lines in Figure 5 represents the lower bound of the MTTF, and the area covered by the horizontal dotted lines plus the area covered by the vertical dotted lines represents the upper bound.

Using the summation formula of geometric progression, we can get the upper bound and lower bound of MTTF:

$$\begin{aligned} MTTF_u &= T[1 + R(T) + R^2(T) + R^3(T) + ...] \qquad (18)\\ &= \frac{T}{1 - R(T)}, \end{aligned}$$

$$\begin{aligned} MTTF_l &= T[R(T) + R^2(T) + R^3(T) + ...] \qquad (19)\\ &= \frac{T}{1 - R(T)} - T. \end{aligned}$$

In Equation 18 and Equation 19, $R(T)$ is the reliability of the memory at time T when only probabilistic scrubbing is applied (can be calculated using Equation 16).

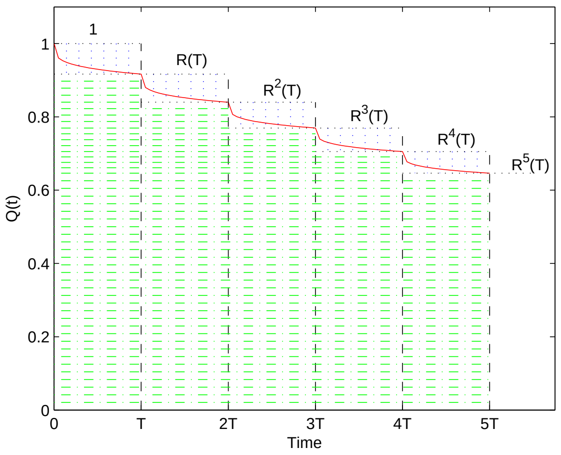

Fig. 5: Variation of the Reliability with Both Types of Scrubbing with Time; Upper and Lower Bound of MTTF

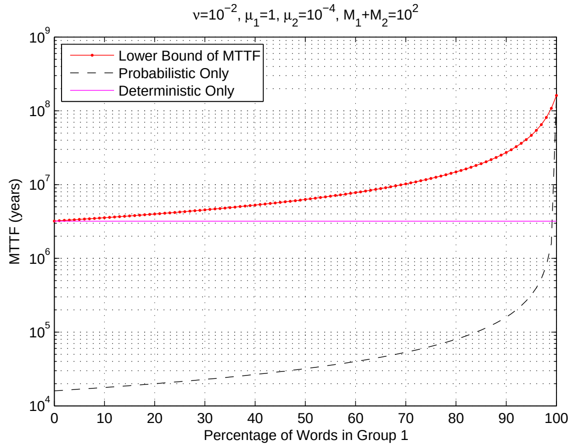

Fig. 6: MTTF Given by the Comprehensive Scrubbing Model for a Memory With Two Different Access Rates.

B. Impact of Deterministic Scrubbing on Reliability

Deterministic scrubbing can be applied when probabilistic scrubbing cannot provide sufficient protection, and the comprehensive scrubbing model proposed by this paper includes both types of scrubbing. In order to better understand the characteristics of the proposed model for comprehensive scrubbing, we consider a case where the memory has in total 100 words, which can be divided into two groups. The first group has a write rate of 1, and the second group has a write rate of 10^{-4}. The deterministic scrub rate is 0.01 for every location in the memory. Figure 6 presents the MTTF of the entire memory when group 1 has different number of words.

The lower dotted curve in Figure 6 shows the MTTF when only probabilistic scrubbing is considered (Equation 10), and the horizontal line shows the MTTF of deterministic scrubbing. When all the words are in group 2, the memory has the lowest MTTF because group 2 has a lower write rate. If deterministic scrubbing is combined with probabilistic scrubbing, then the MTTF can be greatly elevated. From Figure 6 we can see that deterministic scrubbing provides a “base protection” for the memory and it helps the most when most words have low write rates.

From the discussion on the model for comprehensive scrubbing, we can conclude that deterministic scrubbing can improve the MTTF of the entire memory when part of the memory is less frequently written than other parts, as long as the deterministic scrub rate is higher than the lowest write rate of all the locations. In addition, if there are locations with write rate of zero, then adding deterministic scrubbing is an effective way to provide a minimum level of protection.

V. Validation of the Proposed Models

Two sets of Monte Carlo simulations were run to validate the proposed models. MATLAB programs were written to simulate upsets in the memory as well as the scrubbing behavior of the memory. Figure 7 presents a flow chart of the MATLAB program used for the proposed model for non-uniform probabilistic scrubbing. The program used for the proposed model for mixed scrubbing is very similar to Figure 7, with only a few slight changes. The key is to generate random scrubbing events and bit error events with their intervals being exponentially distributed. When two errors show up in a single word, a failure of the memory occurs and the time to failure is recorded. After 100 iterations of such a simulation, the mean time to failure is calculated and then compared to the theoretical value obtained by the proposed models. It is shown that both proposed models give results within 3% of the mean value of the simulated MTTFs.

Figure 8 shows the Monte Carlo simulation results for the proposed model with non-uniform probabilistic scrubbing. In this simulation, $\lambda = 10^{-3}$ per second and μ_i randomly distributes in the range of (100,200) per second. Therefore μ_i is five orders of magnitude higher than λ and Equation 10 is valid. In Figure 8, the mean value of the simulation results is 2.89×10^{-5} years, and the MTTF estimate given by Equation 10 is 2.86×10^{-5} years. The difference between the theoretical value and the simulated value is 1.03%.

Figure 9 presents the Monte Carlo simulation results for the proposed model with mixed scrubbing. In this simulation, $\lambda = 10^{-3}$ per second, μ_i randomly distributes in the range of (100,200) per second, and the deterministic scrub interval is $T = 0.02$ (or, $\nu = 50$). In Figure 9, the mean value of the simulation results is 4.17×10^{-5} years, the MTTF of the memory when only probabilistic scrubbing is considered is 2.82×10^{-5} years, the MTTF of the memory when only deterministic scrubbing is considered is 1.97×10^{-5} years, and the MTTF of mixed scrubbing is 4.25×10^{-5} years. Again, the estimate given by Equation 19 is very close to the mean value of the simulation results, with

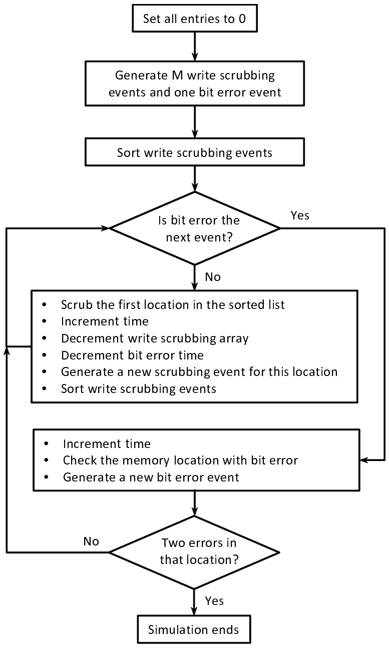
Set all entries to 0
Generate M write scrubbing events and one bit error event
Sort write scrubbing events
Is bit error the next event?
Yes
No
Scrub the first location in the sorted list
Increment time
Decrement write scrubbing array
Decrement bit error time
Generate a new scrubbing event for this location
Sort write scrubbing events
Increment time
Check the memory location with bit error
Generate a new bit error event
No
Two errors in that location?
Yes
Simulation ends

Fig. 7: Flow Chart of the Monte Carlo Simulation.

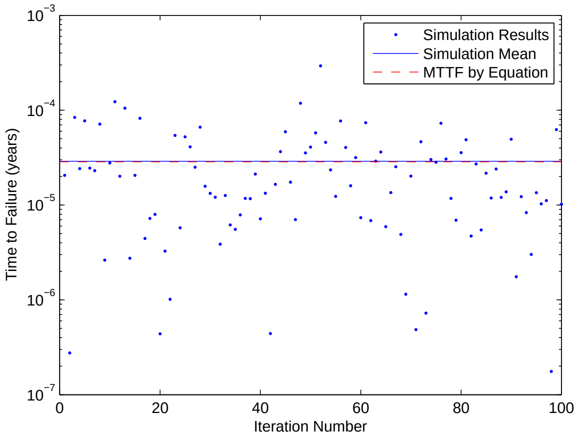

Fig. 8: Comparison of Monte Carlo Simulation Results and the Proposed Model for Non-Uniform Probabilistic Scrubbing.

a difference of 2.08%.

VI. A Real Life Example Illustrating The Value of the Proposed Models

FPGAs are an attractive technology for DSP applications because they are particularly suitable for parallel algorithm implementation. In [7], Lavin et al. described the design of a Space-time Coded Telemetry Receiver (SCTR), which can solve the issue of data dropouts due to the use of multiple transmit antennas. Figure 10 shows a block diagram of this design. The SCTR design spreads over three FPGAs, called brik1, brik2, and brik3, respectively. The information of BRAM memory use in each block is summarized by Table IV.

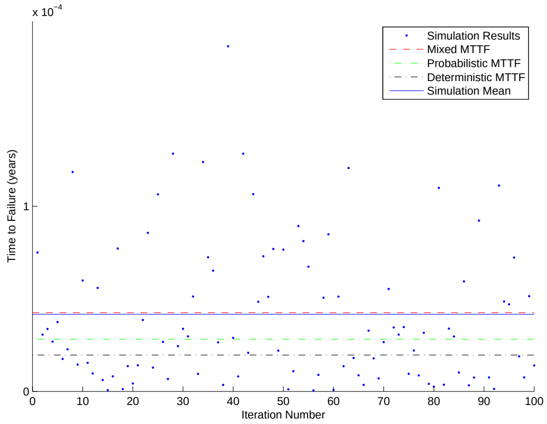

Fig. 9: Comparison of Monte Carlo Simulation Result and the Proposed Model for Mixed Scrubbing.

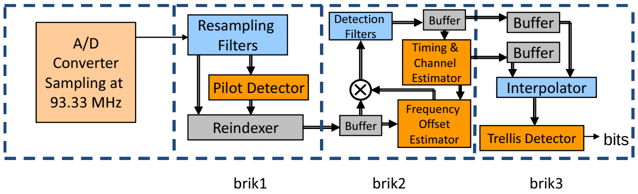

Fig. 10: A Block Diagram of the 3-FPGA MIMO Receiver [7].

Simulations were run on each FPGA block for 1.5×10^6 cycles, and the write access patterns of all memories in each block were collected. The data width of the BRAM memories were 18 bits each, and clock period was approximately $5ns$. The first column of Table IV gives the number of memory words in each design block. The "Least μ_i" column presents the write rate (per second) of the least written location in each block. The third column, "Never", shows the number of words that were never written to. Using Equation 10, the last column gives the MTTF estimates when all memories in each block are considered as an entity.

TABLE IV: BRAM Use in the SCTR Design

Block	Words of Memory	Least μ_i	Never	MTTF (years)
Brik1	8,448	8.68×10^4	0	4.05×10^{12}
Brik2	5,664	0	12	<32.35 *
Brik3	2,752	2.67×10^3	0	2.59×10^{11}

* The MTTF of brik2 is calculated using the equation for SEC/DED memory without scrubbing [3].

In brik1, there are 23 BRAMs and 8,448 words in total. All memory words in brik1 were written, and the least written location has a write rate of 8.68×10^4. Using Equation 10, the MTTF of the memory on brik1 is calculated as 4.05×10^{12} years. Brik3 has 32 BRAMs and 2,752 words. Similar to brik1, all memory words in brik3 were written, and the memory has an MTTF of 2.59×10^{11} years. Brik2 has 18 BRAMs and 5,664 words. However, 12 words from two of the BRAMs were never written to and thus have a probabilistic scrub rate of zero. Because Equation 10 is not accurate when $\mu_i = 0$, the MTTF of the 12 non-written words is calculated using the equation for SEC/DED memory without scrubbing [3] and the result is 32.35 years. Thus the MTTF of the BRAMs on brik2 must be shorter than 32.35 years. This confirms the discussion in Section III that the words with zero write rate drastically reduce the MTTF of the entire memory system. If the 12 words can be excluded from the memory, then the MTTF will be elevated to 8.05×10^{10} years.

This example further shows that in DSP applications, most memory cells tend to be written frequently because of the streaming of large amounts of data through memory. Thus, memories that are regularly written do not need to be protected by extra deterministic scrubbing. However, it also shows how severely the words with low write rate can hurt the overall reliability of the memory. Therefore, memories which have words with very low or zero write rate must be scrubbed. For example, adding a scrubber is the most straightforward way to protect those infrequently written locations. Alternatively, if the memory access pattern is known, the designer may be able to modify the design and have it periodically correct these locations. Another possible solution is replacing these locations with triplicated registers if there are only a few such locations.

VII. Conclusion

In this paper, previous reliability models for SEC/DED memory with scrubbing are discussed and compared, and two new models are derived based on the characteristics of FPGA applications. The proposed model for probabilistic write scrubbing takes into account non-uniform write rates, and the proposed model for mixed scrubbing combines both probabilistic write scrubbing and deterministic scrubbing. The proposed models indicate that memory locations that are frequently overwritten tend to be reliable while locations that have rare write access may need extra protection in order to maintain the overall reliability or the entire memory. Therefore it can help the designers if they know the memory access patterns of their designs. In addition, Matlab-based Monte Carlo simulations show that both models give results within 3% of the simulation mean value for MTTF.

DSP is a common application of FPGAs. Experiments showed that in DSP applications, most memory locations are frequently rewritten and do not require deterministic scrubbing. For those few words that are seldom or never written, solutions such as deterministic scrubbing or triplication are required to protect the overall reliability.

References

[1] "Virtex-4 FPGA User Guide," tech. rep., Xilinx Corporation, December, 1 2008. UG070(v2.6).

[2] "Virtex-5 FPGA User Guide," tech. rep., Xilinx Corporation, May, 17 2010. UG190(v5.3).

[3] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of Scrubbing Recovery-Techniques for Memory Systems," *IEEE Transactions On Reliability*, vol. 39, pp. 114–122, April 1990.

[4] J. Ayache and M. Diaz, "A Reliability Model for Error Correcting Memory Systems," in *Reliability, IEEE Transactions on*, vol. R-28, pp. 310–315, October 1979.

[5] S. A. Elkind and D. P. Siewiorek, "Reliability and Performance of Error-Correcting Memory and Register Arrays," in *Computers, IEEE Transactions on*, vol. C-29, pp. 920–927, October 1980.

[6] G. Allen, L. Edmonds, C. W. Tseng, G. Swift, and C. Carmichael, "Single-Event Upset (SEU) Results of Embedded Error Detect and Correct Enabled Block Random Access Memory (Block RAM) Within the Xilinx XQR5VFX130," in *Nuclear Science, IEEE Transactions on*, pp. 3426–3431, December 2010.

[7] C. Lavin, B. Nelson, J. Palmer, and M. Rice, "An FPGA-based Space-time Coded Telemetry Receiver," in *Aerospace and Electronics Conference*, (Dayton, OH), July 2008.