Optimizing Scrubbing Sequences for Advanced Computer Memories

Pedro Reviriego, *Member, IEEE*, Juan Antonio Maestro, *Member, IEEE*, and Sanghyeon Baeg, *Member, IEEE*

***Abstract*—Advanced memories are designed using smaller geometries and lower voltages. This enables larger levels of integration and reduced power consumption, but makes memories more prone to suffer multibit soft errors. In this scenario, scrubbing is a fundamental technique to avoid the accumulation of errors, which would lead to a failure of the system. Scrubbing is usually implemented in advanced memories. However, when the percentage of multibit soft errors is significant, the scrubbing sequence (the order in which the memory is scrubbed) becomes important for the reliability of the system. In this paper, a new procedure to perform scrubbing is presented, which offers a significant improvement in the reliability. In the presence of multiple cell upsets, the mean time to failure could be doubled with respect to the traditional approach.**

***Index Terms*—Interleaving distance, memory, multiple cell upset (MCU), soft error.**

I. Introduction

MEMORY RELIABILITY is a critical issue in most computer applications [1], [2]. One example of errors that occur in memories are those caused when a radiation particle hits a memory device [3]–[6]. The impact can produce changes in the stored values of some memory cells leading to data corruption [7], [8]. Those errors are not permanent and can be removed by writing back the correct values to those memory cells. Therefore, they are commonly known as soft errors. Most soft errors produce isolated bit flips, usually known as single event upsets (SEUs) [9]–[11].

To mitigate the effects of soft errors, memories typically implement per-word error correction mechanisms [12]–[14]. In this technique, a number of additional redundancy bits are added to each word to detect and correct errors. Single error correction (SEC) codes [15]–[17] are commonly used in memory designs. These codes can correct a single error per word such that data corruption occurs only when there are two or more errors in a word.

SEC-protected memories can handle single errors but are not able to deal with events that cause multiple cell upsets (MCUs) [18]–[20]. Although SEUs are still the most common phenomena, MCUs are increasingly present in advanced memories as lower voltages and smaller geometries are used [18], [21], [22]. This means that a particle is more likely to affect more than one cell when it hits the memory device. Typically, the affected cells in an MCU are physically close together and also close to the area impacted by the particle [23]. To deal with MCUs, SEC codes are combined with interleaving [24]. Interleaving ensures that bits that belong to the same logical word are stored in cells that are physically distant. In this way, an MCU cannot affect two cells of the same logical word. When interleaving is used, MCUs appear as multiple single errors in a number of logical words and can be effectively handled by the SEC code.

A memory protected with SEC and interleaving can effectively remove single-word errors caused by SEUs and MCUs. However, these effects tend to accumulate with time on more and more words, making data corruption produced by a second error event on the same word more likely. The average number of events needed to cause a failure has been studied in the literature both for SEUs [25]–[27] and for MCUs [28]. The results show that when the arrival rate of error events is large, reliability may not be sufficient for many applications.

To increase reliability, a technique called scrubbing is commonly used in memories protected with SEC [27], [29]. Scrubbing consists in reading the memory sequentially, and if a single error is detected on any word, it is written back again in order to remove the error. The process is done continuously, and the entire memory is read each T_s seconds, which is called the scrubbing period (time between two consecutive scrubbing processes). Scrubbing prevents errors from accumulating in the memory, as they will be removed in at most a scrubbing interval, i.e., in T_s seconds. The cost associated with scrubbing is twofold: First, a fraction of the memory bandwidth is dedicated to scrubbing rather than to system functions; and second, scrubbing increases power consumption as more memory operations are required. From a reliability point of view, the smallest the scrubbing period, the better, since errors are removed faster. However, the use of small scrubbing periods will increase the cost associated with this technique.

The idea introduced in this paper is to modify the scrubbing process in order to detect and correct the errors caused by MCUs more effectively. This is done by exploiting the locality of the errors in an MCU. The proposed approach provides an increased level of reliability when compared with the traditional scrubbing using the same T_s. Alternatively, the proposed scheme can be used to reduce T_s while maintaining the reliability level, thus reducing power consumption and increasing the memory bandwidth available for system use.

Manuscript received September 7, 2009; revised October 30, 2009; accepted December 15, 2009. Date of publication December 28, 2010; date of current version June 4, 2010. This work was supported in part by the Spanish Ministry of Science and Innovation under Grant AYA2009-13300-C03-01, by the Regional Government of Madrid, and by the European Union FEDER Programme.

P. Reviriego and J. A. Maestro are with the Universidad Antonio de Nebrija, 28040 Madrid, Spain (e-mail: previrie@nebrija.es; jmaestro@nebrija.es).

S. Baeg is with the School of Electrical Engineering and Computer Science, Hanyang University, Ansan 425-791, Korea (e-mail: bau@hangyang.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TDMR.2009.2039481

1530-4388/$26.00 © 2010 IEEE

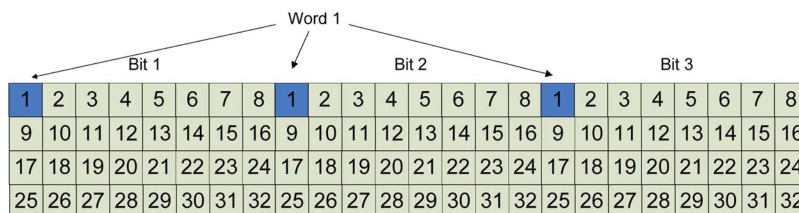

Fig. 1. Memory organization.

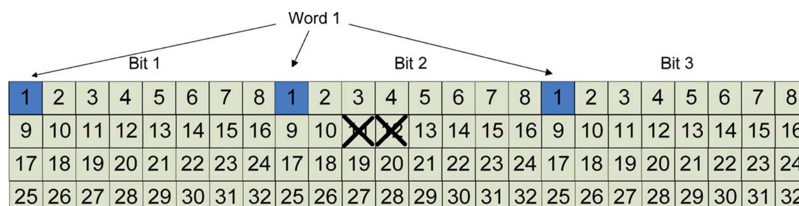

Fig. 2. MCU example.

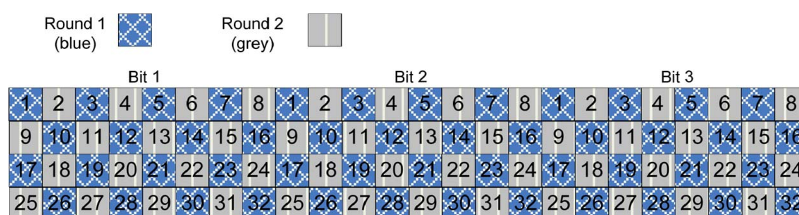

Fig. 3. Alternative scrubbing sequence.

The rest of this paper is organized as follows. The problem assumptions and reliability estimates are introduced in Section II. The proposed scrubbing scheme is presented in Section III. Then, in Section IV, the performance of the new scheme is analyzed in terms of reliability and memory operations overheads, and finally, a case study is presented in Section V.

II. Problem Assumptions and Reliability Estimates

The following summarizes the assumptions used in the analysis and simulations presented in the rest of this paper.

1) The memory is protected with per-word SEC.
2) The memory implements an interleaving scheme such that MCUs cause only single errors on multiple words.
3) Scrubbing is implemented using part of the available memory bandwidth. The read/write operations required are uniformly distributed over the scrubbing period T_s.
4) Events are supposed to arrive following a Poisson distribution.

In order to characterize the reliability, the following parameters will be used.

1) Mean time to failure (MTTF): The factor that measures reliability in this approach. It represents how much time the memory should work on average until it fails.
2) Event arrival rate (λ): Number of events that are produced in a memory word per unit of time.
3) The probability that an event causes i errors: $p(i)$.
4) Mean number of events to failure (METF): This represents how many events the memory has suffered on average until reaching failure (during the time determined by the MTTF). Following a well-known relation for Poisson processes (see [25])

$$\text{MTTF} = \text{METF}/\lambda. \tag{1}$$

III. Proposed Scrubbing Scheme

The memory organization shown in Fig. 1 is used to show the proposed scheme. This structure has been previously used in the memory analyzed in [21], and implements interleaving with a distance of eight between cells of the same word. The number in each cell is the address of the logical word associated to it. Only 3 bits of each word are shown. The rest of the bits in each word would be grouped in similar blocks. Additional words are also placed below the portion shown for the first 32 words.

Traditional scrubbing would read the words in sequential order, in an iterative way. Doing this, an error could take T_s seconds to be corrected, in the worst case. When an MCU occurs, a number of physically close cells would suffer errors, as shown in Fig. 2. In this particular example, bit 2 of words 11 and 12 have suffered an error, which will be cleared when scrubbing reaches those words. This will take on average $T_s/2$ seconds.

However, it is possible to propose more elaborated scrubbing sequences in order to reduce the time needed to detect and correct the errors caused by MCUs.

As an example, an alternative scrubbing scheme is shown in Fig. 3. In this case, the process is divided into two consecutive rounds. In each round, alternate words are processed. Therefore, each round will only deal with half the number of

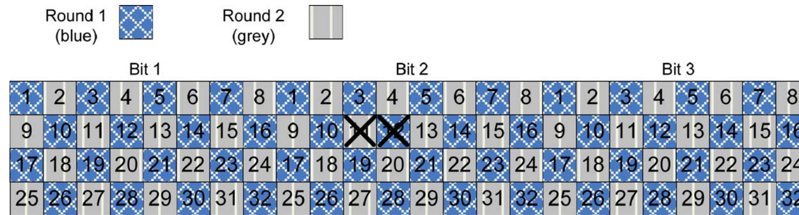

Fig. 4. MCU example.

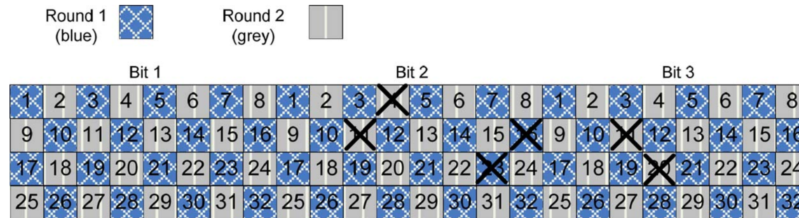

Fig. 5. MCUs for which there are no benefits using the proposed scheme.

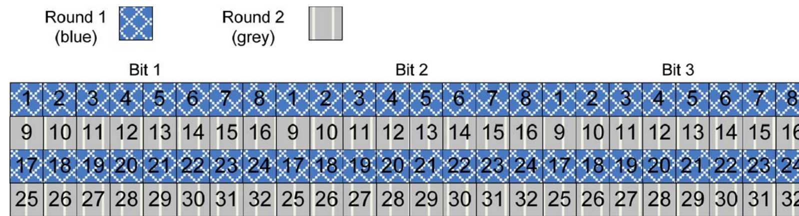

Fig. 6. Scrubbing scheme for a memory on which diagonal MCUs are dominant.

total words. With this change, the time needed to complete one scrubbing process remains unchanged, which is T_s seconds.

Obviously, this reordering of the scrubbing operations does not increase reliability by itself. Now, a further change to the scrubbing process is introduced in order to handle MCUs more efficiently. When an error is detected, the adjacent bits are immediately checked in order to verify if an MCU has happened. If some of the adjacent bits are identified as having suffered an upset, the scrubbing process is automatically stopped, and the wrong words are corrected. After this, the scrubbing continues its normal operation.

With this change, reliability is effectively increased. To illustrate this, an example will be used. Consider the 2-bit MCU shown in Fig. 4. If scrubbing is performing round 1, then an error will be detected in bit 2 of word 12. Scrubbing will then stop and check words 4, 11, 20, and 13. This will detect the error in word 11, which would launch the correction process. After this, scrubbing would continue with round 1 on word 14. If the MCU is produced while in round 2, a similar procedure would eliminate it. This means that MCUs are effectively removed, in the worst case, each $T_s/2$ seconds instead of each T_s seconds, as in traditional scrubbing. This intuitively should increase memory reliability.

However, the benefits of the proposed scrubbing would only apply to MCUs that have at least one error in each round set. In the example shown before, this would be the case for all horizontal and vertical double MCUs whose errors are adjacent. Examples of patterns for which the proposed technique would provide no benefit are shown in Fig. 5. They correspond to diagonal MCUs. Therefore, the proposed organization of the rounds used so far is targeted for memories in which horizontal or vertical errors are dominant. Determining which types of MCUs are dominant may be complicated, as the MCU shape can be influenced by memory design and layout and also by the angle of incidence of the radiation particle when it hits the device. This issue will be discussed further in the following sections.

For memories on which other MCU patterns are dominant, the same idea can be used changing the cells that are checked in each round. As an example, if a memory in which diagonal errors are dominant is considered, the scheme shown in Fig. 6 could be used to ensure the correction of diagonal errors in just one scrubbing round.

The main overheads associated with the proposed technique are the additional read and write memory operations required each time an error is detected as well as the increased complexity of the scrubbing process. The former is quantified in the next section, showing that it is negligible, while the latter should not be an issue since the additional control logic is simple to implement.

IV. Performance Analysis

In this section, the performance of the proposed scrubbing scheme is analyzed in terms of the reliability improvement that it provides versus traditional scrubbing. In addition, the number of additional memory operations that are required for the accelerated error correction process is studied. Reliability will be assessed in terms of the MTTF, which provides a simple reliability estimate for comparison. Simulation results to validate the analysis are also presented.

A. Reliability Analysis

1) SEU Scenario—Traditional Analysis: The MTTF of memories that use SEC and scrubbing has been widely studied in the literature [25]–[28] showing that when the scrubbing period T_s is small, it can be approximated by

$$\text{MTTF} \cong \frac{T_s}{P_f} \tag{2}$$

where P_f is the probability of failure in a scrubbing period. P_f can be estimated in different ways (see [25]–[28] for more details). In this paper, the following approximation will be used. Assuming a per-word event arrival rate λ and a memory of M words, the number of errors that will have accumulated, on average, will be

$$e_{\text{accum}} = \frac{\lambda \cdot M \cdot T_s}{2}. \tag{3}$$

This is derived by noting that only errors that have occurred in the previous T_s seconds can still be in the memory, and that of those, on average, a half would have already been scrubbed. In this situation, a failure would happen if a new event affects a memory word already affected by a previous error. Then, a double error would be produced, which would not be corrected by the SEC codes. The probability of this scenario is proportional to the number of words that already have one error:

$$P_f^{\text{event}} \cong \frac{e_{\text{accum}}}{M} = \frac{\frac{\lambda \cdot M \cdot T_s}{2}}{M} = \frac{\lambda \cdot T_s}{2}. \tag{4}$$

And finally, as on average $n_e = \lambda \cdot M \cdot T_s$ events arrive on a scrubbing period, the failure probability in that interval can be approximated by

$$P_f \cong n_e \cdot P_f^{\text{event}} = \lambda \cdot M \cdot T_s \cdot \frac{\lambda \cdot T_s}{2} = \frac{(\lambda \cdot T_s)^2 \cdot M}{2}. \tag{5}$$

Using (2), the following expression for the MTTF is obtained:

$$\text{MTTF}_{\text{traditional}} \cong \frac{T_s}{P_f} \cong \frac{2}{\lambda^2 \cdot T_s \cdot M}. \tag{6}$$

2) Extension to MCUs: In the previous derivation, single soft errors have been assumed. When MCUs are considered, the situation changes slightly, since in this case the number of errors is always greater than the number of events. In [28], it has been proved that the MTTF produced by MCUs can be approximated by considering the errors as independent. In this way, each event would produce on average the following number of errors:

$$E_{\text{event}} = \sum_{i=1}^{\infty} i \cdot p(i) \tag{7}$$

where $p(i)$ is the probability that an event causes i cell errors. The MTTF can be then approximated by [modifying the event arrival rate in (6) with (7)]

$$\text{MTTF}_{\text{traditional}} \cong \frac{T_s}{P_f} \cong \frac{2}{(\lambda \cdot E_{\text{event}})^2 \cdot T_s \cdot M}. \tag{8}$$

This is the MTTF assuming traditional scrubbing.

3) Proposed Scrubbing Algorithm: Now, the proposed scheme is considered assuming that it is able to detect all MCUs in one scrubbing round. In this case, the average number of errors in the memory can be approximated by

$$e_{\text{accum}} = p(1) \cdot \frac{\lambda \cdot M \cdot T_s}{2} + \sum_{i=2}^{\infty} i \cdot p(i) \cdot \frac{\lambda \cdot M \cdot T_s}{4} \tag{9}$$

where the first term are the errors due to SEUs, and the second term the errors caused by MCUs $(i > 1)$, which are effectively removed each $T_s/2$ seconds (and therefore stay on average $T_s/4$ seconds in the memory). Following a similar derivation, the probability of failure in a scrubbing period can be approximated by [see (5) modified by (7)]

$$P_f \cong \frac{\lambda \cdot T_s}{2} \cdot \left(p(1) + \frac{\sum_{i=2}^{\infty} i \cdot p(i)}{2} \right) \cdot \sum_{i=1}^{\infty} i \cdot p(i) \cdot \lambda \cdot M \cdot T_s$$

$$= \frac{\lambda^2 \cdot M \cdot (T_s)^2}{2} \cdot E_{\text{event}} \cdot \left(p(1) + \frac{\sum_{i=2}^{\infty} i \cdot p(i)}{2} \right). \tag{10}$$

Finally, the MTTF can be then approximated by [see (8)]

$$\text{MTTF}_{\text{proposed}} \cong \frac{2}{\lambda^2 \cdot E_{\text{event}} \cdot \left(p(1) + \frac{\sum_{i=2}^{\infty} i \cdot p(i)}{2} \right) \cdot T_s \cdot M}. \tag{11}$$

In order to compare both techniques, the increase in MTTF provided by the proposed scheme can be measured by computing the following ratio:

$$\frac{\text{MTTF}_{\text{proposed}}}{\text{MTTF}_{\text{traditional}}} \cong \frac{E_{\text{event}}}{\left(p(1) + \frac{\sum_{i=2}^{\infty} i \cdot p(i)}{2} \right)} = \frac{2 \cdot E_{\text{event}}}{p(1) + E_{\text{event}}}. \tag{12}$$

This ratio will be larger when the average number of cell errors is large and the percentage of SEUs is small. In other words, when the MCUs are dominant. In the limit case, where there are no single errors, the MTTF increase will be a factor of two. Therefore, as MCUs tend to increase as memory technology progresses, more benefits will be obtained from the proposed technique. In the following sections, the MTTF increase is analyzed for different commercial memories to

show that the impact can be significant in current memory technologies.

B. Additional Memory Operations Analysis

In the previous section, it was implicitly assumed that the scrubbing period of the proposed technique was the same as in traditional scrubbing. However, the proposed scheme requires extra memory accesses when an error is detected to immediately check the neighbor cells. In this section, the number of those operations is analyzed to show that is it negligible.

The number of additional memory accesses is related to the number of errors that occur in a scrubbing period. In order to make scrubbing useful, the scrubbing period should be such that the number of accumulated errors in the memory is much smaller than the METF of the memory when no scrubbing is used. Otherwise, the memory would likely fail in one scrubbing period, and therefore this mechanism will not produce any benefit. The METF of a memory protected with SEC when no scrubbing is used can be approximated (see, for example, [25]) by

$$\mathrm{METF}|_{\mathrm{no_scrubbing}}^{\mathrm{SEC}} \cong \sqrt{\frac{\pi \cdot M}{2}}. \tag{13}$$

Assuming that for each detected error B neighboring cells are read, the number of additional memory accesses N_{add} per scrubbing period can be bounded by

$$N_{\mathrm{add}} \ll B \cdot \mathrm{METF}|_{\mathrm{no_scrubbing}}^{\mathrm{SEC}} \cong B \cdot \sqrt{\frac{\pi \cdot M}{2}}. \tag{14}$$

As in a scrubbing period M operations are done for traditional scrubbing, the increase in the number of operations can be bounded by

$$\frac{M + N_{\mathrm{add}}}{M} < 1 + \frac{B \cdot \sqrt{\frac{\pi \cdot M}{2}}}{M} = 1 + B \cdot \sqrt{\frac{\pi}{2 \cdot M}} \tag{15}$$

which for large M would be close to one $(N_{\mathrm{add}} \ll M)$. For example, for a 64-K memory and $B = 4$, the value is bounded by 1.019. For larger memories, the bound would be even closer to one, showing that the overhead of the proposed technique should be negligible in most cases.

Therefore, it is reasonable to assume that the additional memory operations do not affect the scrubbing period and therefore the analysis presented in the previous section is valid.

C. Simulation Experiments

To validate the theoretical analysis presented in the previous sections, a number of simulation experiments have been done. The proposed scrubbing technique has been simulated and the time to failure of 100 000 simulations have been averaged to estimate the MTTF for different configurations. The same experiments have also been performed for the traditional scrubbing. In all the experiments, only 2-bit MCUs are considered, which are related to probability $p(2)$. These MCUs are assumed to be horizontal or vertical, which are the most common types

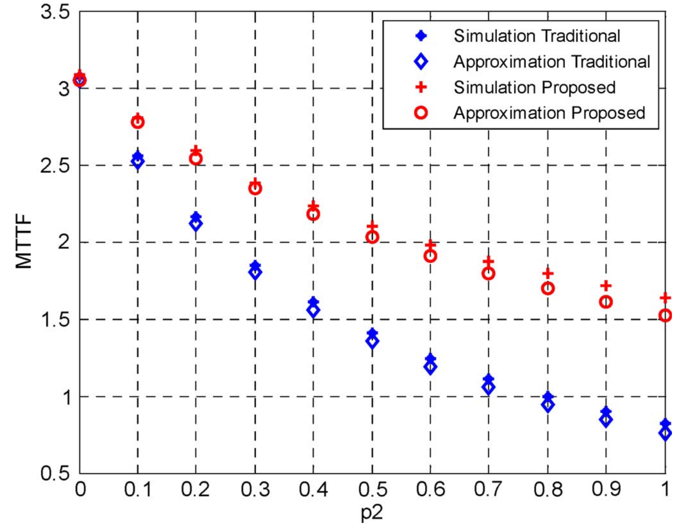

Fig. 7. Simulation results for $M = 64$ Kwords, $T_s = 0.1$, and $\lambda = 0.01$ for different values of $p(2)$.

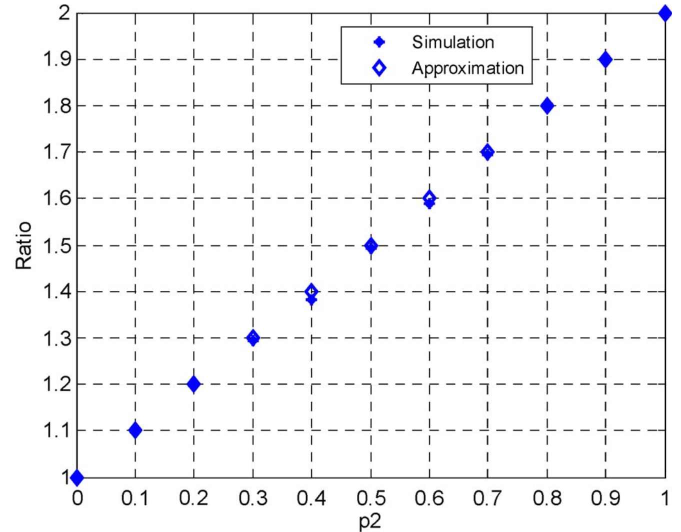

Fig. 8. Ratio of proposed versus traditional for results for $M = 64$ Kwords, $T_s = 0.1$, and $\lambda = 0.01$ for different values of $p(2)$.

of MCUs in many scenarios. Most larger MCUs will also be detected in one round and scrubbed effectively each $T_s/2$, and therefore, the results presented are also applicable to this case.

In Fig. 7, the obtained MTTFs for different values of $p(2)$ are shown. They show a good agreement with the theoretical approximations and the MTTF decreases as $p(2)$ increases, as expected. For large values of $p(2)$, the MTTF provided by the proposed scrubbing technique increases substantially compared to the tradition technique. To further illustrate this increase, the ratios of the proposed versus traditional MTTFs are shown in Fig. 8 for the simulation and the theoretical approximation. The ratio increases up to two when $p(2) = 1$, as predicted by (12).

In Fig. 9, the results for different values of the memory size M are shown, which are in line with the theoretical approximations. The MTTF increase of the proposed technique is shown in Fig. 10, where the ratio of the proposed versus the traditional technique is shown for simulation and approximation.

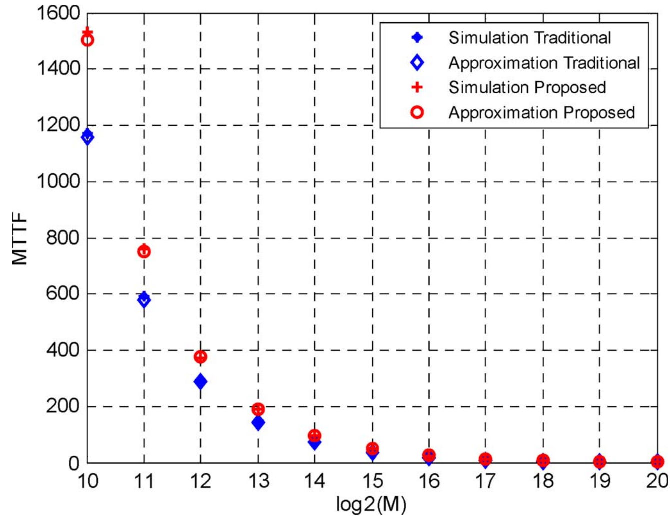

Fig. 9. Simulation results for $p(2) = 0.3$, $T_s = 0.01$, and $\lambda = 0.01$ for different values of M.

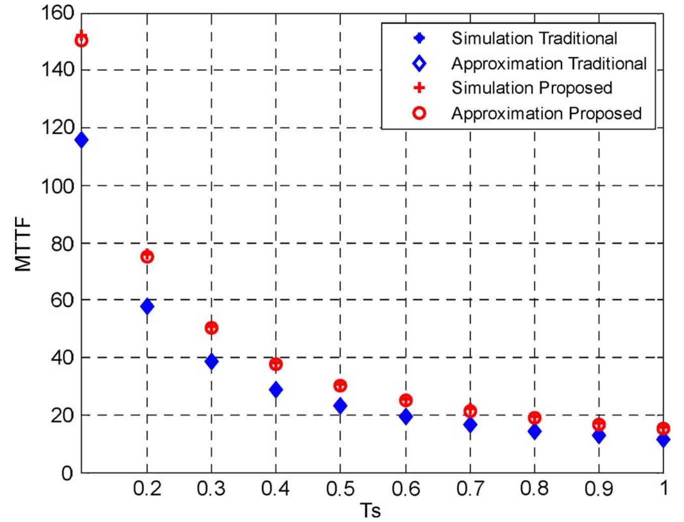

Fig. 11. Simulation results for $p(2) = 0.3$, $M = 1$ Kwords, and $\lambda = 0.01$ for different values of T_s.

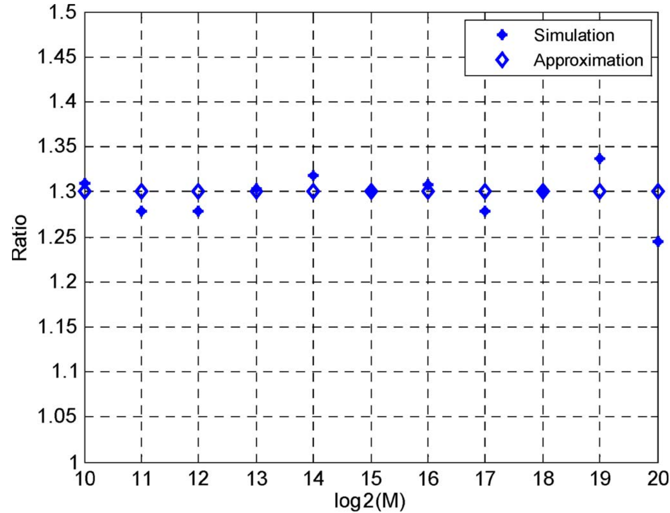

Fig. 10. Ratio of proposed versus traditional for $p(2) = 0.3$, $T_s = 0.01$, and $\lambda = 0.01$ for different values of M.

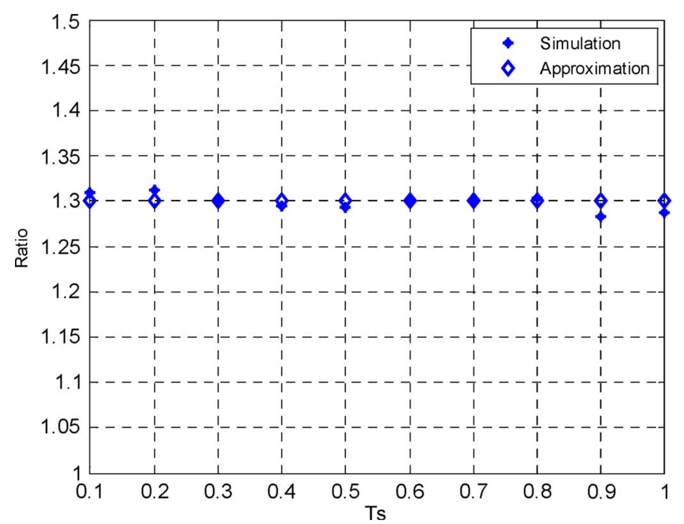

Fig. 12. Ratio of proposed versus traditional $p(2) = 0.3$, $M = 1$ Kwords, and $\lambda = 0.01$ for different values of T_s.

The result is in line with the value of E_{event} (which is 1.3 in this case, since $p(2) = 0.3$), given by the theoretical approximation (12).

In Fig. 11, the results for different values of T_s are shown, and show the expected behavior. The MTTF increase of the proposed technique is shown in Fig. 12 where the ratio of the proposed technique versus the traditional one is plotted for the simulation and the approximation. Again, the result is in line with the value of E_{event}.

The results presented show that the theoretical approximations are accurate when the number of errors per scrubbing period is low. They also confirm the increase in MTTF provided by the proposed scrubbing scheme, approximated by (12).

V. Application to Advanced Memories

In this section, the impact of the proposed scrubbing technique on the MTTF of advanced memories is analyzed.

Three different memory technologies have been studied, which have been previously characterized with real radiation experiments. They correspond to advanced geometries (65 and 45 nm) for which MCUs are a major concern. The memories were exposed to white beams up to 800 MeV at the LANCE site and neutron beams up to 180 MeV at the TSL site. Multiple devices were used and for each one multiple tests were performed. For all tests, the mean time between upsets was much larger than the mean time of an SRAM read cycle for the entire memory. Such configuration was achieved by adjustment of the flux intensity. Once an error was detected, a checking procedure was launched to check the error types. More details on the experiments are given in [24].

The relevant parameter in this experiment is the MCU distribution $p(n)$ assuming that the proposed scrubbing technique is able to correct all MCUs in one scrubbing round (i.e., the MCU patterns are appropriate for the scrubbing round sets). Those $p(n)$ values are shown in Fig. 13, showing a large percentage

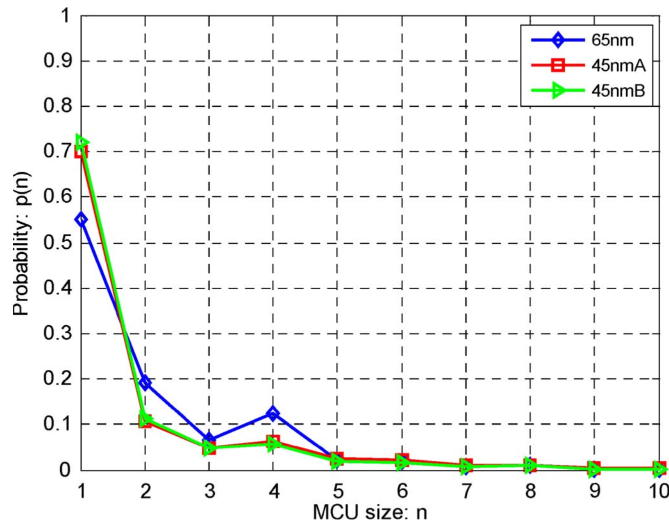

Fig. 13. MCU size distribution $p(n)$ for different memories.

TABLE I
AVERAGE NUMBER OF ERRORS PER EVENT

	65nmA	45nmA	45nmB
E_{event}	2.0649	1.9062	1.7573

TABLE II
MTTF INCREASE FOR THE PROPOSED SCRUBBING

	65nmA	45nmA	45nmB
Ratio	1.5783	1.4625	1.4187

of MCUs. The average number of errors per event (E_{event}) is given in Table I. The data presented in this table are counterintuitive, as there is a larger proportion of MCUs in 65 nm than in 45 nm. The reason of this is that the 45-nm memory design was enhanced based on the acquired experience on the 65-nm memory process, making the former more reliable (see [24]). Using the MTTF ratio approximation (12), the MTTF increase obtained by using the proposed technique can be estimated. The results are shown in Table II. For all the memories, the MTTF increase is over 40%. This clearly illustrates the benefits of the proposed scrubbing that will be even larger for more advanced memories that are more likely to suffer MCUs.

In the previous discussion, it was assumed that all MCUs were detected and corrected in one scrubbing round of the proposed technique. This is an approximation, since diagonal MCUs will not be detected in a single round. To better analyze the impact of this assumption, the detailed data presented in [21] will be used. The reported error patterns and their relative occurrence are shown in Tables III–V.

The first analysis is to quantify the percentage of double MCUs that may not be detected in a single round: from the mentioned tables, those range from 9.1% to 15.8%, depending on the energy. This shows that most double MCUs would be effectively scrubbed in one round. If the same analysis is done on the triple and quadruple MCUs, the percentage of triple MCUs that may not be scrubbed in a round ranges from 2.8% to 7.9%, while for quadruple errors the range is 6.6% to 14.7%. In all those estimations, the patterns that are not detailed in [21]

TABLE III
TWO-BIT MCU ERROR PATTERNS REPORTED IN [21]
(NORMALIZED TO 1000)

Energy	**Double-bit upset type**			
				Others
22MeV	920	34	15	28
47MeV	861	62	27	27
95MeV	792	79	40	40
144MeV	799	62	44	56

TABLE IV
THREE-BIT MCU ERROR PATTERNS REPORTED IN [21]
(NORMALIZED TO 1000)

Energy	**Triple-bit upset type**					
						Others
22MeV	920	34	15	0	3	28
47MeV	861	62	27	13	10	27
95MeV	792	79	40	26	23	40
144MeV	799	62	44	23	16	56

TABLE V
FOUR-BIT MCU ERROR PATTERNS REPORTED IN [21]
(NORMALIZED TO 1000)

Energy	**Quadruple-bit upset type**					
						Others
22MeV	726	57	66	47	38	66
47MeV	621	79	103	84	42	70
95MeV	482	154	109	99	41	116
144MeV	455	114	136	98	49	147

TABLE VI
MTTF INCREASE FOR THE PROPOSED SCRUBBING WITH 15% MCUS NOT DETECTED IN ONE ROUND

	65nmA	45nmA	45nmB
Ratio	1.4523	1.3676	1.3349

(labeled as "Others") are assumed as not detected in a scrubbing round. This is a conservative assumption, and therefore, in reality, the number of undetected MCUs in one round will be significantly lower than that of the values reported. Assuming, as the worst case, a 15% of MCUs that are not detected in one round, the data of Table II can be recalculated. To achieve this, the average time that errors spend in the memory before they are scrubbed has been increased accordingly. The results are presented in Table VI and show that a significant increase in the MTTF is still obtained with these conservative assumptions by using the proposed scrubbing scheme.

The analysis of these data suggests that in many cases, most of the MCUs will be effectively scrubbed in one round and that therefore substantial increases in the MTTF could be achieved.

Finally, it is important to notice that the MCU patterns usually depend on many factors, like the angle of incidence of the radiation particle, and therefore, it may be difficult to predict

which error patterns would be dominant in a given application. In any case, the proposed technique would always improve the MTTF.

VI. Conclusion

In this paper, a new scrubbing scheme has been proposed which improves the memory reliability when MCUs are present. This is achieved by exploiting the physical correlation of the errors produced by an MCU. For memories in which MCUs are dominant, the MTTF achieved when implementing the new approach can be up to two times the one provided by traditional scrubbing. This is an interesting result, as MCUs are becoming increasingly common as technology scales.

The cost associated with the new scheme is a negligible number of additional memory accesses for scrubbing and more sophisticated control logic to implement the scrubbing process. Both should have a minor impact on the system.

The new scheme can also be used to reduce the number of memory accesses required to obtain a given MTTF. This would leave more memory bandwidth for system use and reduce the power consumption.

Finally, the same principle can be applied when larger MCUs are dominant by dividing the scrubbing process in more rounds to detect these larger MCUs earlier. This would provide even larger increases in the MTTF.

References

[1] L. Schiano, M. Ottavi, and F. Lombardi, "Markov models of fault-tolerant memory systems under SEU," in *Proc. Rec. Int. Workshop Memory Technol., Des. Test.*, Aug. 2004, pp. 38–43.

[2] G. C. Cardarilli, A. Leandri, P. Marinucci, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Design of a fault tolerant solid state mass memory," *IEEE Trans. Rel.*, vol. 52, no. 4, pp. 476–491, Dec. 2003.

[3] C. A. Gossett, B. W. Hughlock, M. Katoozi, G. S. LaRue, and S. A. Wender, "Single event phenomena in atmospheric neutron environments," *IEEE Trans. Nucl. Sci.*, vol. 40, no. 6, pp. 1845–1852, Dec. 1993.

[4] Y. Tosaka, H. Kanata, T. Itakura, and S. Satoh, "Simulation technologies for cosmic ray neutron-induced soft errors: Models and simulation systems," *IEEE Trans. Nucl. Sci.*, vol. 46, no. 3, pp. 774–779, Jun. 1999.

[5] J. E. Vinson, "Circuit reliability of memory cells with SEU protection [for space application]," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 6, pp. 1671–1678, Dec. 1992.

[6] A. S. Brogna, F. Bigongiari, F. Bertuccelli, W. Errico, S. Giovannetti, E. Pescari, and R. Saletti, "SEU protected CPU for slow control on space vehicles," in *Proc. 2nd IEEE Int. Workshop Electron. Des., Test Appl.*, Jan. 2004, pp. 422–424.

[7] J. F. Ziegler and W. A. Lanford, "The effect of sea level cosmic rays on electronic devices," *J. Appl. Phys.*, vol. 52, no. 6, pp. 4305–4312, Jan. 1981.

[8] R. D. Schrimpf and D. M. Fleetwood, *Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices*. Singapore: World Scientific, 2004.

[9] T. C. May and M. H. Wood, "A new physical mechanism for soft errors in dynamic memories," in *Proc. 16th Annu. Int. Rel. Phys. Symp.*, 1978, pp. 33–40.

[10] E. Normand, "Single event upset at ground level," *IEEE Trans. Nucl. Sci.*, vol. 43, no. 6, pp. 2742–2750, Dec. 1996.

[11] D. G. Mavi and P. H. Eaton, "Soft error rate mitigation techniques for modern microcircuits," in *Proc. 40th Annu. Rel. Phys. Symp.*, 2002, pp. 216–225.

[12] T. P. Haraszti, *CMOS Memory Circuits*. Norwell, MA: Kluwer, 2000.

[13] R. C. Baumann, "Soft errors in advanced computer systems," *IEEE Des. Test. Comput.*, vol. 22, no. 3, pp. 258–266, May/Jun. 2005.

[14] M. Nicolaidis, "Design for soft error mitigation," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 405–418, Sep. 2005.

[15] R. W. Hamming, "Error detecting and correcting codes," *Bell Syst. Tech. J.*, vol. 29, pp. 147–160, Apr. 1950.

[16] M. A. Bajura, Y. Boulghassoul, R. Naseer, S. DasGupta, A. F. Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N. Damoulakis, "Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 54, pt. 2, no. 4, pp. 935–945, Aug. 2007.

[17] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM J. Res. Develop.*, vol. 28, no. 2, pp. 124–134, Mar. 1984.

[18] A. D. Tipton, J. A. Pellish, R. A. Reed, R. D. Schrimpf, R. A. Weller, M. H. Mendenhall, B. Sierawski, A. K. Sutton, R. M. Diestelhorst, G. Espinel, J. D. Cressler, P. W. Marshall, and G. Vizkelethy, "Multiple-bit upset in 130 nm CMOS technology," *IEEE Trans. Nucl. Sci.*, vol. 53, pt. 1, no. 6, pp. 3259–3264, Dec. 2006.

[19] A. M. Chugg, M. J. Moutrie, and R. Jones, "Broadening of the variance of the number of upsets in a read-cycle by MBUs," *IEEE Trans. Nucl. Sci.*, vol. 51, pt. 2, no. 6, pp. 3701–3707, Dec. 2004.

[20] A. M. Chugg, M. J. Moutrie, A. J. Burnell, and R. Jones, "A statistical technique to measure the proportion of MBUs in SEE testing," *IEEE Trans. Nucl. Sci.*, vol. 53, pt. 1, no. 6, pp. 3139–3144, Dec. 2006.

[21] D. Radaelli, H. Puchner, S. Wong, and S. Daniel, "Investigation of multibit upsets in a 150 nm technology SRAM device," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2433–2437, Dec. 2005.

[22] R. K. Lawrence and A. T. Kelly, "Single event effect induced multiple-cell upsets in a commercial 90 nm CMOS digital technology," *IEEE Trans. Nucl. Sci.*, vol. 55, pt. 1, no. 6, pp. 3367–3374, Dec. 2008.

[23] S. Satoh, Y. Tosaka, and S. A. Wender, "Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAMs," *IEEE Electron Device Lett.*, vol. 21, no. 6, pp. 310–312, Jun. 2000.

[24] S. Baeg, S. Wen, and R. Wong, "SRAM interleaving distance selection with a soft error failure model," *IEEE Trans. Nucl. Sci.*, vol. 56, pt. 2, no. 4, pp. 2111–2118, Aug. 2009.

[25] R. M. Goodman and M. Sayano, "The reliability of semiconductor RAM memories with on-chip error-correction coding," *IEEE Trans. Inf. Theory*, vol. 37, no. 3, pp. 884–896, May 1991.

[26] M. Blaum, R. Goodman, and R. McEliece, "The reliability of single-error protected computer memories," *IEEE Trans. Comput.*, vol. 37, no. 1, pp. 114–119, Jan. 1988.

[27] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of scrubbing recovery-techniques for memory systems," *IEEE Trans. Rel.*, vol. 39, no. 1, pp. 114–122, Apr. 1990.

[28] P. Reviriego, J. A. Maestro, and C. Cervantes, "Reliability analysis of memories suffering multiple bit upsets," *IEEE Trans. Device Mater. Rel.*, vol. 7, no. 4, pp. 592–601, Dec. 2007.

[29] G. C. Yang, "Reliability of semiconductor RAMs with soft-error scrubbing techniques," *Proc. Inst. Elect. Eng.—Comput. Digital Tech.*, vol. 142, no. 5, pp. 337–344, Sep. 1995.

Pedro Reviriego (A'03–M'04) received the M.Sc. and Ph.D. degrees (with honors) in telecommunications engineering from the Technical University of Madrid, Madrid, Spain, in 1994 and 1997, respectively.

From 1997 to 2000, he was an R&D Engineer with Teldat, Madrid, working on router implementation. In 2000, he joined Massana to work on the development of 1000BaseT transceivers. During 2003, he was a Visiting Professor with the University Carlos III, Leganés, Madrid. From 2004 to 2007, he was a Distinguished Member of Technical Staff with LSI Corporation, working on the development of Ethernet transceivers. He is currently with the Universidad Antonio de Nebrija, Madrid. His research interests are fault tolerant systems, performance evaluation of communication networks, and the design of physical layer communication devices. He has authored numerous papers in international conferences and journals. He has also participated in the IEEE 802.3 standardization for 10GBaseT.

Juan Antonio Maestro (M'07) received the M.Sc. degree in physics and the Ph.D. degree in computer science from the Universidad Complutense de Madrid, Madrid, Spain, in 1994 and 1999, respectively.

He has served both as a Lecturer and a Researcher with several universities, such as the Universidad Complutense de Madrid, Universidad Nacional de Educación a Distancia (Open University), Madrid, and Saint Louis University, Madrid. He currently manages the Computer Architecture and Technology Group, Universidad Antonio de Nebrija, Madrid. His current activities are oriented to the space field, with several projects on reliability and radiation protection, as well as collaborations with the European Space Agency. He is the author of numerous technical publications, both in journals and international conferences. Aside from this, he has worked for several multinational companies, managing projects as a Project Management Professional and organizing support departments. His areas of interest include high-level synthesis and cosynthesis, signal processing and real-time systems, fault tolerance, and reliability.

Sanghyeon Baeg (S'91–M'95) received the B.S. degree in electronic engineering from Hanyang University, Seoul, Korea, in 1986, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas, Austin, in 1988 and 1991, respectively.

From 1994 to 1997, he was a Staff Researcher with Samsung Electronics Company, Kihung, Korea. In 1995, he was dispatched to Samsung Semiconductor, Inc., San Jose, CA, and worked as a member of the Technical Staff. In 1997, he joined Cisco Systems, Inc., San Jose, and worked as a Hardware Engineer, Technical Leader, and Hardware Manager. Since 2004, he has been an Associate Professor with the School of Electrical Engineering and Computer Science, Hanyang University, Ansan, Korea. His work has focused on reliable computing, low-power content-addressable memory, very large scale integration design-for-testability implementation and methodologies, silicon failure analysis, and yield enhancement. He is the holder of many U.S. patents in these fields.

Dr. Baeg received an Inventor Recognition Award from Semiconductor Research Cooperation in 1993. He was an IEEE 1149.6 working group member in 2003.