Mitigating the Effects of Large Multiple Cell Upsets (MCUs) in Memories

JUAN ANTONIO MAESTRO and PEDRO REVIRIEGO, Universidad Antonio de Nebrija
SANGHYEON BAEG, Hanyang University
SHIJIE WEN and RICHARD WONG, Cisco Systems

Reliability is a critical issue for memories. Radiation particles that hit the device can cause errors in some cells, which can lead to data corruption. To avoid this problem, memories are protected with per-word error correction codes (ECCs). Typically, single-error correction and double-error detection (SEC-DED) codes are used. As technology scales, errors caused by radiation particles on memories tend to affect more than one cell—what is known as a multiple cell upset (MCU). To ensure that only a single cell is affected in each word, interleaving is used. With interleaving, cells that belong to the same word are placed at a sufficient distance such that an MCU will only affect a single cell on each word. The use of interleaving significantly increases the cost of the device. Also, determining the interleaving distance (ID) required to avoid MCUs causing double errors is not trivial. Typically, accelerated radiation experiments with a limited number of particle hits are used. They provide a lower bound on the required ID, but larger MCUs may occur with a low probability. But even if the percentage of such large MCUs is very low, the impact on reliability can be significant. This article presents a technique to mitigate the effects of large MCUs that is, those that exceed the ID, on memory reliability. The proposed approach is able to correct most double errors caused by large MCUs by exploiting the locality of the errors within an MCU.

Categories and Subject Descriptors: B.3.4 [**Memory Structures**]: Reliability, Testing, and Fault-Tolerance; B.7.3 [**Integrated Circuits**]: Reliability and Testing; E.4 [**Data**]: Coding and Information Theory

General Terms: Design, Reliability

Additional Key Words and Phrases: Fault-tolerant memory, Error-correcting codes, high-level protection technique, protection against radiation

ACM Reference Format:
Maestro, J. A., Reviriego, P., Baeg, S., Wen, S., and Wong, R. 2011. Mitigating the effects of large multiple cell upsets (MCUs) in memories. ACM Trans. Des. Autom. Electron. Syst. 16, 4, Article 45 (October 2011), 10 pages.
DOI = 10.1145/2003695.2003705 http://doi.acm.org/10.1145/2003695.2003705

1. INTRODUCTION

With technology scaling, the occurrence of multiple cell upsets becomes increasingly common in memories, posing a threat to reliability, as discussed in Radaelli et al.

This research was supported by the Spanish Ministry of Education and Science under project AYA2009-13300-C03.
Authors' addresses: J. A. Maestro and P. Reviriego, Departamento de Ingeniería Informática, Universidad Antonio de Nebrija, Calle Pirineos 55, 28040 Madrid, Spain; email: {previrie, jmaestro}@nebrija.es. S. Baeg, School of Electrical Engineering and Computer Science, Hanyang University, 1271 Sa1-Dong Sangrok-Gu Ansan, Kyung-Gi-Do, Korea; email: bau@hanyang.ac.kr. S. Wen, and R. Wong, Component Engineering Group, Cisco Systems Inc. 170 W. Tasman Dr. San Jose CA 95134; email: shwen@cisco.com, rickwon@cisco.com.

DOI 10.1145/2003695.2003705 http://doi.acm.org/10.1145/2003695.2003705

[2005]; Tipton et al. [2006]; and Maiz et al. [2003]. The occurrence of multiple errors in a given event can lead to a failure, even when single-error correction and double-error detection codes (SEC-DED) are used to protect memories. The most common approach to protect memories from multiple errors has been the use of interleaving in the physical arrangement of the memory cells, such that cells that belong to the same logical word are separated. As the errors in an MCU are physically close, as discussed in Satoh et al. [2000], they will cause single errors in different words that can be corrected by the SEC-DED codes. This physical clustering of errors is due to the nature of the phenomena that causes the errors–namely, radiation particle hits—which cause errors that are localized along the trajectory of the particle's impact.

However, interleaving may have a large impact on floor-planning, access time, and power consumption, as discussed in Baeg et al. [2009] and Dutta and Touba [2007]. This overhead grows with the interleaving distance (ID) because the interconnections are longer and introduce more delay and capacitance. These limitations make the use of small ID values more attractive. The problem is that if the ID is not sufficiently large, a small percentage of MCUs can cause double errors. In the following, we refer to such MCUs as large MCUs.

The selection of the ID of a memory is typically based on accelerated radiation experiments, as discussed in Baeg et al. [2009]. In these experiments, devices of the target technology are exposed to radiation sources that cause many errors in a short period of time. The errors are then analyzed to determine the percentage of MCUs and their sizes. Using these results, an ID value larger than the larger observed MCU is selected. Typically, thousands of errors are recorded during the experiments, which means that larger MCUs than those observed may occur with a very small probability.

From a reliability point of view, the double errors caused by large MCUs can have a large impact. This is so because as soon as such an MCU occurs, an uncorrectable error and hence a failure occurs. In terms of the mean time to failure (MTTF), if λ is the error event arrival rate for the memory and p_L is the probability that an error event is a large MCU, then the MTTF for failures caused by large MCUs will be

$$MTTF|_L = \frac{1}{\lambda \cdot p_L}. \tag{1}$$

Obviously, if p_L is small, the MTTF will be large. The problem is that the MTTF for failures caused by other error events can also be large. In fact, SEC-DED codes are commonly combined with scrubbing to achieve large MTTFs, as discussed in Saleh et al. [1990] and Yang [1995]. Scrubbing consists of periodically reading the memory words to correct errors, which prevents errors from accumulating, and therefore increases the MTTF. The MTTF of a memory suffering single errors on which scrubbing is done with period t_s can be approximated, following Saleh et al. [1990], by

$$MTTF|_S = \frac{2 \cdot M}{\lambda^2 \cdot t_s} \tag{2}$$

where M is the number of words in the memory. This shows that by using small scrubbing periods, large MTTFs can be achieved. This approximation can be extended for MCUs when sufficiently large interleaving is used, as discussed in Reviriego et al. [2007]. However, scrubbing is not effective against large MCUs, as they cause a double error that cannot be corrected and removed. This means that even a small percentage of large MCUs can degrade the MTTF. Large MCUs will not have a significant impact when

$$MTTF|_S \ll MTTF|_L, \tag{3}$$

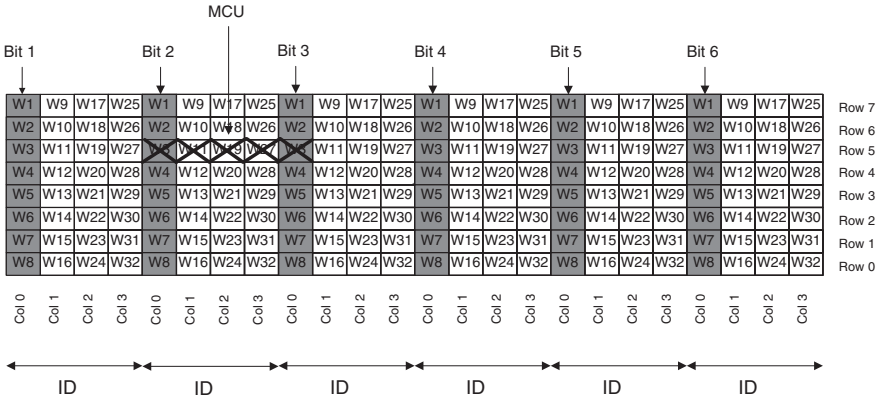

Fig. 1. Memory structure and example of a large MCU.

which results in

$$p_L \ll \frac{\lambda \cdot t_s}{2 \cdot M}. \tag{4}$$

For a terrestrial environment in which the device suffers an error per day and the scrubbing period is one hour for a 1M word memory, we have $p_{\mathrm{L}} \ll 2 \cdot 10^{-8}$. Clearly, even a very small amount of large MCUs can significantly degrade the MTTF.

In the rest of the article, a technique to mitigate the effects of large MCUs on memory reliability is proposed and analyzed.

2. PROPOSED TECHNIQUE

To present the proposed technique, the memory structure described in Radaelli et al. [2005], which is shown in Figure 1, will be used. In Figure 1, an ID value of 4 is used and only the first bits of each word are shown. An example of large MCU is also shown in Figure 1. In this case it affects bits 2 and 3 of word 3. The first interesting observation is that large MCUs cause errors on adjacent bits of the memory word, as noted in Dutta and Touba [2007]. The second observation is that it is likely that cells in between those bits also suffer errors. In the case of Figure 1, those errors occur on bit 2 of words 11, 19 and 27.

The errors caused by an MCU tend to be physically close, as they occur in the area affected by the particle hit, as discussed in Lawrence and Kelly [2008]. The errors occur along the trajectory of the particle, which means that it is likely that if two distant cells are corrupted by an MCU, then some of the cells that lie in between are also corrupted by the MCU. This is illustrated in Figure 2, where some of the MCU patterns reported in Radaelli et al. [2005] are shown. The pattern at the bottom-right will cause a double error when the memory structure of Figure 1 is used, and will also cause single errors in some words.

The proposed approach uses those observations to identify double errors caused by large MCUs and correct them. To do so, when an uncorrectable error is detected, the syndrome is checked to see if it corresponds to a double adjacent error. If that is the case, the words whose cells are in between the cells in error are read. If those words have errors in the bit that corresponds to the cells in between the double error, then we assume that a large MCU has occurred and the affected bits are corrected.

To explain the approach in more detail, a basic understanding of the SEC-DED codes is needed. These codes take an input word and compute a number of parity bits that are stored with the original data. Then, when the word is read, the parity bits

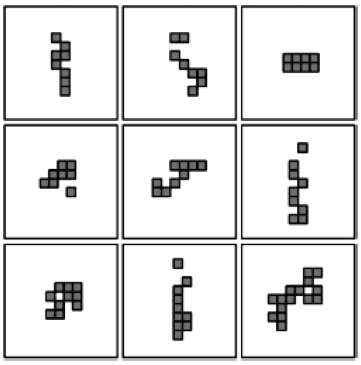

Fig. 2. Example of MCUs reported in Radaelli et al. [2005].

are recomputed and compared with the stored ones. The result of the comparison is known as the syndrome. If the syndrome is zero, then it is assumed that no error has occurred; otherwise an error has occurred. SEC-DED codes are designed such that the valid words have a minimum distance of four. This enables single-error correction and double-error detection, as discussed in Chen and Hsiao [1984]. This is done as follows: if the syndrome corresponds to one of the single-error syndromes, then the error is assumed to be a single error and is corrected. Otherwise, if the syndrome is not zero and does not correspond to a single-error syndrome, a failure is detected.

For large MCUs that affect two adjacent bits in a word, a failure will be detected. For an N-bit word, there will be N-1 combinations of two adjacent bits. Therefore, the syndromes that correspond to those errors can be computed, and when there is a failure, we can check if the syndrome matches one of those combinations. Then, that would give us the bits on which we would expect errors on the words that have their cells in between the ones that caused the double error. For the example in Figure 1, a failure will be detected in word 3, bits 2 and 3. This means that we expect some errors in bit 2 of words 11, 19, and 27. Therefore, we read those words, and if that is the case we can assume that the double error was caused by a large MCU and correct it.

The proposed scheme has to be analyzed in detail to ensure that the correction is only performed when the double error was caused by a large MCU. In the analysis, only the combinations of two error events are considered. This is reasonable, as most failures will be caused by two error events and, in any case, SEC-DED codes provide no guarantee of detecting even triple errors.

A necessary condition for a correction to take place is that there are errors in a specific bit of the words whose cells lie in between the one that suffered the double error. This means that an error event has occurred on that part of the device. This event may have affected none, one, or two bits of the word that has suffered the double error, as illustrated in Figure 3 for the case of a double error on bits 2 and 3. If it has affected two bits, then it would be the large MCU that we are correcting, so there is no issue. If it has affected one bit, then there must be an error on a different bit of word 5 (bit 5 in the figure) that is making the syndrome match that of the adjacent error on bits 2, 3. This would mean that an error on bits 2, 3 has the same syndrome as an error on bits 3,5. However, this would imply that an error on bits 2, 5 has a syndrome of zero, which is impossible, as the code is SEC-DED and all double errors have a syndrome that is not zero. Therefore, the syndrome will not match that of the double adjacent error, and therefore no correction will be done. Finally, if it has not affected bit 2 or 3, then there must be a large MCU on other two adjacent bits of the word (4 and 5 in the figure). In this case, there will be some double errors in words 11,19, and 27 that will

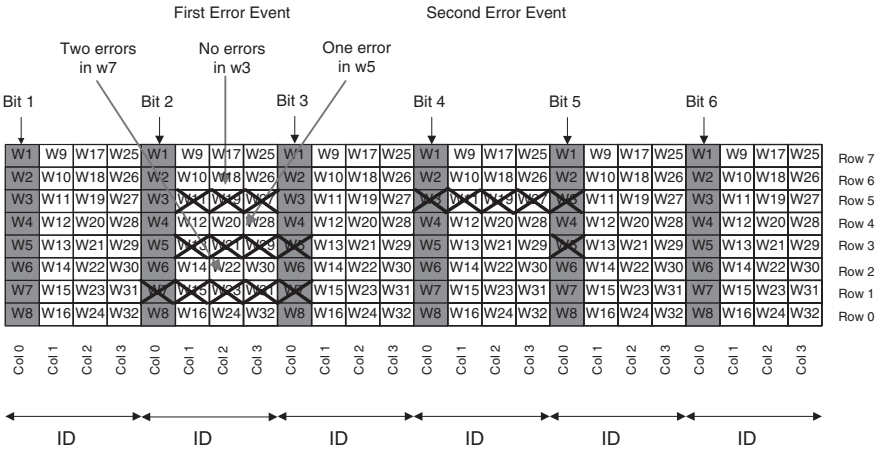

Fig. 3. Examples of MCU patterns affecting bit 2 of some words combined with a second error event.

cause a failure independently of the large MCU. Hence in all cases we only perform the correction when the error was caused by a large MCU.

The examples presented so far have considered large MCUs that exceed the ID by one, in our case that is an error that affects bits that are distributed over five columns. Larger MCUs will cause double errors in more than one word, and that complicates the detection and correction process. In the following, only large MCUs that exceed the ID by one are considered. This can be reasonable, as MCUs exceeding the ID should occur very rarely and, of those, most would exceed the ID by one, as discussed in Reviriego et al. [2010]. This means that correcting the MCUs that exceed the ID by one can provide a significant improvement in terms of reliability. The proposed approach may be extended to larger MCUs by using more complicated correction algorithms.

From the previous discussion, it becomes clear that the proposed scheme can correct all MCUs that exceed the ID by one when there is no other error event in the same memory row. This is very useful when scrubbing is used, as in that case most errors are removed shortly after they occur. Also, since the probability of having large MCUs is small, the probability of having a large MCU and another error event on the same row is small.

When there is a second error event in the same row in which a large MCU that exceeds the ID by one has occurred, there are combinations that can lead to an undetected failure. For example, when the second error event affects another bit of the word that suffered a double error as consequence of the large MCU, then that word has a triple error. This is illustrated in Figure 4. SEC-DED codes cannot guarantee the detection of triple errors, and some of them may be misinterpreted as single errors (as the code minimum distance is four) and miscorrected. This would cause a failure in which the data is corrupted and the error is not detected. That is, two events on the same memory row can cause an undetected failure when one of them is a large MCU. The proposed approach does not address these failures—however codes with reduced triple error miscorrection probability, such as the ones proposed in Richter et al. [2008], can be used to reduce the number of failures.

3. LARGE MCU CORRECTION ALGORITHM

The proposed scheme can be implemented as part of the scrubbing process. Figure 5 shows a block diagram of such an implementation. The memory words are read cyclically to remove errors. When a single error is detected, the word is corrected and written

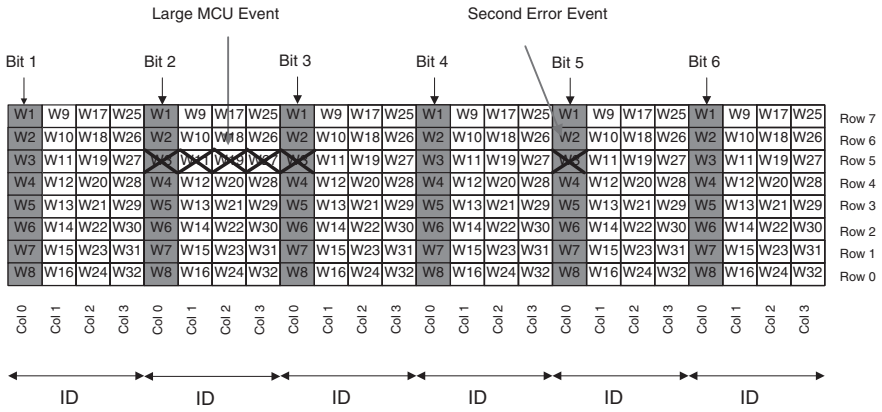

Fig. 4. Example of triple errors with two events.

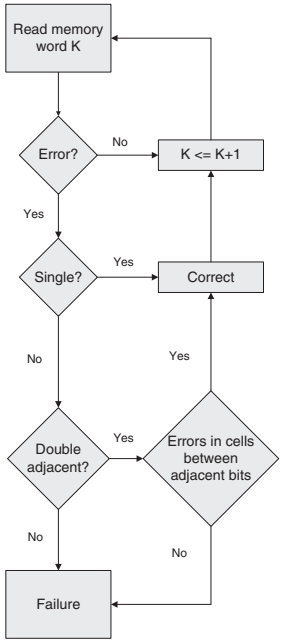

Fig. 5. Block diagram of the correction algorithm.

back to the memory. If the error is not a single error, then the syndrome is checked to see if it matches one of the values that correspond to double adjacent errors. If that is not the case, an uncorrectable error has occurred and the process reports a failure. If it matches the syndrome of an adjacent error, it means that a large MCU may have occurred. Then, the bits in error are detected from the syndrome value. Using those bits, the bit in which there should be errors in words affected by the large MCU is determined. Those words are the ones that have bits in cells that are physically between the cells of the two adjacent bits in error. Finally, those words are read. If there are errors on those words in the expected bit, then the process assumes that a large MCU has occurred and corrects the adjacent errors. If there are no errors on those words in that bit, a failure is reported.

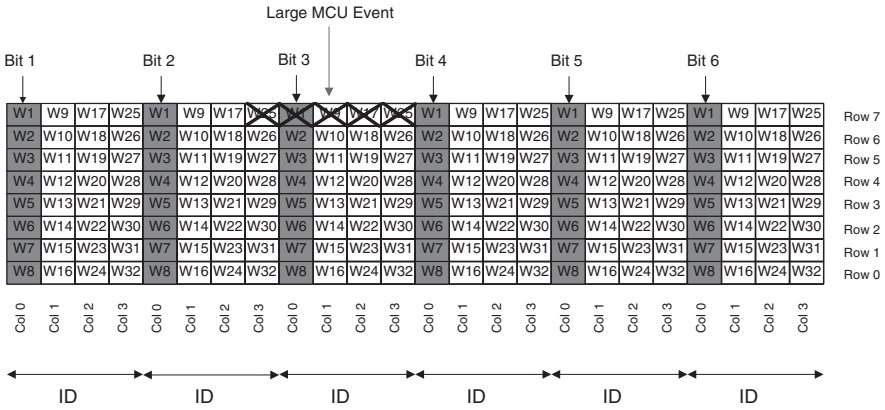

Fig. 6. Example of large MCU.

Depending on the order in which the memory words are read, the algorithm may fail to correct the large MCU. This would occur when the single errors caused by a large MCU are corrected before the word that contains the double error is read. This would cause the correction algorithm to fail, which is illustrated in Figure 6 where a large MCU caused a double error on bits 2 and 3 of word 25. If words are read sequentially from the first one, then the errors on words 1, 9, and 17 would be cleared before the double error is detected when reading word 25. Therefore, no errors will be found in bit 3 of those words and the double error would not be corrected. This problem can be avoided by recording the errors that have been detected during the scrubbing round, such that if the words that need to be checked have already been scrubbed, the process can check the list of errors to determine if the were errors on them.

It should be noted that the proposed algorithm only incurs in additional processing and memory accesses when a double error is detected. In most scrubbing operations, the word read will have no errors. This is so because the scrubbing process avoids the accumulation of errors in the memory. Therefore, the probability of finding an error will be low. Additionally, most errors will be single errors for which our technique has no added overhead. This means that the processing described to correct large MCUs will occur in a very small fraction of the scrubbing operations. Hence its impact on performance is expected to be negligible.

The algorithm is simple, and can be easily adopted in systems in which the scrubbing process is implemented in software running in a processor. For systems in which scrubbing is embedded in the memory devices and implemented in hardware, the proposed approach is less attractive.

4. RELIABILITY ANALYSIS

In this section, the proposed scheme is analyzed in terms of its benefits on the mean time to failure (MTTF). To that end, it is assumed that error events occur randomly, following a Poisson process, and are uniformly distributed across the memory words, as in previous studies like, for example, Saleh et al. [1990]. These are reasonable assumptions when the error events are caused by radiation particle hits that are not correlated and can affect any part of the device. The error events can be single errors or MCUs. For MCUs, only horizontal errors are considered, and the probability that an event is a large MCU that exceeds the ID will be denoted as p_L. The proposed technique only protects against MCUs that exceed the ID by one. If the probability of

Table I. Estimates of p[k] for the Examples

	p[1]	p[2]	p[3]	p[4]	p[5]	p[6]	p[7]	p[8]
Example 1	0.95	0.049	0.001	0.000	0.000	0.000	0.000	0.000
Example 2	0.80	0.150	0.040	0.01	0.000	0.000	0.000	0.000
Example 3	0.65	0.250	0.080	0.019	0.001	0.000	0.000	0.000

such MCUs is denoted as p[ID+1], obviously, p[ID+1] $\leq$ p_L, as large MCUs include those that exceed the ID by one or more. In fact, if we refer to p[k] as the probability that an error event spans exactly k columns, then

$$p_L = \sum_{k=ID+1}^{\infty} p[k]. \tag{5}$$

When the proposed technique is not used, the MTTF for failures caused by large MCUs will be given by

$$MTTF|_L = \frac{1}{\lambda \cdot p_L} = \frac{1}{\lambda \cdot \sum_{k=ID+1}^{\infty} p[k]}. \tag{6}$$

When the proposed scheme is used, errors that exceed the ID by one are corrected, and therefore the MTTF for failures caused by large MCUs increases to

$$MTTF|_L^{\text{Proposed}} = \frac{1}{\lambda \cdot (p_L - p[ID+1])} = \frac{1}{\lambda \cdot \sum_{k=ID+2}^{\infty} p[k]}. \tag{7}$$

Finally, the ratio of both MTTFs would be

$$\frac{MTTF|_L^{\text{Proposed}}}{MTTF\|_L} = \frac{\sum_{k=ID+1}^{\infty} p[k]}{\sum_{k=ID+2}^{\infty} p[k]} = 1 + \frac{p[ID+1]}{\sum_{k=ID+2}^{\infty} p[k]}. \tag{8}$$

This shows the improvement of using the proposed technique to mitigate the effects of large MCUs. The benefit can be significant, as the values of p[k] typically decrease as k increases.

The actual MTTF of the memory will depend on both the failures caused by large MCUs and those caused by error accumulation, as discussed in Reviriego et al. [2010]. Increasing the MTTF due to failures caused by large MCUs will limit the impact of large MCUs on the final MTTF. In fact, when Eq. (3) is met, the effect of large MCUs will be negligible.

5. APPLICATIONS

This section discusses the benefits of the proposed technique in different examples. The examples assume that there are estimates of the p[k] obtained through accelerated radiation testing or based on previous designs that used the same technology. Those estimates can differ from the values observed in the field, due to the limited number of experiments or to different environmental conditions.

The estimates of p[k] for the examples are shown in Table I. With those values the ID would be at least four in the first example, five in the second, and six in the third one. In many designs, only ID values that are a power of two are implemented, as discussed in Reviriego et al. [2010]. One reason is that, in many cases, memory compilers are used to generate the memory blocks and those compilers typically only support a limited number of ID values. This means that, in the first example, an ID of four will be used, while the ID will be eight for the other two cases.

Let us now consider a situation where the real values of p[k] differ from the estimates, and take the values given in Table II. This would cause failures due to large MCUs in example one. For the other two examples, the ID would be sufficient to avoid large

Table II. Real Values of p[k] for the Examples

	p[1]	p[2]	P[3]	p[4]	p[5]	p[6]	p[7]	p[8]
Example 1	0.90	0.090	0.006	0.003	0.001	0.000	0.000	0.000
Example 2	0.65	0.200	0.100	0.045	0.004	0.001	0.000	0.000
Example 3	0.60	0.250	0.100	0.040	0.006	0.003	0.001	0.000

MCUs. If the proposed technique is used, the large MCUs in example one would be corrected so that, effectively, they will not cause failures. For the other examples, the proposed technique will provide no benefit.

In the first example, the margin from the largest value of k for which the estimated p[k] is larger than zero to the ID is one. This margin is four and three in the second and third examples. This means that small deviations in the observed p[k] can cause large MCUs in the first example. The proposed technique adds some additional margin, as it can correct errors that span ID+1 columns. That is, the margin would go up to two in the first example. In summary, the proposed technique is especially useful when the estimated p[k] are such that a given ID covers all the MCUs, but with little margin.

The benefits of the technique in terms of MTTF can be illustrated for the first example by assuming a memory size of 10^6 words, an error arrival rate of one error per day and a scrubbing period of one hour. Then, the MTTF for failures due to error accumulation can be calculated using Eq. (2). The value obtained is 48000000 days. If large MCUs are not corrected, the MTTF for failures due large MCUs can be calculated using Eq. (1). The value obtained is 1000 days. In this case, failures caused by large MCUs would clearly dominate and limit the overall MTTF to 1000 days. If the proposed technique is used, then large MCUs are corrected and do not affect the overall MTTF that would be limited only by error accumulation (48000000 days, as calculated before). This example clearly shows how the proposed technique can bring substantial benefits in some cases.

6. CONCLUSIONS

In this article, a technique to mitigate the effects of large MCUs on memory reliability has been presented. The proposed approach can correct double errors caused by MCUs that exceed the interleaving distance by one. This provides some extra margin when selecting the interleaving distance in a design. The usefulness of the technique has been illustrated with some examples. The proposed scheme could be extended to provide additional margin, but at the cost of more complex correction algorithms.

REFERENCES

Baeg, S., Wen, S., and Wong, R. 2009. Interleaving distance selection with a soft error failure model. *IEEE Trans. Nuclear Sci. 56*, 4, 2111–2118.

Chen, C. L. and Hsiao, M. Y. 1984. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. *IBM J. Res. Dev. 28*, 2, 124–134.

Dutta, A. and Touba, N. A. 2007. Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code. In *Proceedings of the IEEE VLSI Test Symposium*. IEEE Los Alamitos, CA, 349–354.

Lawrence, R. K. and Kelly, A. T. 2008. Single event effect induced multiple-cell upsets in a commercial 90 nm CMOS digital technology. *IEEE Trans. Nuclear Sci. 55*, 6, 3367–3374.

Maiz, J., Hareland, S., Zhang, K., and Armstrong, P. 2003. Characterization of multi-bit soft error events in advanced SRAMs. In *Proceedings of the IEEE International Electron Devices Meeting*. IEEE, Los Alamitos, CA, 21.4.1–21.4.4.

Radaelli, D., Puchner, H., Wong, S., and Daniel, S. 2005. Investigation of multi-bit upsets in a 150 nm technology SRAM device. *IEEE Trans. Nuclear Sci. 52*, 6, 2433–2437.

Reviriego, P., Maestro, J. A., Baeg, S., Wen, S., and Wong, R. 2010. Protection of memories suffering MCUs through the selection of the optimal interleaving distance. *IEEE Trans. Nuclear Sci. 57*, 4, 2124–2128.

Reviriego, P., Maestro, J. A., and Cervantes, C. 2007. Reliability analysis of memories suffering multiple bit upsets. *IEEE Trans. Device Materials Reliability 7*, 4, 592–601.

RICHTER, M., OBERLAENDER, K., AND GOESSEL, M. 2008. New linear SED-DED codes with reduced triple bit error miscorrection probability. In *Proceedings of the 14th IEEE International On-Line Testing Symposium* (*IOLTS*). IEEE, Los Alamitos, CA, 37–42.

SATOH, S., TOSAKA Y., AND WENDER, S. A. 2000. Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAMs. *IEEE Electron Device Lett. 21*, 6, 310–312.

SALEH, A. M., SERRANO, J. J., AND PATEL, J. H. 1990. Reliability of scrubbing recovery-techniques for memory systems. *IEEE Trans. Reliability 39*, 1, 114–122.

TIPTON, D., PELLISH, J. A., REED, R. A., SCHRIMPF, R. D., WELLER, R. A., MENDENHALL, M. H., SIERAWSKI, B., SUTTON, A. K., DIESTELHORST, R. M., ESPINEL, G., CRESSLER, J. D., MARSHALL, P. W., AND VIZKELETHY, G. 2006. Multiple-bit upset in 130 nm CMOS technology. *IEEE Trans. Nuclear Sci. 53*, 6, 3259–3264.

YANG, G. C. 1995. Reliability of semiconductor RAMs with soft-error scrubbing techniques. *IEE Proc. Comput. Digital Tech. 142*, 5, 337–344.

Received December 2010; revised March 2011; accepted June 2011