

# Designing Ad-hoc Scrubbing Sequences to Improve Memory Reliability Against Soft Errors

Pedro Reviriego  
 Universidad Antonio de Nebrija  
 C/ Pirineos 55  
 28040 Madrid, Spain  
 previrie@nebrija.es

Juan Antonio Maestro  
 Universidad Antonio de Nebrija  
 C/ Pirineos 55  
 28040 Madrid, Spain  
 jmaestro@nebrija.es

Sanghyeon Baeg  
 Hanyang University  
 1271 Sa1-Dong Sangrok-Gu Ansan  
 Kyung-Gi-Do, Korea  
 bau@hangyang.ac.kr

## ABSTRACT

In this paper, we propose the use of ad-hoc scrubbing sequences to improve memory reliability. The key idea is to exploit the locality of the errors caused by a Multiple Cell Upset (MCU) to make scrubbing more efficient. The starting point is the MCU distributions for a given device. A procedure is presented that uses that information to determine an ad-hoc scrubbing sequence that maximizes reliability. The approach is then applied to a case study and results show a significant increase in the Mean Time To Failure (MTTF) compared with traditional scrubbing.

## Categories and Subject Descriptors

B.8.1 [Performance and Reliability]: Reliability, Testing and Fault-Tolerance.

## General Terms

Reliability.

## Keywords

Multiple Cell Upsets (MCUs), memories, reliability, scrubbing, radiation.

## 1. INTRODUCTION

Reliability is a major concern for memories and more so as technology scales. Soft errors caused by radiation particles are an important threat to memory reliability [1]. A soft error changes the value of a memory cell without causing permanent damage to the circuit. This can lead to data corruption and therefore Error Correction Codes (ECCs) are typically used to protect memories from soft errors. Per word Single Error Correction, Double Error Detection (SEC-DED) codes are commonly used such that single errors in a word can be corrected [2].

When small geometries are used to manufacture memories, a radiation particle can create more than one cell upset, this is

known as Multiple Cell Upsets (MCU). MCUs pose a challenge to SEC-DED codes as they can affect two bits of the same word causing an uncorrectable error. To protect memories from MCUs, SEC-DED codes are typically combined with interleaving [3] such that bits that belong to the same word are placed in cells that are physically apart. For example, the scheme used in [3] is illustrated in Figure 1, where only the first three bits of each word are shown. As the errors caused by an MCU are physically close [4], if interleaving is used they affect single bits of different words. Those errors can be corrected by the SEC-DED code in each word. An example of an MCU affecting two cells is shown in Figure 1. In this case bit 2 on words 11 and 12 suffer an error, but both are correctable with the SEC-DED code.

Even if SEC-DED codes and interleaving are used, failures can occur due to error accumulation. That is a particle hit creates an error in a bit of a given word and after some time another particle hit creates another error on a different bit. To reduce the probability of failure, the memory words could be read periodically such that when a single error is detected, the word is written back with the correct data. This process is called scrubbing and it is commonly used to increase memory reliability [5]. Scrubbing however reduces the effective memory bandwidth available to the system and increases power consumption.

In a typical implementation scrubbing uses a small percentage of the memory accesses to cyclically read and correct the memory. The time required to complete a cycle is denoted as the scrubbing period ( $T_s$ ). During a scrubbing period, the read operations are uniformly distributed. This reduces the impact on the system as the memory can be used at all times having to wait a only few extra cycles in the worst case (when a scrubbing operation is ongoing).

In [6], an idea was proposed to optimize scrubbing when MCUs are a relevant fraction of the error events. The proposal was to modify the scrubbing sequence so that memory words are not read sequentially. Instead the words are divided in two groups and each group is read sequentially. When one group is complete scrubbing proceeds with the other one. If during the process an error is detected nearby words are also checked immediately to clear errors. This increases the reliability as most errors caused by MCUs are cleared faster than with traditional scrubbing. In [6], it was shown that this scheme could increase the MTTF of memories significantly.

In this paper the approach is taken one step further by designing an ad-hoc scrubbing sequence for a given memory. The ad-hoc sequence is tailored to the specific MCU patterns observed in the

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC'11, June 5-10, 2011, San Diego, California, USA  
 Copyright © 2011 ACM 978-1-4503-0636-2/11/06...\$10.00

device and can therefore achieve even larger increases in the MTTF. The rest of the paper is organized as follows, in section two a procedure to select a scrubbing sequence that increases the reliability is presented and the benefits of the proposed approach in terms of the increase in MTTF is evaluated. The use of the

proposed procedure is illustrated by means of a case study in section three. Finally the conclusions are presented in section four.

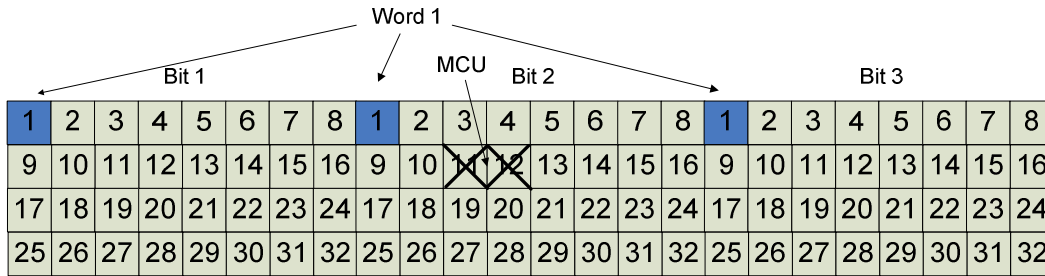


Figure 1. Example of memory organization with interleaving.

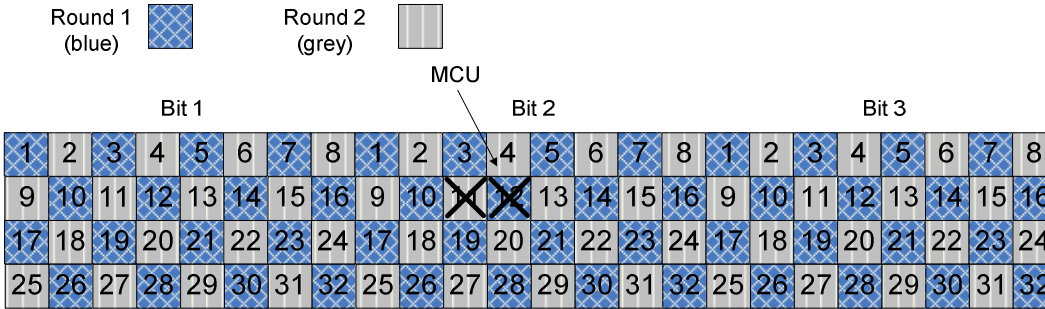


Figure 2. Alternative scrubbing sequence proposed in [6].

2. DESIGNING SCRUBBING SEQUENCES

In this section the design of scrubbing sequences that optimize memory reliability is considered. In the following, it is assumed that MCUs that affect cells that are all adjacent are dominant. This is the case for example in [3]. More generally errors in an MCU are typically [4] physically close which is in line with the assumption. Therefore the scrubbing is optimized to reduce the time required to clear those MCUs. For this a procedure is presented to determine the best scrubbing sequence in a systematic manner.

To discuss the benefits of modified scrubbing sequences the scheme proposed in [6] will be used. This scheme is illustrated in Figure 2. The scrubbing cycle is divided in two rounds such that most MCUs are detected in a single round. An example is shown in Figure 2 where an MCU has an error on each round so that it will be detected (and corrected) in a single round. If the error on word 11 is detected first then cells 3,10,12,19 will be checked immediately so that the error on word 12 is also cleared in that round. In general when an MCU produces errors on cells that belong to two rounds the average time to clear the errors is reduced to one half. However for MCUs that affect only cells that belong to one round there is no benefit. To illustrate these effects the time to clear an MCU that occurs on the first word of the memory (that is scrubbed at the beginning of each round) is illustrated in Figure 3. When the MCU produces errors in one round only it will take up to  $T_s$  to clear the error. This worst case occurs when the scrubbing process happens to be at round one ( $R_1$ ) and has checked the first word just before the error occurs. When the scrubbing is checking other parts of the memory the

time to clear the error is given in Figure 3. If the MCU produces errors on both rounds, the errors are cleared at the very beginning of each round reducing the time it stays in the memory.

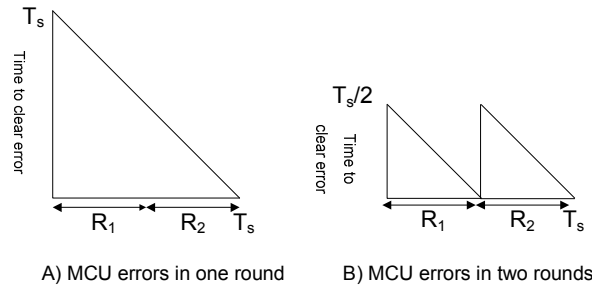


Figure 3. Distributions of the time to clear an MCU depending on whether it has errors in one or two rounds.

To compare scrubbing sequences, the reduction in the average time that an error stays in the memory will be used and the sequence with the largest reduction will be the one selected. The reduction can be computed as follows

$$R_{total} = \frac{\sum_{i=1}^N \left( i \cdot p(i) \cdot \sum_j (p_i(j) \cdot R_i(j)) \right)}{\sum_{i=1}^N (i \cdot p(i))} \tag{1}$$

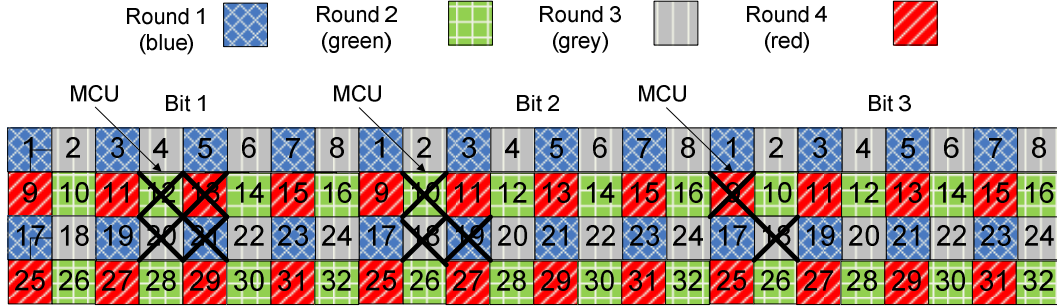


Figure 4. Ad-hoc scrubbing sequence with four rounds.

where  $p(i)$  is the percentage of  $i$ -bit MCUs and  $p_i(j)$  the percentage of  $i$ -bit MCUs that have pattern of error  $j$ . Finally  $R_i(j)$  is the reduction factor for pattern. For the scheme in Figure 2 the reduction factor for two bit MCUs will be 2 for adjacent horizontal and vertical errors and 1 for diagonal errors. This means that the scheme is suited for memories in which two bit horizontal and vertical MCUs are dominant.

As our assumption is that MCUs that affect cells that are all adjacent are dominant, the scrubbing sequences can be limited to those that cover a square of two by two cells. All error patterns in which all cells are adjacent will be covered by such pattern. The scrubbing sequence can then be decomposed in two or four rounds. In Figure 4 an example of a four round sequence is shown and also some examples of MCU patterns with errors in adjacent cells are illustrated. In this case an MCU can have errors in one, two, three or four rounds. The time that an MCU stays in the memory in each case is shown in Figure 5. It can be observed that for MCUs with errors in two rounds, the average time will be worse than for the scheme proposed in [6] when the errors are in rounds that are at a distance of one. On the other hand the average time is reduced for MCUs that have errors on three or four rounds. This is clearly observed in the reduction factor for each of the cases in Figures 3 and 5 that are shown in Table 1.

Table 1. Reduction Factors ( $R_f$ )

Two Rounds (Figure 3)	Case A)	1.00
	Case B)	2.00
Four Rounds (Figure 5)	Case A)	1.00
	Case B)	2.00
	Case C)	1.60
	Case D)	2.67
	Case E)	4.00

The sequences that have different average times to clear errors are shown in Figure 6 for two and four rounds. All other sequences can be shown to have the same average time to clear errors as one of the sequences in Figure 6. Depending on the sequence some error patterns will be removed faster. For example if the leftmost sequence is used, horizontal and diagonal 2 bit adjacent MCUs will have errors on two rounds while vertical ones will have errors on one round only. The same applies to the four round sequences, for example the rightmost one will have errors on rounds at

distance two for vertical errors and at distance one for horizontal and diagonal errors. Therefore it is suited for memories in which vertical errors are dominant.

Assuming that the error patterns and their distributions are known, the best scrubbing sequence can be selected by computing (1) for all the sequences in Figure 6 and selecting the one that produces the largest value. The reduction factors for each pattern  $R_i(j)$  can be directly taken from Table 1.

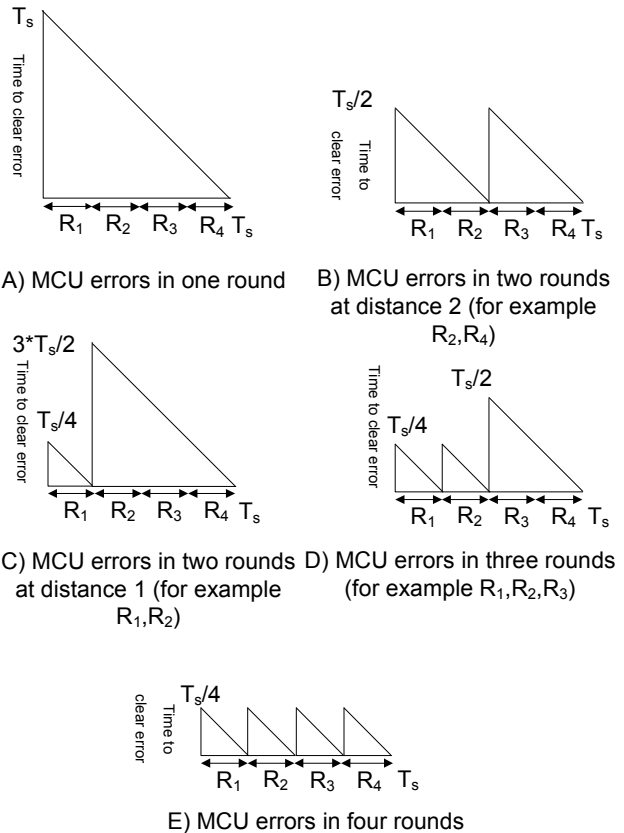
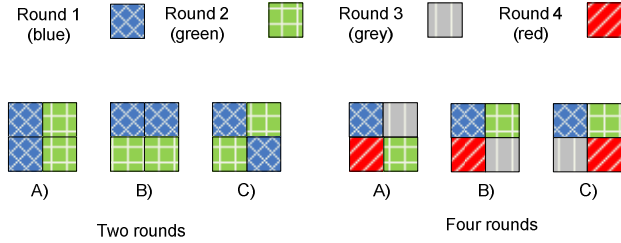


Figure 5. Distributions of the time to clear an MCU for a four round scheme.



**Figure 6. Different sequences for two and four rounds.**

Memories are typically characterized using accelerated radiation testing before they are qualified for production. Simulations are also performed in the design phase [7]. During those tests, the number of errors, the error patterns, etc. are recorded. That information is then analyzed and used to improve the soft error resilience of future designs and to understand the reliability of the devices tested. The results are also important to select an optimal interleaving distance [8]. The results from accelerated radiation testing can also be used to design an ad-hoc scrubbing sequence that maximizes the MTTF of the memory using the procedure presented. This means that no extra testing is need to apply the proposed approach.

To evaluate the benefits of the proposed approach, the Mean Time to Failure (MTTF) obtained can be compared with that of traditional scrubbing. The MTTF of memories that use SEC and scrubbing has been widely studied in the literature (see for example [5]), showing that when the scrubbing period  $T_s$  is small, it can be approximated by

$$MTTF \cong \frac{T_s}{P_f} \quad (2)$$

where  $P_f$  is the probability of failure in a scrubbing period.  $P_f$  can be approximated as follows, assuming a per-word error event arrival rate  $\lambda$  and a memory of  $M$  words, the number of errors that will have accumulated, on average, at any given time instant will be:

$$e_{accum} = \frac{\lambda \cdot M \cdot T_s}{2} \quad (3)$$

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:

$$P_f^{event} \cong \frac{e_{accum}}{M} = \frac{\lambda \cdot M \cdot T_s}{2M} = \frac{\lambda \cdot T_s}{2} \quad (4)$$

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:

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

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

$$MTTF_{traditional} \cong \frac{T_s}{P_f} \cong \frac{2}{\lambda^2 \cdot T_s \cdot M} \quad (6)$$

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 [9], 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:

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

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 expression (7)):

$$MTTF_{traditional} \cong \frac{T_s}{P_f} \cong \frac{2}{(\lambda \cdot E_{event})^2 \cdot T_s \cdot M} \quad (8)$$

When the proposed scheme is used  $e_{accum}$  will be reduced by  $R|_{total}$  given by (1). The increase in the MTTF compared with traditional scrubbing can then be computed as:

$$\frac{MTTF_{proposed}}{MTTF_{traditional}} = R|_{total} \quad (9)$$

The increase will be larger when MCUs are a significant percentage of the error events as it is for those that the proposed approach provides a benefit.

In summary the proposed procedure for designing the scrubbing sequence us as follows:

1. Determine the particles relevant for the environment in which the device will operate. For example for terrestrial applications neutrons and alpha particles.
2. Perform accelerated radiation testing to determine the MCU sizes, patterns and relative frequencies of occurrence.
3. Based on the results of step two use (1) to determine the best scrubbing sequence by selecting the one with the largest reduction factor.

### 3. CASE STUDY

In this section, the proposed approach is applied to a specific device for which the error patterns and distributions are publicly available [3].

The devices tested in [3] have the memory organization shown in Figure 1 and were exposed to neutron beams. The distribution of the sizes of the error events for each value of the particle energy used in the testing are shown in Table 2. It can be observed that MCU of size two are the larger proportion of MCUs and that the proportion decreases as the size increases. This is in line with other studies [7]. Tables 3 to 5 show the error patterns observed for each category. Horizontal errors are dominant for two bit MCUs. For larger MCUs, errors tend to affect adjacent cells this is in line with our assumptions.

**Table 2. Distribution of error sizes (percentages)**

Size	Single	MCU-2	MCU-3	MCU-4	Rest
22MeV	73	20	5	1.5	0.5
47MeV	64.5	23	9	2.5	1
95MeV	57.5	23	12	5	0.25
144MeV	53	25	13	6	3

**Table 3. 2-bit MCU error patterns reported in [3] (Normalized to 1000)**

Energy	Double-bit upset type			
				Others
22MeV	773	136	80	11
47MeV	681	180	117	22
95MeV	653	192	132	23
144MeV	686	156	133	25

**Table 4. 3-bit MCU error patterns reported in [3] (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

Using the information in Tables 2 to 5 on the MCU patterns and frequencies and applying the reduction factors in Table 1, the total reduction factors have been computed using (1) for all the sequences in Figure 6. Then the sequence with the largest reduction factor is selected as the best scrubbing sequence. The result is that the one shown in Figure 4 is the best sequence. This

sequence has a reduction factor of 2 for 2 bit horizontal MCUs and greater for most 3 and 4 bit MCUs. Therefore it adapts well to the error patterns in Tables 3 to 5.

**Table 5. 4-bit MCU error patterns reported in [3] (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

To illustrate the benefits of the proposed approach, in the following the MTTF increase is calculated from the reduction factors using (9). The results obtained for the MTTF increase are shown in Table 6 for all the sequences in Figure 6. These results are conservative in that for MCUs of size larger than four no reduction factor is used as no details on the error patterns is given in [3]. The third row corresponds to the modified scrubbing using two rounds proposed in [6] while the fourth is the one selected using the proposed procedure. It can be observed that in both cases the increase is significant compared to conventional scrubbing and that the proposed scheme provides an additional increase. It can also be observed how the benefit of the proposed approach increases with the energy of the Neutron. This can be explained as the proportion and size of the MCUs increase with particle energy [10]. An interesting observation is that the two rounds sequence B) on Table 6 produces the smallest MTTF increment. This is because in this case, 2 bit horizontal MCUs, which are dominant, have errors only in one scrubbing round so that their reduction factor is one. Another interesting observation is that the variation of the increase for the different scrubbing sequences is smaller for four scrubbing rounds than for two rounds. This is due to the fact that the reduction factors for 2 bit MCUs are closer with four rounds (1.6,2) than with two rounds (1,2) so that there is less dependency on the error patterns.

**Table 6. MTTF Increase for the sequences in Figure 6**

Neutron energy		22MeV	47MeV	95MeV	144MeV
Two rounds	A)	25.27%	31.72%	36.11%	39.96%
	B)	12.08%	19.58%	25.29%	26.47%
	<b>C) (Figure 2)</b>	<b>26.60%</b>	<b>33.34%</b>	<b>37.55%</b>	<b>40.43%</b>
Four rounds	<b>A) (Figure 4)</b>	<b>31.11%</b>	<b>41.39%</b>	<b>48.56%</b>	<b>52.93%</b>
	B)	26.86%	38.26%	46.21%	50.10%
	C)	27.19%	38.97%	46.99%	50.65%

From the results in Table 6 it may seem that using four rounds is always more efficient than using two rounds as all two round patterns have smaller MTTF increments than the worst four round pattern. This is not true as there are other cases for which a two round sequence will be better. For example if 2 bit horizontal and vertical MCU are overwhelmingly dominant the two round sequence C) in Figure 6 will provide a reduction factor of 2 for both while any four round sequence can only provide a factor of 2 for one of them and of 1.6 for the other. In that case a two round sequence will provide a larger MTTF increment than the four round sequences. Therefore the design procedure should evaluate all the sequences in Figure 6 to guarantee that the best one is selected.

For the case study considered, a relevant increase in MTTF can be achieved using the proposed approach. As technology scales MCU proportion and size will continue to grow [7], this means that the benefits will be even larger. If eventually MCU of sizes larger than two become dominant (something that is not expected in the near future [7]) then more complex scrubbing sequences that use a larger number of rounds could be useful to further increase the MTTF. The overall process would be similar but using more rounds that cover a larger number of cells (for example a square of four by four) and having to compute the reduction factor for a larger number of sequences as there will be more combinations that produce different results.

#### 4. CONCLUSIONS

In this paper the use of ad-hoc scrubbing sequences has been proposed to increase memory reliability. The results show that when MCUs are a relevant fraction of the error events the technique can provide significant increases in the MTTF. The information needed to design the ad-hoc scrubbing sequence (MCU sizes and patterns) is typically available as part of the device characterization process so that no additional testing is required. With that information a procedure to select an ad-hoc scrubbing sequence that increases the MTTF has been proposed. This process has been illustrated with a realistic case study. Finally it is worth noticing that as technology scales the benefits of ad-hoc scrubbing will increase making it more attractive. This is a direct consequence of the increase in MCU proportion and size with technology scaling.

#### 5. ACKNOWLEDGEMENTS

This work was supported by the Spanish Ministry of Science and Education under Grant AYA2009-13300C03 and by the GRRC

program of Gyeonggi province ([GRRC Hanyang2010-A01], developing sensor network SoC equipped with low power and high reliability characteristics).

#### REFERENCES

- [1] Baumann, R., *Soft errors in advanced computer systems*, IEEE Design and Test of Computers, 22, 3, (May/June2005), pp. 258–266,
- [2] Goodman R.M. , and Sayano M., *The reliability of semiconductor RAM memories with on-chip error-correction coding*, IEEE Trans. on Information Theory, 37, 3, (May1991), pp. 884 - 896.
- [3] Radaelli D., Puchner H., Wong S. and Daniel S., *Investigation of multi-bit upsets in a 150 nm technology SRAM device*, IEEE Trans. on Nuclear Science, 52, 6, (Dec. 2005), pp. 2433–2437.
- [4] Satoh S., Tosaka Y., Wender S.A., *Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAM's*, IEEE Electron Device Letters, 21, 6, (Jun 2000), pp. 310 – 312.
- [5] Saleh A.M., Serrano JJ., and Patel J.H., *Reliability of scrubbing recovery-techniques for memory systems*, IEEE Trans. on Reliability, Vol. 39, Issue 1, 1990, pp. 114 - 122.
- [6] Reviriego, P., Maestro, J.A. and Baeg S., *Optimizing Scrubbing Sequences for Advanced Computer Memories* IEEE Transactions on Device and Materials Reliability, 10, 2, (June 2010), pp. 192-200.
- [7] Ibe E., Taniguchi, H., Yahagi, Y., Shimbo K. and Toba T., *Impact of Scaling on Neutron-Induced Soft Error in SRAMs From a 250 nm to a 22 nm Design Rule*, IEEE Trans on Electron Devices, 57 , 7, (July 2010) ,pp. 1527 – 1538.
- [8] Baeg, S., Wen, S. and Wong R., *SRAM Interleaving Distance Selection with a Soft Error Failure Model*, IEEE Trans on Nuclear Science, 56, 4, (Aug 2009), pp. 2111 – 2118.
- [9] Reviriego P., Maestro, J. A., and Cervantes C., *Reliability analysis of memories suffering multiple bit upsets*, IEEE Trans. On Device and Materials Reliability, 7, 4, (Dec 2007), pp. 592-601.
- [10] Maestro, J.A. and Reviriego, P., *Study of the effects of MBUs on the reliability of a 150 nm SRAM device*, 45th ACM/IEEE Design Automation Conference (DAC), 2008 , pp. 930 – 935.