Matrix-Based Codes for Adjacent Error Correction

Costas A. Argyrides, Pedro Reviriego, Dhiraj K. Pradhan, and Juan Antonio Maestro

***Abstract*—Memories are one of the most widely used elements in electronic systems, and their reliability when exposed to single events upsets (SEUs) has been studied extensively. As transistor sizes shrink, multiple cells upsets (MCUs) are becoming an increasingly important factor in the reliability of memories exposed to radiation effects. To address this issue, built-in current sensors (BICS) or Parity codes have recently been applied in conjunction with single error correction/double error detection (SEC-DED) codes to protect memories from MCUs. In this paper, this approach is taken one step further, proposing specific codes optimized to provide protection against errors in adjacent bits in memories. By exploiting the locality of errors within an MCU and the error detection and location capabilities of parity codes, the proposed codes result in both a better protection level and a reduced cost. This technique improves memory reliability by 675X compared to Hamming Codes (HC) and 38X the mean time to failure (MTTF) compared to Reed Muller Codes (RMC) for clustered MCUs.**

***Index Terms*—Fault-tolerance, reliability, testing.**

I. Introduction

THE technology scaling process provides high-density, low cost, high-performance integrated circuits. These circuits are characterized by high operating frequencies, low voltage levels and small noise margins and will be increasingly sensitive to temporary faults [1]. In very deep sub-micron technologies, the product's field-level reliability is severely impacted by single-event upsets (SEU), like atmospheric neutrons and alpha particles. This does not only affect memories but logic as well. When these particles hit the silicon bulk, they create minority carriers, which if collected by the source/drain diffusions, could change the voltage level of the node.

Transient faults are also a major concern in space applications, with potentially serious consequences for the spacecraft, including loss of information, functional failure or loss of control [2]. Although SEUs are a major concern in space and terrestrial applications, multiple cell upsets (MCU) have also become an important problem in the memory design process. It has become an important problem because the error rate of memories is increased due to the continuing technology shrinkage [3], [4] and therefore the probability of having multiple errors increases.

Manuscript received September 07, 2009; revised December 02, 2009 and February 03, 2010; accepted February 06, 2010. Date of current version August 18, 2010. This work was supported in part by the Spanish Ministry of Science and Innovation under Grant AYA2009-13300-C03-01, the Regional Government of Madrid, and in part by the European Union FEDER programme.

C. A. Argyrides and D. K. Pradhan are with the Department of Computer Science at the University of Bristol, Bristol, U.K. (e-mail: costas@computer.org; costas@cs.bris.ac.uk; pradhan@cs.bris.ac.uk).

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

Digital Object Identifier 10.1109/TNS.2010.2043265

Another challenge in the memory design is that MCUs [5] can be induced by direct ionization or nuclear recoil after the passage of a high-energy ion [6]. Another important issue is that the experiments in memories under protons and heavy ion fluxes in [7], [8] show that the probability of having multiple errors grows when the size of memory is increased.

Unfortunately, the solutions of packaging and shielding cannot effectively be used to shield against SEUs and MCUs since they may be caused by neutrons which can easily penetrate through packages [3], [9].

The use of interleaving in the physical arrangement of the memory cells has been the most common approach to deal with multiple errors. This approach has been used, so that cells that belong to the same logical word are separated. If the errors in an MCU are physically close they will cause single errors in different words that can be corrected by the SEC-DED codes as discussed in [10].

However, interleaving cannot be used, for example, in small memories or register files, and in other cases, its use may have an impact on floor-planning, access time and power consumption, as discussed in [11]. For those reasons, the use of more sophisticated codes or the combination of different codes has been proposed in [12] to deal with MCUs when the use of interleaving is not a valid option [13]. More recently, codes that can correct multiple errors only when they are adjacent have also been proposed [11]. These codes are tailored to the specific patterns of errors in an MCU (again, the errors will tend to be physically close) and therefore can achieve effective protection at a reduced cost.

An alternative approach to protect memories is the use of built in current sensors (BICS) that are able to detect the occurrence of errors by detecting changes in the current, as proposed in [14] and [15]. The sensors are placed in the columns of the memory block and they detect unexpected current variations on each of the memory bit positions.

The protection can be optimized with error correcting codes that are tailored to the specific problem of a memory that suffers MCUs [16]. This is the objective of this paper, in which such codes are proposed.

The article is organized as follows. The related work covering techniques to cope with MCUs is presented in Section II. In Section III, the proposed codes are introduced, together with their correction capabilities. Then, in Section IV, the reliability and the cost analysis of the proposed codes are studied in detail and compared with the traditional codes. Finally, the conclusions of the work are presented in Section V.

II. Related Work

In [14], [17] authors proposed the BICS in conjunction with codes to deal with MCUs in memories, as an alternative to the traditional combination of interleaving, parity and SEC-DED

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	C_1	C_2
X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	C_3	C_4
X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	C_5	C_6
X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	X_{32}	C_7	C_8
D_1	D_3	D_5	D_7	D_9	D_{11}	D_{13}	D_{15}		
D_2	D_4	D_6	D_8	D_{10}	D_{12}	D_{14}	D_{16}		

Fig. 1. Logical organization of a 32-bit word.

codes. This new application of BICS would be useful in situations where the use of interleaving is not possible or appropriate, as discussed before. The combination of BICS with error protection codes can correct one or two errors per word. In particular, the codes can detect that there has been an error in the words, since there are only one or two errors per-word in the worst case, for parity and SEC-DED respectively. Using the additional information from the BICS about the faulty columns, the errors could be located, and combining that information with the one from the codes, the errors could be corrected in most cases.

Most recently in [18], authors combine vertical BICS with multiple parity bits per word to guarantee the correction of multiple errors caused by an MCU (bounded). The number of extra parities depends on the maximum distance between errors in an MCU as shown in [10], [19] and [20].

Another way to cope with MCU was reported in [21]. In this paper, authors avoid the use of BICS and combine the SEC-DED and parity codes to protect SRAM memories against MCUs. In this work, they guarantee the correction of any double errors even if the errors are not bounded. A new approach that combines the work proposed in [21] and [18] is presented in this paper. Authors in [21] arrange the codeword into a matrix format using horizontal SEC-DED and vertical parities, and in [18], authors use horizontal parities at a given distance L in conjunction with vertical BICS.

In this paper, we propose to arrange the codeword in matrix format, as in [21], and use L-distance parities horizontally and vertically.

III. Proposed Technique

The traditional assumptions when dealing with MCUs in memories, is that it can be assumed that the maximum distance between errors in an MCU is bounded by a certain value, as shown in [10], [19] and [20]. In [18], authors considered only the maximum horizontal distance and they added BICS vertically. With this technique, all single errors and multiple errors with distance equal or less than L can be corrected, where L-1 is the maximum horizontal distance between the errors in any MCU. Given this assumption, all errors in a given word will occur in a group of up to L consecutive columns.

In the proposed technique, we arrange the bits in a matrix way and we add parity bits horizontally and vertically. The logical organization of the bits is illustrated in Fig. 1. By rearranging the word into a matrix way and following the technique from [18], we can correct up to four distance-1 or adjacent errors.

Where C_i and D_j are parity bits

$$C_1 = X_1 \oplus X_3 \oplus X_5 \oplus X_7 \tag{1}$$
$$C_2 = X_2 \oplus X_4 \oplus X_6 \oplus X_8 \tag{2}$$

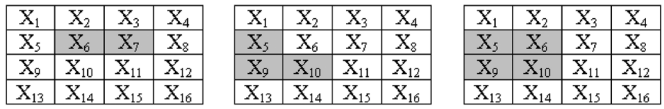

Fig. 2. Example of patterns used in the distance-1 faults.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	C_1	C_2
X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	C_3	C_4
X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	C_5	C_6
X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	X_{32}	C_7	C_8
X_{33}	X_{34}	X_{35}	X_{36}	X_{37}	X_{38}	X_{39}	X_{40}	C_9	C_{10}
X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	X_{46}	X_{47}	X_{48}	C_{11}	C_{12}
X_{49}	X_{50}	X_{51}	X_{52}	X_{53}	X_{54}	X_{55}	X_{56}	C_{13}	C_{14}
X_{57}	X_{58}	X_{59}	X_{60}	X_{61}	X_{62}	X_{63}	X_{64}	C_{15}	C_{16}
D_1	D_3	D_5	D_7	D_9	D_{11}	D_{13}	D_{15}		
D_2	D_4	D_6	D_8	D_{10}	D_{12}	D_{14}	D_{16}		

Fig. 3. Logical organization of 64 bits for distance-1 faults.

and similar for the rest of the other C_i

$$D_1 = X_1 \oplus X_{17} \tag{3}$$
$$D_2 = X_9 \oplus X_{25} \tag{4}$$

and similar for the rest of the other D_j.

Those parity checks enable the correction of multiple errors that are at a distance of one (adjacent), as those shown in Fig. 2.

Note that the proposed technique may be extended for higher number of bits, like 64, to reduce the percentage of redundant bits, as shown in Fig. 3.

We can also improve the protection against larger MCUs by setting the word as an 8x8 matrix with 4 parity bits per row and column (4). Such parity bits, for example, for the first row, are

$$C_1 = X_1 \oplus X_5 \tag{5}$$
$$C_2 = X_2 \oplus X_6 \tag{6}$$
$$C_3 = X_3 \oplus X_7 \tag{7}$$
$$C_4 = X_4 \oplus X_8 \tag{8}$$

and for the first column

$$D_1 = X_1 \oplus X_{33} \tag{9}$$
$$D_2 = X_9 \oplus X_{41} \tag{10}$$
$$D_3 = X_{17} \oplus X_{49} \tag{11}$$
$$D_4 = X_{25} \oplus X_{57}. \tag{12}$$

This approach is shown in Fig. 4 and can correct MCUs whose errors are clustered up to a distance of three. Some patterns of such MCUs are illustrated in Fig. 5.

IV. Reliability and Cost Analysis

For the estimation of the error detection/correction capabilities of the proposed technique and the previous one, we have used a fault injection method. Fault injection is one of the key methods to estimate the error detection/correction capabilities of the circuits which utilize error detection and correction codes [21]. Using a fault injection method, the coverage of the proposed technique was estimated for two scenarios: independent

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	C_1	C_2	C_3	C_4
X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	C_5	C_6	C_7	C_8
X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	X_{24}	C_9	C_{10}	C_{11}	C_{12}
X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	X_{32}	C_{13}	C_{14}	C_{15}	C_{16}
X_{33}	X_{34}	X_{35}	X_{36}	X_{37}	X_{38}	X_{39}	X_{40}	C_{17}	C_{18}	C_{19}	C_{20}
X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	X_{46}	X_{47}	X_{48}	C_{21}	C_{22}	C_{23}	C_{24}
X_{49}	X_{50}	X_{51}	X_{52}	X_{53}	X_{54}	X_{55}	X_{56}	C_{25}	C_{26}	C_{27}	C_{28}
X_{57}	X_{58}	X_{59}	X_{60}	X_{61}	X_{62}	X_{63}	X_{64}	C_{29}	C_{30}	C_{31}	C_{32}
D_1	D_5	D_9	D_{13}	D_{17}	D_{21}	D_{25}	D_{29}				
D_2	D_6	D_{10}	D_{14}	D_{18}	D_{22}	D_{26}	D_{30}				
D_3	D_7	D_{11}	D_{15}	D_{19}	D_{23}	D_{27}	D_{31}				
D_4	D_8	D_{12}	D_{16}	D_{20}	D_{24}	D_{28}	D_{32}				

Fig. 4. Logical organization of 64 bits for distance-3 faults.

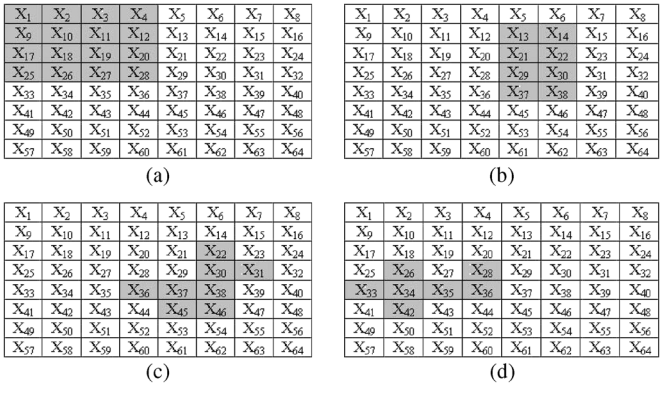

Fig. 5. Example of patterns used in the distance-3 faults.

TABLE I
CORRECTION COVERAGE, USING RANDOM FAULTS (32-BITS)

Corrupt bits per word	RMC	HC	Proposed as in (Fig. 1)
1	100%	100%	100%
2	100%	0%	87.60%
3	100%	0%	65.80%
4	0%	0%	41.60%

errors and clustered errors. The second one is closer to the errors caused by real MCUs.

A. Fault Injection Experiments

Without loss of generality, we have considered the coverage of the proposed technique for a 32-bit data word, since the protection code can be applied on each data word of a given memory. Both independent and clustered errors were injected. For each number of errors, we have used random multiple fault injection and about 10,000 experiments for each case were conducted. The obtained values are portrayed in Table I for Reed Muller Coder (RMC) [22] and Hamming Codes (HC) [23]. For each protection method, we illustrate the protection coverage. The first column shows the number of faulty bits in a word. We also assume two different word sizes of 16 and 64 bits.

The errors caused by an MCU are physically close, as discussed in [10]. We have run a second set of fault injection experiments based on this assumption. In the next fault injection experiment, we have injected up to four errors with distance $\mathrm{L} = 1$ between the errors and reported the correction coverage of the proposed technique. As we can observe from this table, all errors were corrected and the correction coverage of the proposed technique is better than HC, and even better than RMC. Examples of the patterns used for this analysis are illustrated in Fig. 2.

TABLE II
CORRECTION COVERAGE, USING DISTANCE-1 FAULTS

Corrupt bits per word	RMC	HC	Proposed
1	100%	100%	100%
2	100%	0%	100%
3	100%	0%	100%
4	0%	0%	100%
5	0%	0%	0%

These results show how the proposed technique provides single error correction as HC, but can also provide effective protection against errors that are physically close exceeding in this case the performance of more sophisticated codes like RMC.

B. Reliability Analysis

In order to analyze the capability of fault detection and correction of the mentioned protection codes, it is required to use the values in Tables I and II with their corresponding probabilities. For this purpose, we assume the following statements which were also assumed in [21].

1) Transient faults occur with a Poisson distribution.
2) Bit failures are independent for the first type of experiments and clustered with distance 1 for the second type.

In Fig. 6 and Fig. 7, we can see how the reliability improves using our technique in the scenario that the errors caused by an MCU are physically closed. In Fig. 6, the reliability of a 32-bit word protected with HC, RMC and the proposed technique is illustrated. In Fig. 7, the memory reliability while protected using the proposed technique, RMC and HC, is also illustrated. In Fig. 7, we assume that the fault rate is $\lambda = 10^{-5}$, and we can see that our technique improves memory reliability compared to the RMC technique by 2X and the reliability compared to HC by more than 40X. Examples of the patterns used for this analysis are shown in Fig. 2.

Tables III, IV, and V portray the mean time to failure (MTTF) of the proposed technique, RMC and HC codes for the 1 MBit memory in different fault rates in the scenario of distance-1 faults for 16 to 64 bits codewords. In these tables, we can see that the proposed technique improves the memory MTTF by more than 38X compared to HC and more than 2.2X compared to RMC.

In Fig. 8 and Fig. 9, we can see how the reliability improves using our technique in the scenario that the errors caused by an MCU are up to distance-3. In Fig. 8, the reliability of a 64-bit word protected with HC, RMC and the proposed technique is illustrated. In Fig. 9, the memory reliability while protected using the proposed technique, RMC and HC is also illustrated. In Fig. 9, we assume that the fault rate is $\lambda = 10^{-5}$, and we can see that our technique improves memory reliability compared to the RMC technique by almost 40X, and also improves reliability compared to HC by more than 300X. Examples of patterns used for this analysis are shown in Fig. 5.

For the scenario of distance-3 faults and different faults rates for 16 to 64-bit codewords, we have portrayed the

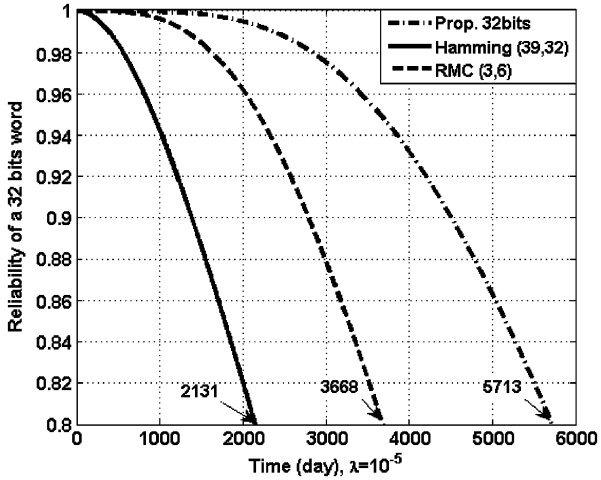

Fig. 6. Reliability (probability of not failing) of a single word, $\lambda = 10^{-5}$ (distance-1 faults).

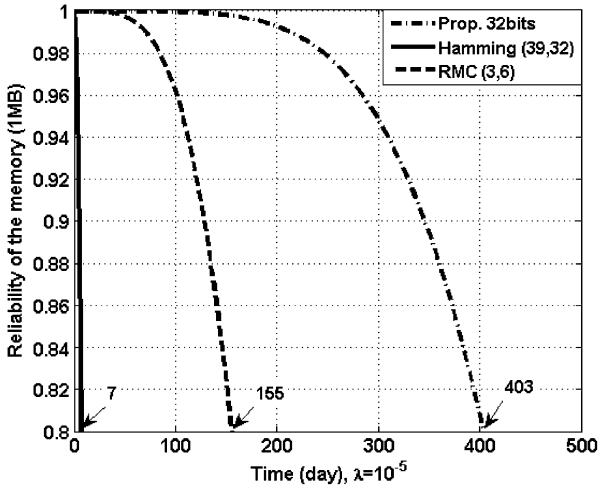

Fig. 7. Reliability (probability of not failing) of the 1 Mbit memory, $\lambda = 10^{-5}$ (distance-1 faults).

TABLE III
MTTF IN DAYS FOR DISTANCE-1 FAULTS (16 bits)

Fault Rate	Proposed 16 bits	HC (22,16)	Improv. over HC	RMC (2,5)	Improv. over RMC
$\lambda=10^{-4}$	78.93	1.67	47.37X	35.50	2.22X
$\lambda=10^{-5}$	789.34	16.66	47.37X	354.95	2.22X
$\lambda=10^{-6}$	7893.40	166.62	47.37X	3549.50	2.22X

TABLE IV
MTTF IN DAYS FOR DISTANCE-1 FAULTS (32 bits)

Fault Rate	Proposed 32 bits	HC (39,32)	Improv. over HC	RMC (3,6)	Improv. over RMC
$\lambda=10^{-4}$	50.68	1.33	38.19X	20.70	2.45X
$\lambda=10^{-5}$	506.77	13.27	38.19X	207.03	2.45X
$\lambda=10^{-6}$	5067.70	132.70	38.19X	2070.30	2.45X

MTTF of the proposed technique, RMC and HC codes in Table VI to Table VIII for a memory size of 1 MBit. The proposed technique can improve the MTTF of the memory by more than 31X compared to HC and more than 675X compared to RMC.

TABLE V
MTTF IN DAYS FOR DISTANCE-1 FAULTS (64 bits)

Fault Rate	Proposed 64 bits	HC. (72,64)	Improv. over HC	RMC (4,7)	Improv. over RMC
$\lambda=10^{-4}$	29.13	0.74	39.49X	10.25	2.84X
$\lambda=10^{-5}$	291.27	7.38	39.49X	102.53	2.84X
$\lambda=10^{-6}$	2912.70	73.76	39.49X	1025.30	2.84X

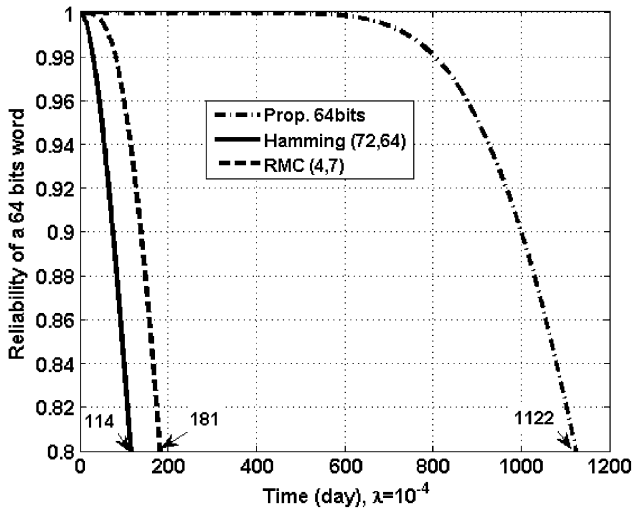

Fig. 8. Reliability (probability of not failing) of the word, $\lambda = 10^{-5}$ (distance-3 faults).

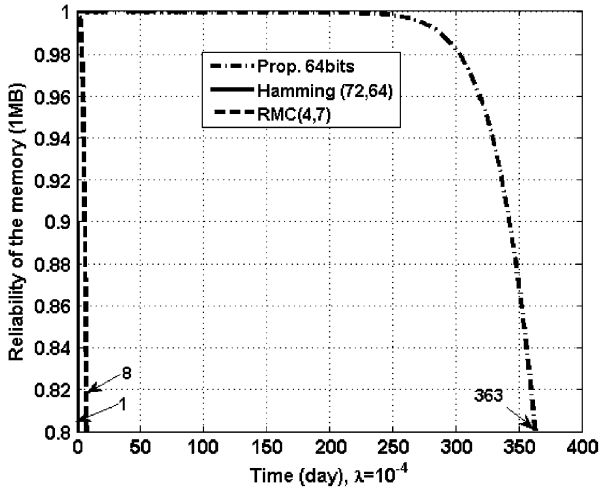

Fig. 9. Reliability (probability of not failing) of the 1 Mbit memory, $\lambda = 10^{-5}$ (distance-3 faults).

C. Cost of the Technique

In Table IX and Table X we can see the number of bits required to implement this coding technique for protection against distance-1 and distance-3 faults respectively. The number of bits required to implement classic coding techniques is also portrayed. From this table, we can see that HC has the least overhead as well as the worst coverage, reliability, and MTTF performance, as shown in Table I to Table VIII and Fig. 6 to Fig. 9.

TABLE VI
MTTF IN DAYS FOR DISTANCE-3 FAULTS (16 bits)

Fault Rate	Proposed 16 bits	HC (22,16)	Improv. over HC	RMC (2,5)	Improv. over RMC
$\lambda=10^{-4}$	1124.90	1.67	675.13X	35.50	31.69X
$\lambda=10^{-5}$	11249.00	16.66	675.12X	354.95	31.69X
$\lambda=10^{-6}$	112490.00	166.62	675.12X	3549.50	31.69X

TABLE VII
MTTF IN DAYS FOR DISTANCE-3 FAULTS (32 bits)

Fault Rate	Proposed 32 bits	HC (39,32)	Improv. over HC	RMC (3,6)	Improv. over RMC
$\lambda=10^{-4}$	655.69	1.33	494.11X	20.70	31.67X
$\lambda=10^{-5}$	6556.90	13.27	494.13X	207.03	31.67X
$\lambda=10^{-6}$	65569.00	132.70	494.13X	2070.30	31.67X

TABLE VIII
MTTF IN DAYS FOR DISTANCE-3 FAULTS (64 bits)

Fault Rate	Proposed 64 bits	HC (72,64)	Improv. over HC	RMC (4,7)	Improv. over RMC
$\lambda=10^{-4}$	392.69	0.74	532.39X	10.25	38.30X
$\lambda=10^{-5}$	3926.90	7.38	532.42X	102.53	38.30X
$\lambda=10^{-6}$	39269.00	73.76	532.42X	1025.30	38.30X

TABLE IX
COST OF THE TECHNIQUE (EXTRA BITS) FOR $L = 1$

N	Proposed	HC	Matrix [21]	RMC[22]
16	16	6	20	16
32	24	7	28	32 (22*)
64	32	8	48	64 (29*)

*Based on shortened version of Reed Muller coding.

TABLE X
COST OF THE TECHNIQUE (EXTRA BITS) FOR $L = 3$

N	Proposed	HC	Matrix [21]	RMC [22]
16	32	6	20	16
32	48	7	28	32 (22*)
64	64	8	48	64 (29*)

*Based on shortened version of Reed Muller coding.

The shortened version of RMC requires slightly fewer redundant bits but it has more complicated decoding circuitry [22] compared to the decoding circuitry of simple parity codes. Finally, the matrix coding requires more redundant bits than the proposed technique. Therefore, the proposed codes are an interesting option to protect memories from clustered MCUs when interleaving cannot be used, as they achieve larger MTTFs with reduced cost. Note that the other techniques are not modified for distance $L = 3$ faults and thus we have the same cost.

We have also used a metric to assess the proposed technique and calculate its efficiency compared to the one of the Hamming Codes and the Reed Muller codes. We refer to it as "MTTF per Cost (MpC)," and it is calculated as in (13)

$$\mathrm{MpC} = \mathrm{MTTF}/\mathrm{Cost} \tag{13}$$

TABLE XI
MTTF PER COST FOR $L = 1$

N	Proposed	HC	Improv.	RMC	Improv.
16	49.33	2.78	17.76X	22.18	2.22X
32	21.12	1.90	11.14X	6.47 (9.41)	3.26X (2.24X*)
64	9.10	0.92	9.87X	1.60 (3.54)	5.68X (2.57X*)

*Based on shortened version of Reed Muller coding.

TABLE XII
MTTF PER COST FOR $L = 3$

N	Proposed	HC	Improv.	RMC	Improv.
16	703.06	2.78	253.17X	22.18	31.69
32	273.20	1.90	144.12X	6.47 (9.41)	42.23X (29.03X*)
64	122.72	0.92	133.11X	1.60 (3.54)	76.60X (34.71X*)

*Based on shortened version of Reed Muller coding.

where cost is the number of extra bits required to implement the technique.

In Table XI and Table XII, the results of the MpC metric for distance-1 and distance-3 faults respectively are portrayed. In Table XI, we can see the proposed technique is more efficient compared to HC by more than 17X at the 16-bit codeword scenario, and more than 2.22X compared to the RMC one. For the 32-bit codeword, we have an improvement of more than 11X, 3X and 2X compared to HC, RMC and the shortened version of RMC respectively. For the 64-bit codeword memory, the improvement compared to the HC is reduced, but it is still higher at more than 9.5X and for the RMC and shortened version of the RMC is more than 5X and more than 2.5X respectively.

The improvement of the efficiency of the proposed technique for distance-3 faults is portrayed in Table XII. The efficiency is improved by more than 31X compared to RMC and more than 250X compared to HC for a 16-bit codeword. Additionally, the efficiency is improved by more than 144X compared to HC at the 32-bit codeword, and also improves the efficiency of the RMC and the shortened RMC by 42X and 29X respectively. The MpC improvement of the 64-bit codeword compared to the HC is reduced, but it is still high at more than 122X. In the case of RMC, the improvement is 76X (and 34X for the shortened version). The increase factor in this table (Table XII) is larger than in the previous one (Table XI) because the classical techniques fail to protect against large MCUs.

V. CONCLUSIONS

In this paper, we have proposed a novel technique to cope with clustered errors caused by MCUs. We have shown that our technique improves the reliability and the MTTF of the memory by more than 675X compared to the traditional HC technique and more than 38X compared to RMC, when clustered distance-3 MCUs are considered. When distance-1 MCUs are considered, we have improved the MTTF of the memory by more than 47X compared to HC, and more than 2.8X compared to RMC. The cost of the technique is lower than the traditional Reed Muller

codes when used to protect against distance-1 faults, and equal when used for distance-3 faults. We did not compare our technique with the well-known Reed Solomon (RS) because the cost for incorporating such technique in memories is huge due to its complex decoding algorithm.

The results of the proposed technique for the MpC metric reveal that we gain an improvement of more than 76X compared to RMC and more than 34X compared to the shortened version of RMC. We have improved the MpC compared to HC by more than 250X. At the same time, the proposed technique can also correct all single errors and therefore can provide effective protection against SEUs and MCUs.

REFERENCES

[1] International Technology Roadmap for Semiconductors [Online]. Available: http://www.itrs.net/ (last access on July 2009).

[2] J. Barth, C. Dyer, and E. Stassinopoulos, "Space, atmospheric and terrestrial radiation environments," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 3, pp. 466–482, Jun. 2003.

[3] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," *IEEE Trans. Nucl. Sci.*, vol. 47, no. 6, pp. 2586–2594, Dec. 2000.

[4] P. Ferreyra, C. Marques, R. Ferreyra, and J. Gaspar, "Failure map functions and accelerated mean time to failure tests: New approaches for improving the reliability estimation in systems exposed to single event upsets," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 1, pp. 494–500, Feb. 2005.

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

[6] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, and R. Reis, "Analyzing area and performance penalty of protecting different digital modules with hamming code and triple modular redundancy," in *Proc. 15th Symp. on Integrated Circuits and Systems Design*, 2002, pp. 95–100.

[7] J. Karlsson, P. Liden, P. Dahlgren, R. Johansson, and U. Gunneflo, "Using heavy-ion radiation to validate fault-handling mechanisms," *IEEE Micro.*, vol. 14, no. 1, pp. 8–11, 1994, 13–23.

[8] R. Reed, M. Carts, P. Marshall, C. Marshall, O. Musseau, P. McNulty, D. Roth, S. Buchner, J. Melinger, and T. Corbiere, "Heavy ion and proton-induced single event multiple upset," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 6, pp. 2224–2229, Dec. 1997.

[9] N. Seifert, D. Moyer, N. Leland, and R. Hokinson, "Historical trend in alpha-particle induced soft error rates of the alpha microprocessor," in *Proc. IEEE 39th Annual Int. Reliability Physics Symp.*, 2001, pp. 259–265.

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

[11] A. Dutta and N. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in *Proc. 25th IEEE VLSI Test Symp.*, May 2007, pp. 349–354.

[12] G. Neuberger, F. de Lima, L. Carro, and R. Reis, "A multiple bit upset tolerant SRAM memory," *ACM Trans. Design Automation Electron. Syst.*, vol. 8, no. 4, pp. 577–590, 2003.

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

[14] F. L. Vargas, M. Nicolaidis, and B. Courtois, "Quiescent current monitoring to improve the reliability of electronic systems in space radiation environments," in *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, 1993, pp. 596–600.

[15] J.-C. Lo, "Analysis of a BICS-only concurrent error detection method," *IEEE Trans. Comput.*, vol. 51, no. 3, pp. 241–253, Mar. 2002.

[16] D. Mavis, P. Eaton, M. Sibley, R. Lacoe, E. Smith, and K. Avery, "Multiple bit upsets and error mitigation in ultra-deep submicron SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 6, pp. 3288–3294, Dec. 2008.

[17] B. Gill, M. Nicolaidis, and C. Papachristou, "Radiation induced single-word multiple-bit upsets correction in SRAM," in *Proc. IEEE Int. On-Line Testing Symp.*, 2005, pp. 266–271.

[18] P. Reviriego and J. A. Maestro, "Efficient error detection codes for multiple-bit upset correction in SRAMs with BICS," *ACM Trans. Design Auto. Electron. Syst.*, vol. 14, no. 1, pp. 1–10, 2009.

[19] 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.

[20] 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.

[21] C. Argyrides, H. R. Zarandi, and D. K. Pradhan, "Matrix codes: Multiple bit upsets tolerant method for SRAM memories," in *Proc. IEEE Int. Symp. on Defect and Fault-Tolerance in VLSI Systems*, 2007, pp. 340–348.

[22] C. Argyrides, S. Loizidou, and D. K. Pradhan, "Area reliability trade-off in improved Reed Muller coding," in *Proc. 8th Int. Workshop on Embedded Computer Systems*, Jul. 2008, pp. 116–125.

[23] R. W. Hamming, "Error detecting and error correcting codes," *The Bell Syst. Tech. J.*, vol. 26, no. 2, pp. 147–160, 1950.