An Efficient, Low-Cost ECC Approach for Critical-Application Memories

Felipe Silva
LESC-DETI- Federal University of Ceará (UFC)
Fortaleza, Brazil
gaspar@lesc.ufc.br

Walter Freitas
GTEL-DETI-Federal University of Ceará (UFC)
Fortaleza, Brazil
walter@gtel.ufc.br

Jarbas Silveira
LESC-DETI- Federal University of Ceará (UFC)
Fortaleza, Brazil
jarbas@lesc.ufc.br

Otávio Lima
LSD-Federal Institut of Ceará
Maracanaú, Brazil
otavio@ifce.edu.br

Fabian Vargas
Faculty of Engineer- Pontifical Catholic University of Rio Grande do Sul (PUC-RS)
Porto Alegre, Brazil
vargas@pucrs.br

César Marcon
Faculty of Computer Science- Pontifical Catholic University of Rio Grande do Sul (PUC-RS)
Porto Alegre, Brazil
cesar.marcon@pucrs.br

ABSTRACT

Multiple Cell Upsets (MCUs) induced by ionizing radiation in memories are becoming more likely to happen due to the continuous technology scaling down. Error Correction Codes (ECCs) are applied for recovering the stored information into its original state providing reliable computer systems. Several ECC are able to deal with MCUs, however, the higher the robustness of an ECC, more area, and energy is required for its implementation, becoming a problem if applied in application where resources are scarce. This article presents the implementation and evaluation of the Matrix Region Section Code (MRSC), a new algorithm for the detection and correction of multiple transient faults in volatile memories with low cost implementation. The experimental results measuring error coverage composed by detection and correction analysis, area, power and delay overheads have shown that MRSC is an excellent option to counteract with MCUs.

CCS CONCEPTS

• Computer systems organization → Reliability • Computer systems organization → Processors and memory architectures

KEYWORDS

Error Correction Codes (ECCs), Multiple Cell Upsets (MCUs), Volatile Memory, Fault Tolerant Systems.

ACM Reference format:

F. Silva, W. Freitas, J. Silveira, O. Lima, F. Vargas and C. Marcon. 2017. An Efficient, Low-Cost ECC Approach for Critical-Application Memories. *In Proceedings of SBCCI '17, Fortaleza -Ceará , Brazil, August 28-September 01, 2017, 6 pages.*
http://dx.doi.org/10.1145/3109984.3110001

1 INTRODUCTION

The increasing technology scaling down in microelectronics enabled the development of high performance Integrated Circuits (IC) allowing implementing complex systems in a tiny circuit area. High operation frequencies and low operating voltage levels characterize these circuits. Besides, these circuits became more susceptible to transient errors due to ionizing radiation coming from high-energy particles, such as alpha particles and neutrons [1]. Soft errors can be caused by the incident radiation in memory devices, changing the stored content and taunting failures in all system. For a space-like critical application, the occurrence of these faults can provoke disastrous consequences [2].

Single Error Correction, Double Error Detection (SEC-DED) codes are heavily used to avoid data corruption caused by soft errors [3][4], whereby their use keeps the memories protected against single errors. However, with the evolution and improvement of scaling technology for IC, the probability of Multiple Cell Upsets (MCUs) increased dramatically [5][6]. When a highly charged particle strikes on memory, more cells are affected, may cause MCUs [7].

MCUs may be driven by many sources as direct ionization or nuclear recoil, caused by the passage of a high-energy ion [8]. It is showed in [1] that packing and shielding are inefficient to prevent MCUs in electronic devices. Moreover, for high-density memories, this probability even increases [9]. The interleaving of memory cells is one of the most effective solutions that allows correcting MCUs using simple SEC-DED codes since the multiple cells affected are generally in adjacent positions. Current sensors are also applied to detect transient faults in memory cells [10]. They detect such faults by monitoring unexpected variations in the power-supply bus of memory blocks [11][12]. This technique can also be improved by the utilization of a more powerful ECC.

Hamming is the first code for a series of ECCs that can correct and detect one error. The Hamming code can be extended to detect double errors; however, without affecting its correction capability. This code is a type of SEC-DED. Standard SEC-DED codes are not able to handle MCUs, requiring more efficacious codes for correcting various MCU patterns. Simultaneously, it is expected that these codes have area and energy efficient implementation and high error correction rates for protecting memory arrays.

Argyrides et al. [13] proposed the Matrix approach that codifies and organizes bits in a matrix format to correct two errors in a 32-bit word. The code presented is composed of four lines of codified bits by Hamming, and an additional line with parity bits is used for detecting adjacent bit errors in the same line. The Matrix decoding consists of syndrome verification and parity bits. Experimental results described by Sanchez-Macitan et al. [14] show that codes with matrix structure are not effective for aggressive patterns of MCUs. Besides, this code has low correction efficiency when compared to a more robust code such as Reed-Muller, but consuming less area and energy.

The Reed-Muller code is a family of ECCs with high complexity and robustness, which is vastly used in critical systems. Reed-Muller (2,5) is an error correction code used for correcting MCUs in flash-based FPGAs [15], where 2 is the order of the code, and 5 is the code length of 2^5.

Argyrides et al. [16] compared Reed-Muller (2, 5) and Matrix regarding error coverage, area, energy consumption, and delay. They concluded that Reed-Muller (2, 5) presents better error coverage than Matrix. Nonetheless, Reed-Muller's family code is more effective than the Matrix Code to deal with MCUs. However, depending on the code format, it demands more complex coding and decoding circuits than Matrix and, therefore, consuming more area, energy, and delay.

Column Line Code (CLC) [17] was developed with the purpose of achieve high rates of correction and detection for aggressive patterns of MCUs, achieving an intermediary cost between Matrix and Reed-Muller (2,5). CLC utilizes Extended Hamming and Parity in a matrix format similar to Matrix code [13]. However, CLC uses more redundancy bits and a higher developed algorithm for correcting and detecting complex MCU patterns.

However, results presented in [17] showed that CLC is more likely to be applied in systems where error coverage has bigger importance than implementation cost, since Matrix still presented itself as a better for option double error scenarios. It is our understanding that an ECC with a cost (area, power and delay) near to Matrix Code and similar error coverage of CLC would be a perfect match for critical applications. In this scenario, this paper presents the Matrix Region Selection Code (MRSC), an ECC with low-cost and robust error coverage capability to fill such gap between Matrix and CLC.

The remaining of this paper is organized as follows. Section 2 describes the MRSC approach, its encoder/decoder structures, and strategies used for MCU detection and correction. Section 3 shows the fault-injection experiments implemented to validate the proposed approach. All experiments compare MRSC with Matrix, Reed-Muller (2, 5), and CLC. The final goal of this section is to verify the efficiency of the proposed technique to deal with MCUs. Section 4 discusses the collateral effects yielded from the use of the proposed technique regarding the area, power and delay increase. Finally, Section 5 draws the conclusions of the paper.

2 PROPOSED METHOD

MRSC is an ECC based on two-dimensional (2D) codes, such as [13][17], that aims to correct and detect MCUs in memories. The code presented in this paper codifies 16 data bits in 32 bits. However, only parity bits are used in encoding data bits to reduce the cost of implementation.

2.1 MRSC Encoding Process

Figure 1 shows the structure of 32 bits of data encoded by MRSC. The cells shaded in gray are the data bits, which were divided into four groups (A, B, C, D), each group with four bits.

A_1	A_2	A_3	A_4	Di_1	Di_3	CbA_{13}	CbA_{24}
B_1	B_2	B_3	B_4	Di_2	Di_4	CbB_{13}	CbB_{24}
C_1	C_2	C_3	C_4	P_1	P_3	CbC_{13}	CbC_{24}
D_1	D_2	D_3	D_4	P_2	P_4	CbD_{13}	CbD_{24}

Figure 1: MRSC Encoded data model.

The cells shaded in green are the *Diagonal bits* (Di) calculated with XOR operations ($\oplus$) in specific data bits:

$$Di_1 = A_1 \oplus B_2 \oplus C_1 \oplus D_2 \quad (1)$$

$$Di_2 = A_2 \oplus B_1 \oplus C_2 \oplus D_1 \quad (2)$$

$$Di_3 = A_3 \oplus B_4 \oplus C_3 \oplus D_4 \quad (3)$$

$$Di_4 = A_4 \oplus B_3 \oplus C_4 \oplus D_3 \quad (4)$$

The cells shaded in blue are *Parity bits* (P) calculated with XOR operations in the data bits columns:

$$P_1 = A_1 \oplus B_1 \oplus C_1 \oplus D_1 \quad (5)$$

$$P_2 = A_2 \oplus B_2 \oplus C_2 \oplus D_2 \quad (6)$$

$$P_3 = A_3 \oplus B_3 \oplus C_3 \oplus D_3 \quad (7)$$

$$P_4 = A_4 \oplus B_4 \oplus C_4 \oplus D_4 \quad (8)$$

The cells shaded orange are *Check bits* (Cb) calculated with XOR operations in interleaved bits of each group:

$$CbA_{13} = A_1 \oplus A_3 \quad (9)$$

$$CbA_{24} = A_2 \oplus A_4 \quad (10)$$

$$CbB_{13} = B_1 \oplus B_3 \quad (11)$$

$$CbB_{24} = B_2 \oplus B_4 \quad (12)$$

$$CbC_{13} = C_1 \oplus C_3 \quad (13)$$

$$CbC_{24} = C_2 \oplus C_4 \quad (14)$$

$$CbD_{13} = D_1 \oplus D_3 \quad (15)$$

$$CbD_{24} = D_2 \oplus D_4 \quad (16)$$

After the calculation of the redundancy bits, the encoding process ends and the 32 bits can be stored. Note that the Di bits and P bits are positioned between the data bits and Cb bits, in order to improve the efficiency of MRSC against MCUs characterized by adjacent error patterns. Figure 2 describes the mains elements of the parity operation of the MRSC encoder.

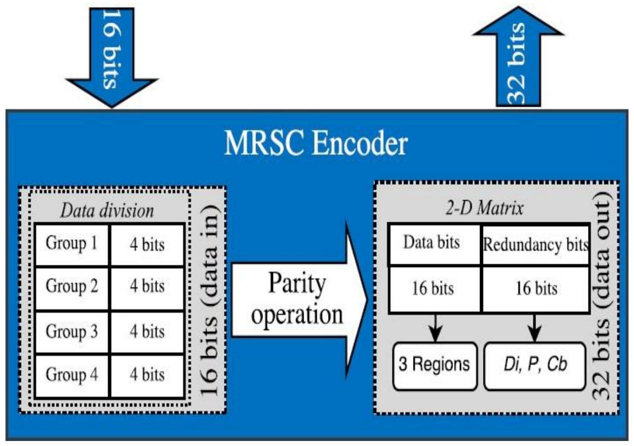


Figure 2: Parity operation of the MRSC encoder.

2.2 MRSC Decoding Process

The decoding process of MRSC is divided into three steps:

Syndrome estimation of the redundancy bits - The syndrome estimation consists of a XOR operation between the redundancy data stored and the recalculated redundancy bits (RDi, RP, and RCb). Therefore, the values for the Syndrome of Diagonal (SDi), Parity (SP) and Check bits (SCb) are estimated by:

$$SDi = Di \oplus RDi \tag{17}$$

$$SP = P \oplus RP \tag{18}$$

$$SCb = Cb \oplus RCb \tag{19}$$

Verification of error decoding conditions - After the calculation of the Syndromes, one of these two conditions need to be satisfied before the error correction execution: (i) both SDi and SP vectors must have at least one value equal to one; (ii) more than one SCb value is equal to one.

These conditions allow the algorithm to detect an error in the data bits region. Their applicability will be explained with more details subsequently.

Selection of the wrong data region and correction process - In this phase of the decoding process, a specific region of the data bits is selected to be corrected. These regions are divided as it shows in Figure 3(a), (b) and (c). The proposal in split the data bits in three regions was elaborated in order to select a specific group of bit to operate the correction process. This approach reduces considerably the area and power cost.

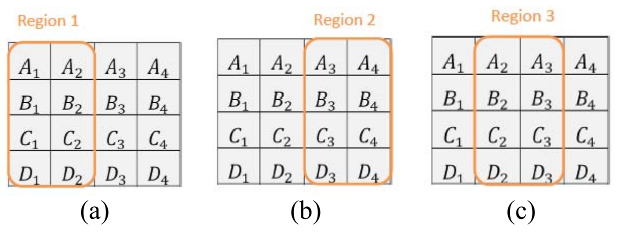


Figure 3: Regions of data bits.

Figure 3 (a), (b) and (c) show that region 1, 2 and 3 are formed by data bits distributed in columns (1 and 2), (3 and 4) and (2 and 3), respectively. The selection of which region will be corrected is defined by the integer sum (+) of specific bits of SDi and SP.

Table 1 presents a group of equations which describes the criterion for region selection of MRSC, where the region with more syndrome bits equals to 1 is be declared as the wrong one (Region 1 or Region 2). If the sum of the equations presents equal value, then the Region 3 is selected.

Table 1: Region selection criterion.

Region selected	Criterion to selection
Region 1	$(SDi_1 + SDi_2 + P_1 + P_2) > (SDi_3 + SDi_4 + P_3 + P_4)$
Region 2	$(SDi_1 + SDi_2 + P_1 + P_2) < (SDi_3 + SDi_4 + P_3 + P_4)$
Region 3	$(SDi_1 + SDi_2 + P_1 + P_2) = (SDi_3 + SDi_4 + P_3 + P_4)$

For regions 1 and 2, the correction procedure consists in a XOR operation between the region selected and the $SCbs$ matrix. Region 3 is a special case where it is strictly necessary that neither of all SDi and SP bits are null, even if the condition II of step 2 is satisfied. Note that Region 3 has its first column formed by values with the even index (2), meaning that the correction performed has to be different from the other regions.

If region 3 is selected, the correction procedure must be performed by $SCbs$ with shifted positions, to align the indexes of $SCbs$ with the matrix of Region 3. In the following section, some correction examples performed by the proposed method are described. Figure 4 summarizes the operation performed by the decoder described in this section.

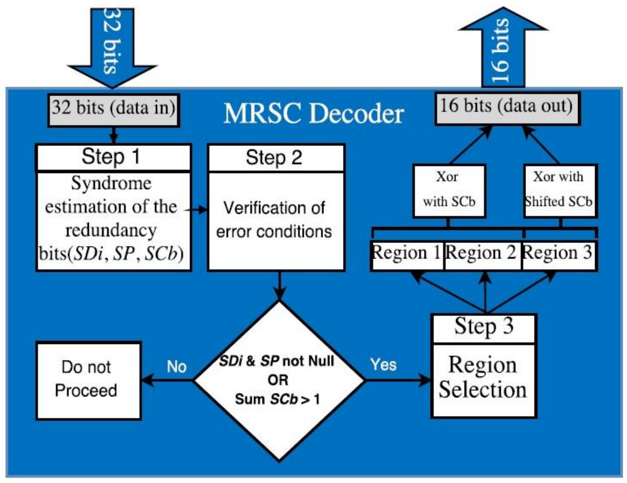


Figure 4: Block diagram of the MRSC decoding algorithm.

2.3 MRSC Correction Examples

In this section are presented some examples that illustrate the running of the MRSC decoder. For these examples was consider a 16-bit data containing the pattern 1000000011111010. Figure 5 shows the data encoded by MRSC.

1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0
1	0	1	0	1	1	0	0

Figure 5: MRSC Encoded data example.

Figure 6 to Figure 9 show situations where MCUs (in red) occurred in some adjacent data bits.

								SCbs	
0	0	0	0	0	1	1	0	$0 \oplus 1 = 1$	$0 \oplus 0 = 0$
1	0	0	0	0	0	0	0	$0 \oplus 1 = 1$	$0 \oplus 0 = 0$
1	1	1	1	1	0	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$
1	0	1	0	1	1	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$

SDi $_{1,2,3,4}$	$0010 \oplus 1110 = 1100$
SP $_{1,2,3,4}$	$1101 \oplus 1101 = 0000$
Region selected	1+1+0+0 > 0+0+0+0

Region 1

Figure 6: MRSC Error example 1.

Figure 6 exemplifies a MCU occurred in the first column. The estimations of SDi, SP, and SCb are showed in adjacent tables. Note that SP values were all equal to 0, which do not satisfies the first verification condition; however, more than one value of SCb

is equal to 1, allowing to proceed to the next step. To determine which region was selected, the analysis of Table 1 is made. Only $SDi_{1,2}$ presented value equal to 1, therefore, Region 1 was selected to be corrected by SCb. A XOR operation is performed in the lines where SCb present value 1.

Figure 7 exemplifies another MCU that occurred in the two last columns of the data bit matrix. For this scenario, both SDi and SP contain at least one value not null. Note that $SDi_{3,4}$ and $SP_{3,4}$ present values equal to 1. The analysis of Table 1 concludes Region 2 of the data bits matrix was selected to be corrected by SCb.

								SCbs	
1	0	1	1	0	1	1	0	$0 \oplus 1 = 1$	$0 \oplus 0 = 1$
0	0	0	0	0	0	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$
1	1	1	1	1	0	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$
1	0	1	0	1	1	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$

***SDi**$_{1,2,3,4}$*	$0010 \oplus 0001 = 0011$	
***SP**$_{1,2,3,4}$*	$1101 \oplus 1110 = 0011$	
Region selected	$0+0+0+0 < 1+1+1+1$	***Region 2***

Figure 7: MRSC Error example 2.

Figure 8 shows an example, where $SDi_{2,4}$ and $SP_{2,3}$ had its values equal to one. According with the expression on Table 1, the Region 3 was selected. In this situation, for the correction procedure, the values of SCb correspondent to each line must be shifted and then, it is made a XOR operation with the data bits matrix selected.

								SCbs	
1	1	0	0	0	1	1	0	$0 \oplus 0 = 0$	$0 \oplus 1 = 1$
0	0	1	0	0	0	0	0	$0 \oplus 1 = 1$	$0 \oplus 0 = 0$
1	1	1	1	1	0	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$
1	0	1	0	1	1	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$

***SDi**$_{1,2,3,4}$*	$0010 \oplus 0111 = 0101$	
***SP**$_{1,2,3,4}$*	$1101 \oplus 1011 = 0110$	
Region selected	$0+1+0+1 = 0+1+1+0$	***Region 3***

Figure 8: MRSC Error example 3.

Figure 9 shows the last example, which the data bits had four cells affected. This case represents a complex pattern of MCU, where not only the data bits region suffered with bit-flip, but also the region with redundancy bits. Note that $SDi_{1,4}$ and SP_3 presents value equal to one, and according with the Table 1, the Region 2 was selected and will be corrected by SCb.

								SCbs	
1	0	1	1	1	1	1	0	$0 \oplus 1 = 1$	$0 \oplus 1 = 1$
0	0	0	1	0	0	0	0	$0 \oplus 0 = 0$	$0 \oplus 1 = 1$
1	1	1	1	1	0	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$
1	0	1	0	1	1	0	0	$0 \oplus 0 = 0$	$0 \oplus 0 = 0$

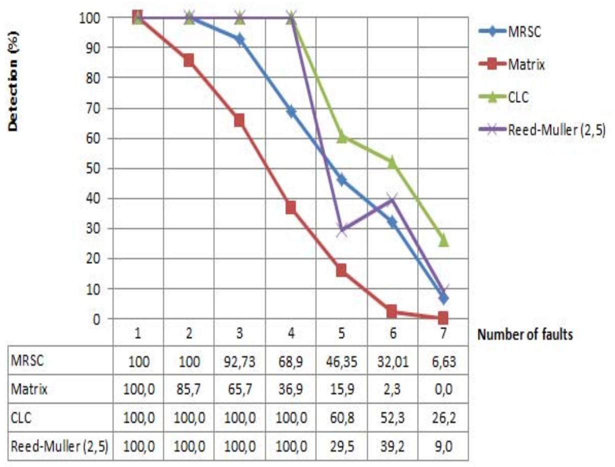
***SDi**$_{1,2,3,4}$*	$0010 \oplus 1011 = 1001$	
***SP**$_{1,2,3,4}$*	$1101 \oplus 1111 = 0010$	
Region selected	$1+0+0+0 < 0+1+1+0$	***Region 2***

Figure 9: MRSC Error example 4.

3 FAILURE INJECTION EXPERIMENTS

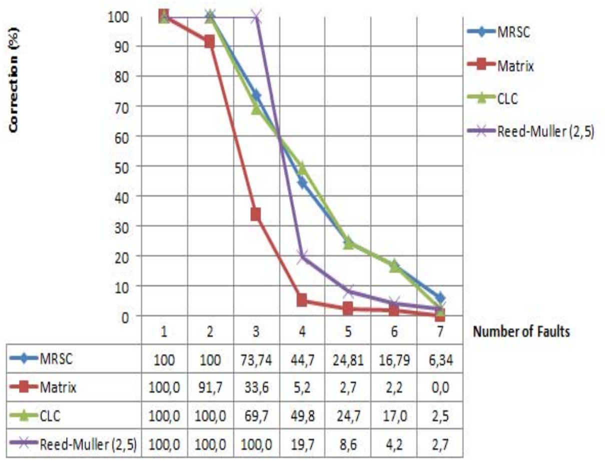
This work validated the proposed technique employing MATLAB. We designed an algorithm with the purpose of inserting failures into random codified words. For this experiment were verify one million words pseudo-randomly generated for seven test scenarios. The testbench inserts one to seven errors in adjacent cells to mimic the structure of MCUs. The faults inserted were considering distance of one for multiple fault cases.

For all experiments, we compare the performance of MRSC with Matrix, CLC, and Reed-Muller (2, 5). The MRSC, Matrix, and Reed-Muller (2, 5) have 32-bit encoded data, whereas CLC have 40-bit, due to the utilization of more parity bits and the Extended Hamming parity bit. Figure 10 and Figure 11 illustrate the experimental results of detection and correction analysis regarding the number of faults.



	1	2	3	4	5	6	7
MRSC	100	100	92,73	68,9	46,35	32,01	6,63
Matrix	100,0	85,7	65,7	36,9	15,9	2,3	0,0
CLC	100,0	100,0	100,0	100,0	60,8	52,3	26,2
Reed-Muller (2,5)	100,0	100,0	100,0	100,0	29,5	39,2	9,0

Figure 10: Detection rate per faulty bits for the analyzed codes.



	1	2	3	4	5	6	7
MRSC	100	100	73,74	44,7	24,81	16,79	6,34
Matrix	100,0	91,7	33,6	5,2	2,7	2,2	0,0
CLC	100,0	100,0	69,7	49,8	24,7	17,0	2,5
Reed-Muller (2,5)	100,0	100,0	100,0	19,7	8,6	4,2	2,7

Figure 11: Correction rate per faulty bits for the analyzed codes.

Figure 10 and Figure 11 demonstrate all methods are 100% effective for a scenario with a single fault. When the number of faults increases, Matrix and Reed-Muller (2, 5) lose efficiency more abruptly for more aggressive fault scenarios. Two reasons can explain both results: (i) Matrix has fewer redundancy bits and it has a simpler algorithm when compared with the other codes; (ii) Reed-Muller (2, 5) has a distance of 8 [15], which mean a sharp correction rate for only three errors.

MRSC achieves better results than Matrix in all fault scenarios, and better than Reed-Muller (2, 5) for aggressive error patterns. Besides, MRSC showed very competitive correction results, especially when compared to CLC. Although CLC depicted very robust results for both analyses, MRSC requires only 16 redundancy bits; whereas CLC needs 24 bits of redundancy, implying significant impact on implementation cost.

4 AREA, POWER AND DELAY ANALYSIS

This section analyzes the results obtained from the area consumption, power dissipation, and delay. We analyze the same codes, which are written in Verilog and synthesized with the Cadence's Encounter RTL Compiler for a 65 nm technology. Table 2 and Table 3 show the area consumption, power dissipation and delay obtained from the synthesis of the encoders and decoders of the four codes, respectively.

Table 2 displays that Reed-Muller (2,5) and MRSC have the highest and the lowest area consumption, power dissipation and delay, respectively. It is important to point out that CLC and Matrix has their codes based on Extended Hamming and Hamming, respectively, along with parity. Meanwhile, MRSC uses only parity and simple logic structures, which represent a major advantage for MRSC by reducing significantly its overall cost in terms of area, power and delay.

Table 2: Synthesis results of the encoders (relative values are normalized according to Reed-Muller (2,5)).

Method	Area		Power		Delay	
	(μm²)	(%)	(mW)	(%)	(ns)	(%)
Reed-Muller (2,5)	504	100.0	0.037	100.0	0.74	100.0
CLC	435	86.3	0.024	64.8	0.35	47.3
Matrix	298	59.1	0.010	27.0	0.15	20.2
MRSC	251	49.8	0.009	24.3	0.14	18.9

Table 3 depicts that Reed-Muller (2, 5) and MRSC have the highest and lowest overall overhead cost for the decoders. Although power dissipation of MRSC being near to 10% bigger than Matrix, its area and delay consumptions are 23% and 16.6% smaller, respectively. CLC has the larger quantity of redundancy bits of all codes analyzed as well as a more complex logic than MRSC and Matrix, which justifies their cost results. Although Reed-Muller (2, 5) presents the same number of redundancy bits of MRSC and Matrix, its majority logic structure is very complex and expensive to be implemented in hardware designs.

Table 3: Synthesis results of the decoders (relative values are normalized according to Reed-Muller (2,5)).

Method	Area		Power		Delay	
	(μm²)	(%)	(mW)	(%)	(ns)	(%)
Reed-Muller (2,5)	4312	100.0	0.737	100.0	2.124	100.0
CLC	1351	31.3	0.076	10.3	1.326	62.4
Matrix	1210	28.1	0.059	8.0	1.264	59.5
MRSC	931	21.6	0.065	8.8	1.055	49.7

It is important to align high error correction rates with low overhead of silicon implementation to choose an efficient ECC for memory devices. The metric applied in [17] was selected to evaluate the relation between error coverage per total cost (Encoder cost and Decoder cost). The expression utilized is described in Equation 20.

$$TCC = \frac{Detection\ rate \times Correction\ rate}{Area\ x\ Power\ x\ Delay} \qquad (20)$$

For each coding method, the parameters of Equation 20 are obtaining with the following rules:

- *Detection rate* and *Correction rate* are the same values depicted in Figure 10 and Figure 11, respectively, which were extracted from the experiments describes in Section 3;
- *Area* and *Power* are achieved summing the corresponding values of modules, the encoder (Table 2) and the decoder (Table 3). We chose this approach because both modules affect the implementation cost jointly;
- *Delay* is achieved considering the greatest value between the encoder and the decoder. We chose this approach because the modules are independent and can operate in parallel.

All these parameters of Equation 20 are normalized dividing their value by the lowest corresponding value; for instance, *Delay* of CLC is 1.110; which was achieved dividing 1.326 ns (delay of the CLC decoder) by 1.195 ns (delay of the MRSC decoder). The range of *TCC* achieved is from 0 to 1; and Figure 12 shows the *TCC* achieved for all methods regarding a range from one to seven faults.

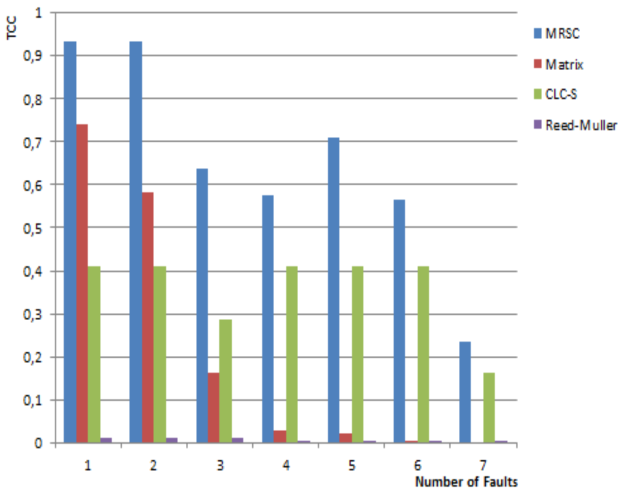


Figure 12: *TCC* for all the methods according to the number of faults.

Figure 12 depicts that MRSC performs better than all other metrics proposed for all fault scenarios due to its lightweight implementation cost and good error coverage. Matrix obtained fine TCC results for one and two faults. From three to seven faults Matrix shows lower results in error correction and detection when compared with the other metrics. Indeed, CLC is mainly focused on aggressive error patterns [17]. The significant cost discrepancy between MRSC and CLC represented a major advantage for the first, since MRSC presented detection and correction rates quite near to CLC, justifying the better results achieved by MRSC. Finally, Reed-Muller (2, 5) achieved the lowest results for TCC in all experiments.

5 CONCLUSIONS

This paper proposes Matrix Region Selector Code (MRSC) an error detection/correction code for memory devices subjected to multiple cell upsets (MCUs). This code is based on parity codes and interleaving to deal with several patterns of MCUs.

Experimental results describing detection and correction analysis indicate that MRSC achieved very competitive results in terms of area, power and delay when compared with CLC, Matrix and Reed-Muller (2, 5). It is important to point out that MRSC requires 33.4% fewer redundancy bits than CLC, which represents a considerable economy in implementation cost (area, power and delay).

MRSC showed up as the lowest cost code of all evaluated codes. The utilization of Hamming and Extended Hamming in the

ECCs Matrix and CLC, respectively, brought advantages in what concern error coverage, as it show in Figure 10 and Figure 11. However, this also increased heavily the cost of both codes when compared with MRSC. Reed-Muller (2, 5) is based on Majority logic, which is an even more complex logic to correct errors, being surpassed in implementation cost analysis by all codes.

Regarding the TCC metric, MRSC presents the best results for all fault scenarios , which means that MRSC has the better tradeoff between error coverage and implementation cost than all codes analyzed.

REFERENCES

[1] P. Hazucha, C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *IEEE Transaction on Nuclear Science*, v. 47, n. 6, pp. 2586-2594, Dec. 2000.

[2] K. LaBel, C. Barnes, C. Marshall, A. Johnston, R. Reed, J. Barth, C. Seidleck, S. Kayali, M. O'Bryan. A roadmap for NASA's radiation effects research in emerging microelectronics and photonics. *IEEE Aerospace Conference*, v. 5, pp. 535-545, 2000.

[3] P. Ferreyra, C. Marques, R. Ferreyra, 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 Transaction on Nuclear Science*, v. 52, n. 1, pp. 494-500, Feb. 2005.

[4] V. Gherman, S. Evain, F. Auzanneau, Y. Bonhomme. Programmable extended SEC-DED codes for memory errors. *IEEE VLSI Test Symposium (VTS)*, pp. 140-145, 2011.

[5] D. Radaelli, H. Puchner, S. Wong, S. Daniel. Investigation of multibit upsets in a 150 nm technology SRAM device. *IEEE Transaction on Nuclear Science*, v. 52, n. 6, pp. 2433-2437, Dec. 2005.

[6] A. Chugg, M. Moutrie, R. Jones. Broadening of the variance of the number of upsets in a read-cycle by MBUs. *IEEE Transactions on Nuclear Science*, v. 51, n. 6, pp. 3701-3707, Dec. 2004.

[7] J. Maestro, P. Reviriego. Study of the effects of MBUs on the reliability of a 150 nm SRAM device. *ACM/IEEE Design Automation Conference (DAC)*, pp. 930-935, 2008.

[8] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, R. Reis, Analyzing area and performance penalty of protecting different digital modules with hamming code and triple modular redundancy. *Symposium on Integrated Circuits and Systems Design*, pp. 95-100, 2002.

[9] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, T. Toba. Impact of Scaling on Neutron-Induced Soft Error in SRAMs From a 250 nm to a 22 nm Design Rule. *IEEE Transactions on Electron Devices*, v. 57, n. 7, pp. 1527-1538, Jul. 2010.

[10] F. Vargas, M. Nicolaidis. SEU-Tolerant SRAM Design Based on Current Monitoring. International Symposium on Fault-Tolerant Computing (FTCS), p. 106-115, 1994.

[11] C.-L. Hsu, M.-H. Ho, C.-F. Lin. Novel Built-In Current-Sensor-Based Testing Scheme for CMOS Integrated Circuits. *IEEE Transactions on Instrumentation and Measurement*, v. 58, n. 7, pp. 2196-2208, Jul. 2009.

[12] D. Toro, M. Arzel, F. Seguin, M. Jezequel. Soft Error Detection and Correction Technique for Radiation Hardening Based on C-element and BICS. *IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 61, n. 12, pp. 952-956, Dec. 2014.

[13] C. Argyrides, H. Zarandi, D. Pradhan. Matrix Codes: Multiple Bit Upsets Tolerant Method for SRAM Memories. *IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT)*, pp. 340-348, 2007.

[14] A. Sanchez-Macian, P. Reviriego, J. Maestro. Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement. *IEEE Transactions on Device and Materials Reliability*, v. 14, n. 1, pp. 574-576, Mar. 2014.

[15] B. Varghese, S. Sreelal, P. Vinod, A. Krishnan. Multiple bit error correction for high data rate aerospace applications. *IEEE Conference on Information Communication Technologies (ICT)*, pp. 1086-1090, 2013.

[16] C. Argyrides, D. Pradhan, T. Kocak. Matrix Codes for Reliable and Cost Efficient Memory Chips. *IEEE Transaction Very Large Scale Integration (VLSI) Systems*, v. 19, n. 3, Mar. 2011.

[17] H. Castro, J. Silveira, A. Coelho, F. Silva, P. Magalhães, O. Lima. A Correction Code for Multiple Cells Upsets in Memory Devices for Space Applications. *14th IEEE International New Circuits and Systems Conference(NEWCAS)*. 2016.

[18] A. Saleh, J. Serrano, J. Patel. Reliability of Scrubbing Recovery Techniques for Memory Systems. *IEEE Transactions on Reliability*, v. 39, n. 1, pp. 114-122, 1990.

[19] S. Miremadi, H. Zarandi. Reliability of Protected Techniques Used in Fault-Tolerant Cache Memories. *IEEE Annual Canadian Conference on Electrical and Computer Engineering*, pp. 776-779, 2005.

[20] Y. Shi, X. Zhang, Z.-C. Ni, N. Ansari, Interleaving for combating bursts of errors. *IEEE Circuits Syst. Mag.,* v. 4, n. 1, pp. 29-42, 2004.