Journal of Electronic Testing (2018) 34:417–433
https://doi.org/10.1007/s10836-018-5738-5

An Extensible Code for Correcting Multiple Cell Upset in Memory Arrays

Felipe Silva[1] · Jardel Silveira[1] · Jarbas Silveira[1] · César Marcon[2] · Fabian Vargas[2] · Otávio Lima Jr[3]

Received: 27 February 2018 / Accepted: 21 June 2018 / Published online: 6 July 2018

Abstract
As the microelectronics technology continuously advances to deep submicron scales, the occurrence of Multiple Cell Upset (MCU) induced by radiation in memory devices becomes more likely to happen. The implementation of a robust Error Correction Code (ECC) is a suitable solution. However, the more complex an ECC, the more delay, area usage and energy consumption. An ECC with an appropriate balance between error coverage and computational cost is essential for applications where fault tolerance is heavily needed, and the energy resources are scarce. This paper describes the conception, implementation, and evaluation of Column-Line-Code (CLC), a novel algorithm for the detection and correction of MCU in memory devices, which combines extended Hamming code and parity bits. Besides, this paper evaluates the variation of the 2D CLC schemes and proposes an additional operation to correct more MCU patterns called extended mode. We compared the implementation cost, reliability level, detection/correction rate and the mean time to failure among the CLC versions and other correction codes, proving the CLCs have high MCU correction efficacy with reduced area, power and delay costs.

Keywords Error correction code (ECC) · MCU · Memories

1 Introduction

The increasing scaling of microelectronics technology boosts the design of high performance integrated circuits characterized by having high operating frequencies and low voltage levels. These circuits are susceptible to transient errors due to ionizing radiation arising from high-energy particles, such as alpha particles and neutrons [8]. Radiation can cause soft errors in memory arrays changing their content and causing errors, which in turn, can be disseminated through the entire system producing severe failures. This concern is even more severe for critical applications such as space systems, where the consequences may be disastrous [13].

Single Error Correction, Double Error Detection (SEC-DED) codes are heavily used to avoid data corruption caused by soft errors [5, 6], whereby their use keeps the memories protected against single errors. As the integration technology scales down, the soft error rate increases and the probability of Multiple Cell Upset (MCU) also increases [4, 18]. MCU may happen, for example, when a highly charged particle strikes on memory, disturbing several cells [16].

Several sources can induce an MCU as direct ionization or nuclear recoil after the passage of a high-energy ion [9]. MCU cannot be avoided by using packaging and shielding solution as proved by [8], and its occurrence increases with the size of the memory device [12]. The interleaving of memory cells belonging to the same word is one of the most used techniques that allows correcting MCUs using simple SEC-DED codes because frequently the affected cells are physically close [2].

Responsible Editor: M. Goessel

✉ Felipe Silva
gaspar@lesc.ufc.br

Jardel Silveira
jardel@lesc.ufc.br

Jarbas Silveira
jarbas@lesc.ufc.br

César Marcon
cesar.marcon@pucrs.br

Fabian Vargas
vargas@computer.org

Otávio Lima, Jr
otavio@ifce.edu.br

[1] Laboratório de Engenharia e Sistemas de Computação (LESC/DETI) /Federal University of Ceará (UFC), Fortaleza, Brazil

[2] Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil

[3] Laboratório de Sistemas Digitais (LSD) /Federal Institute of Ceará (IFCE), Maracanaú, Maracanaú, Brazil

One of the used protection techniques is the addition of built-in current sensors, which detect faults by surveilling unexpected variations in the electrical current that supplies the memory blocks [11, 24]. This technique can be improved by using a robust Error Correction Code (ECC), which is the principal objective of this paper.

Hamming is a base code for a series of ECCs that can correct and detect one error. The Hamming code can be extended to detect double errors, increasing its reliability. Extended Hamming is a type of SEC-DED that adds one parity bit in the codified word for detecting two errors.

MCU cannot be corrected using standard SEC-DED codes, demanding more efficient codes capable of fixing several patterns of upsets on multiple bits. Aligning correction and implementation efficiency is a crucial factor to protect memory devices against MCUs. Therefore, it is expected those codes have area and energy efficient implementation and high error correction rates for protecting memory arrays.

Column-Line-Code (CLC) [3] is an ECC with a 2-Dimensional (2D) format that distributes data and redundancy bits in lines (horizontal rows) and columns (vertical rows). CLC combines Extended Hamming and parity bits homogeneously distributed in the 2D matrix scheme.

A 2D scheme with more redundant bits could correct and detect more error patterns, but with a substantial increase in the implementation cost. For instance, a 16-bit data word requires 24 bits of redundancy using CLC. A lighter 2D scheme reduces the implementation cost in detriment of slightly smaller MCU error coverage.

To improve CLC, we explored some modifications on the 2D scheme used to distribute parity and data bits. As main contribution of this work, we proposed two new versions of CLC; the first one has lower implementation cost targeting lesser demanding applications, the other one presents more redundant bits for more complex applications. We also proposed a new correction step for CLC called extended mode, which enhances the correction of CLC significantly.

Following we present the methodology used in CLC schemes, their algorithms, the encoder and decoder architecture as well as the strategies used for error detection and correction. In sequence, experimental results verify the efficiency of the CLC types for detecting and correcting errors when compared to other codes - Reed-Muller(2,5) [3] and Matrix [26]. We estimate error detection and correction capability, reliability and Mean Time To Failure (MTTF) for the proposed approaches. This work also analyses area and energy consumption of the CLC decoder and encoder implementations. The metric Total Cost per Coverage (*TCC*) presented in [3] was applied to evaluate the tradeoff between error coverage and implementation cost of each code. At the end of this work, we presented the main conclusion.

2 Related Work

Vargas and Nicolaidis [25] described in 1994 the first approach combining current checking with error detection codes in a 2D structure to handle MCUs. They proposed a technique aiming to improve SRAM reliability against Single Event Upsets (SEUs) using the Built-In Current Sensor (BICS). By combining current checking per column with single parity bit per row of an SRAM, the code corrects multiple upsets. From that time on, several other 2D error correction codes to deal with MCUs were proposed with different techniques and effectiveness. This section describes the most relevant related works using 2D structures for detecting and correcting error found in the last 10 years and compares them with our approach.

Reed-Muller is a family of complex and robust ECCs widely used in critical systems. Reed-Muller(2,5) is an ECC used for correcting MCUs in flash-based FPGAs [26], where "2" and "5" are the order and length of the code, respectively, forming a codified word of 32-bits for a 16-bits input. Depending on the order, Reed-Muller codes can correct many patterns of MCUs. However, the bigger the order applied, the higher the implementation cost.

Sánchez-Macián et al. [22] developed SEC-DAED and SEC-DED-TAED, which are methods for performing SEC-DED and Triple Adjacent Error Detection (TAED) using a matrix of bits codified with Hamming code. SEC-DAED can detect patterns of adjacent errors rearranging specific columns of the encoded matrix. Moreover, SEC-DED-TAED provides the matrix analysis to avoid miscorrection of triple errors.

Argyrides et al. [1] proposed the Matrix code, which organizes bits in a matrix format to correct MCUs in a 32-bits codeword. The code is composed of four rows of Hamming codified bits plus an additional row with parity bits to detect adjacent bit errors. The Matrix code focuses on systems subjected to DEC-DED, and its decoding consists of syndrome verification and parity bits. The Matrix code has low correction efficiency when compared to a more robust code like Reed-Muller, but with the advantage of consuming less area and energy.

Reviriego et al. [19] proposed a technique that uses SEC-DED code and parity, forming a 2D structure, which allows detecting double errors employing parity bits. The structure of this code is similar to the Matrix code [1]; however, using more redundant bits. Their work does not provide further analysis regarding the insertion of MCUs. In another approach for 2D codes, Reviriego et al. [20] developed a technique, which combines Hamming code with parity bits for data error detection/correction. A parity bit per row was added to minimize the SEC cost, and the check bits are read and stored only when the parity bits detect a single error.

Guo et al. [7] proposed the Decimal Matrix Code (DMC), which uses decimal sum with parity to create an algorithm capable of detecting and correcting errors. DMC can correct some specific MCU patterns with low implementation cost.

Hsiao et al. [10] proposed the Orthogonal Latin Square (OLS) codes, which assume the number of data bits applied must perform a perfect square (i.e., $m \times m$ bits). Liu et al. [14] extended OLS modifying the parity check bit matrix for supporting 32-bit data. This extension offers two code structures: one with less redundancy bits, lower cost, and lower error correction capability; and one with higher overall cost, but also higher error coverage.

CLC [3] is an ECC based on extended Hamming and parity bits with low area and power overheads that provide high detection and correction rates for protecting memory arrays from MCUs. CLC can be used not only to protect stored information in memory but also to protect information in a communication channel. In this paper, we explore two variations of the 2D structure applied in CLC [3] and the effects of error coverage and implementation cost.

Table 1 compares the ECC techniques presented in this section, regarding the error correction effectiveness and main features of the techniques. A common characteristic of many ECCs described in this work is the usage of a 2D structure for detecting and correcting errors.

3 CLC - Error Correction and Detection Code

CLC(16,40) codified word comprises of 40 bits: 16 data bits (D), 12 check bits (C), 8 column-parity bits (Pc), and 4 row-parity bits (Pr). Figure 1 shows CLC divides the 16 bits word into four words of 4 bits codified in extended Hamming, whereby the bits are relocated on four rows. CLC's structure is like the one presented by Reviriego et al. [19]; however, the CLC's algorithm aims at correcting and detecting several patterns of MCUs.

The information encoded in rows enables SEC-DED; CLC uses the parity information of each column to correct double and even more errors. Column-parity bits $Pc_0, \ldots, Pc_3$ are used to detect errors in the data bits columns, whereas $Pc_4, \ldots, Pc_7$ are used to detect errors in the check bits and row-parity bits. The combination of the extended Hamming in each row with the parity bits allows correcting MCUs in the data word as well as in both check and parity bits. Equations 1 to 3 describe how to calculate C_q, performed by XOR operations ($\oplus$), where q is the index of the check bit.

$$C_q = D_{4q/3+1} \oplus D_{4q/3+2} \oplus D_{4q/3+3} \quad \forall q \in \{0,3,6,9\} \tag{1}$$

$$C_{q+1} = D_{4q/3} \oplus D_{4q/3+2} \oplus D_{4q/3+3} \quad \forall q \in \{0,3,6,9\} \tag{2}$$

$$C_{q+2} = D_{4q/3} \oplus D_{4q/3+1} \oplus D_{4q/3+4} \quad \forall q \in \{0,3,6,9\} \tag{3}$$

According to the extended Hamming, Eq. 4 displays all Pr bits that are calculated as follows:

$$Pr_q = D_{4q} \oplus D_{4q+1} \oplus D_{4q+2} \oplus D_{4q+3} \oplus C_{3q} \oplus C_{3q+1} \oplus C_{3q+2} \forall 0 \le q \le 3 \tag{4}$$

Equations from 5 to 7 show how to calculate the Pc bits, where q is the column index of Pc.

$$Pc_q = D_q \oplus D_{q+4} \oplus D_{q+8} \oplus D_{q+12} \quad \forall 0 \le q \le 3 \tag{5}$$

$$Pc_q = C_{q-4} \oplus C_{q-1} \oplus C_{q+2} \oplus C_{q+5} \quad \forall 4 \le q \le 6 \tag{6}$$

$$Pc_7 = Pr_0 \oplus Pr_1 \oplus Pr_2 \oplus Pr_3 \tag{7}$$

CLC checks each row and column to verify the codeword consistency. The first step of CLC algorithm generates the Syndrome column-parity (SPc) and Syndrome Check bits (SC). This step is performed in two parts: (i) the first one recalculates C using the D bits of the data word, thereby forming a new value of C called Recalculated Check bit (RC); (ii) the second one makes XOR operations with the C bits stored at the codified word, whereby Eq. 8 defines SC.

$$SC_q = C_q \oplus RC_q \quad \forall 0 \le q \le 11 \tag{8}$$

Similarly, Eq. 9 shows the computation of SPc using Recalculated column-parity (RPc) bits.

$$SPc_q = Pc_q \oplus RPc_q \quad \forall 0 \le q \le 7 \tag{9}$$

Table 1 Related work comparison

ECC proposed	Effectiveness	Technique
Vargas et al. [25]	Correction of MCUs	Current checking + parity
Reed-Muller [26]	Correction of MCUs	Majority logic
SEC-DAED, SEC-DED-TAED [22]	SEC + DED + TAED	Hamming code + parity
Matrix code [1]	Correction of MCUs	Hamming code + parity
Reviriego et al. [19]	Double error correction	SEC-DED code + parity
Reviriego et al. [20]	Single error correction	Hamming code + parity
DMC [7]	Specific error patterns	Decimal sum + parity
Liu et al. [14]	Correction of MCUs	Extension of OLS codes
CLC [3]	Correction of MCUs	Extended Hamming+parity

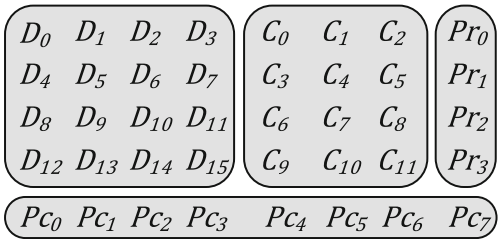


Fig. 1 CLC codeword format

Error detection in codified words is highlighted when the *SPc* and *SC* bits generated in rows and columns, respectively, are not zero. Besides, the *Pr* bits are also recalculated (*RPr*) for each row to cover triple errors in the same row, enabling generating a Syndrome row-parity (*SPr*) word, which is calculated according to Eq. 10.

$$SPr_q = Pr_q \oplus RPr_q \quad \forall 0 \le q \le 3 \tag{10}$$

CLC operates in standard or extended modes allowing correcting MCUs based on parity bits and the extended Hamming (8,4), where "8" is the length of the encoded word and "4" is the number of data bits.

Standard CLC mode (CLC-S) targets low overhead of area and energy. The correction procedure of CLC-S includes the generation of syndrome bits, analysis of the correction table, and rows correction procedure, which is performed once.

Extended CLC mode (CLC-E) executes the correction procedures two times for some cases where more than one row in the CLC encoded data were affected. The double correction is always performed by CLC-E, presenting better coverage of multiple-error patterns but consuming more area and energy than CLC-S. The *SPr* bits allow both CLC modes to correct adjacent triple errors in one row of the data matrix if in each affected column there is no more than one bit-flip. CLC-E corrects any triple error recalculating the syndrome bits. We implemented the scheme of multiple-error patterns by adding only extra logic, without redundancy increase, and without generating extra storage costs, nor increasing the probability of errors resulting from the increase of redundancy bits.

This work explores the efficacy and efficiency of both CLC modes comparing them to Matrix and Reed-Muller(2,5) approaches. Figures 2 and 3 show the encoding and decoding algorithms of CLC for a 16-bit data word. Henceforth, we explain the correction process using CLC-S, which corrects

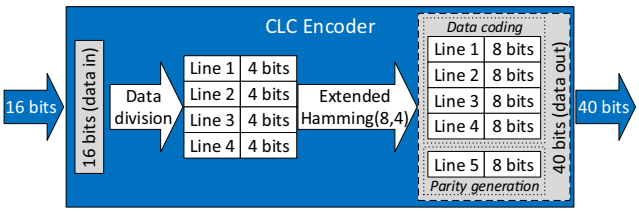


Fig. 2 Organization of the main blocks of the CLC(16,40) encoder

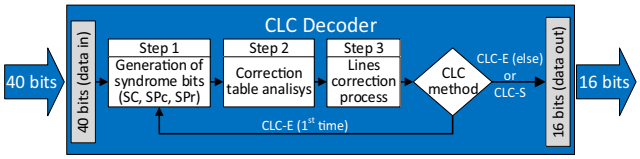


Fig. 3 Organization of the main blocks of the CLC(16,40) decoder

single errors using the extended Hamming decoders, and multiple errors using parity.

Equation 11 computes SCr_q, which is a reduction in one bit of the error detection of all *SC* bits of each row; where $q \in \{0, 1, 2, 3\}$ represents the rows of each set of data bits.

$$SCr_q = SC_{3q} \oplus SC_{3q+1} \oplus SC_{3q+2} \quad \forall 0 \le q \le 3 \tag{11}$$

Table 2 describes how the syndromes are used to correct errors as well as the correction method; i.e., parity or Hamming means the correction method uses *SPc* bits or Hamming algorithm to correct the detected error, respectively.

Syndromes are analyzed for each data row of the encoded word. When all the syndromes are zero, there is no error in the decoded word. Otherwise, the error bits are corrected using extended Hamming algorithm and parity bits. The first method employs classical extended Hamming decoding throughout *SC* and *SPr*. The second method flips the bits according to the error positions indicated by the *SPc* bits. The last syndrome pattern ($SCr = 1$, $SPr = 1$, $SPc = 1$) uses both methods regarding the number of errors in the columns and rows. Extended Hamming is employed when *SPc* bits contain less than three errors. If there is only one row with $SCr = 1$, the *SPc* bits are used; otherwise, extended Hamming is used.

Figure 4 depicts the first example, in which the data word is organized as a matrix, and the values of the calculated syndromes are shown in a table (value *one* indicates the syndrome is not zero; otherwise, the syndrome bits are *zero*). Figure 4 exemplifies a codified word where only D_0 has an error corrected by extended Hamming.

Figure 5 illustrates a triple error on bits D_0, D_1 and D_2 as well as the associated syndromes. In this case, the

Table 2 Correction table based on syndrome bits

SCr	*SPr*	*SPc*	Status	Method
0	0	0	No error	–
0	0	1	Error detected	–
0	1	0	Error detected	–
0	1	1	Triple error corrected	Parity
1	0	0	Error detected	–
1	0	1	Even errors corrected	Parity
1	1	0	Odd errors corrected	Hamming
1	1	1	Odd errors corrected	Hamming and Parity

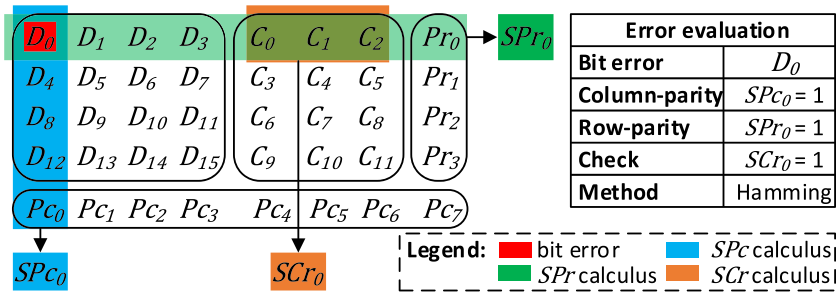


Fig. 4 Single error corrected by CLC

combination of the SPr_0, SPc_0, SPc_1, and SPc_0, bits indicates there are errors on the first three columns of the first row.

Figure 6 exemplifies an error pattern with CLC-E executing a double error checking. In the first error verification, CLC-E corrects D_1 using extended Hamming and D_4 using the combination of the SCr and SPc bits. However, D_0 is still in error. Then, CLC carried out a second error verification step to correct bits through extended Hamming.

Note the correction of D_0 is perceived only after the first step since at the first moment $SPc_0 = 0$ and $SPr_0 = 0$; however, when correcting D_1 and D_4, both SPc_0 and SPr_1 change to 1.

4 Variation of CLC 2D Structure

Section 3 shows the CLC encoder, decoder, correction algorithm and extended mode along with examples of code correction. That CLC divides the 16-bit of data in four rows and applies extended Hamming with 4 bits producing the format CLC(16,40). This section explores two other formats of CLC dividing the same 16 data bits into two and eight rows to explore the effect of code size and format variations: (i) CLC(16,39), which splits the data bits into two rows with 23 bits of redundancy; and (ii) CLC(16,54), which divides the data bits into eight rows and has 38 bits of redundancy. Note that not only the quantity of bits change but also the number of bits used as parity and Hamming.

Figure 7 shows the CLC(16,39) structure, which divides the 16 data bits into two rows applying a code with the Extended Hamming(8,13). The 39 bits of the codeword are divided in the following four fields: 16 Data bits (D), 10 Check bits (C), 13 column-parity bits (Pc) and 2 raw-parity bits (Pr).

Equations 12 to 15 describe how to calculate C_q, where q is the check bit index.

$$C_q = D_{8q} \oplus D_{8q+1} \oplus D_{8q+3} \oplus D_{8q+4} \oplus D_{8q+6} \quad \forall q \in \{0,4\} \quad (12)$$

$$C_q = D_{8q} \oplus D_{8q+2} \oplus D_{8q+3} \oplus D_{8q+5} \oplus D_{8q+6} \quad \forall q \in \{1,5\} \quad (13)$$

$$C_q = D_{8q+1} \oplus D_{8q+2} \oplus D_{8q+3} \oplus D_{8q+7} \quad \forall q \in \{2,6\} \quad (14)$$

$$C_q = D_{8q+4} \oplus D_{8q+5} \oplus D_{8q+6} \oplus D_{8q+7} \quad \forall q \in \{3,7\} \quad (15)$$

Equation 16 displays all Pr bits are calculated according to the extended Hamming.

$$Pr_q = D_{8q} \oplus \ldots \oplus D_{8q+7} \oplus C_{4q} \oplus \ldots \oplus C_{4q+3} \quad \forall q \in \{0,1\} \quad (16)$$

Equations from 17 to 19 show how to calculate the Pc bits, where q is the column index of Pc.

$$Pc_q = D_q \oplus D_{q+8} \quad \forall 0 \le q \le 7 \quad (17)$$

$$Pc_{q+8} = D_q \oplus D_{q+4} \quad \forall 0 \le q \le 3 \quad (18)$$

$$Pc_{12} = Pr_0 \oplus Pr_1 \quad (19)$$

Figure 8 illustrates the CLC(16,54) format with 16 data bits organized in 8 pairs of bits, Extended Hamming(2,6) and 14 columns/rows parity bits.

CLC(16,54) has 38 bits of redundancy, which is a vast quantity proportionally to the data bits, the main reason for this impact is because Extended Hamming for 2 and 4 bits requires the same amount of redundancy bits. Equations 20 to 22 describe how to calculate C_q - q is the check bit index.

$$C_{3q} = D_{2q} \oplus D_{2q+1} \quad \forall 0 \le q \le 7 \quad (20)$$

$$C_{3q+1} = D_{2q} \quad \forall 0 \le q \le 7 \quad (21)$$

$$C_{3q+2} = D_{2q+1} \quad \forall 0 \le q \le 7 \quad (22)$$

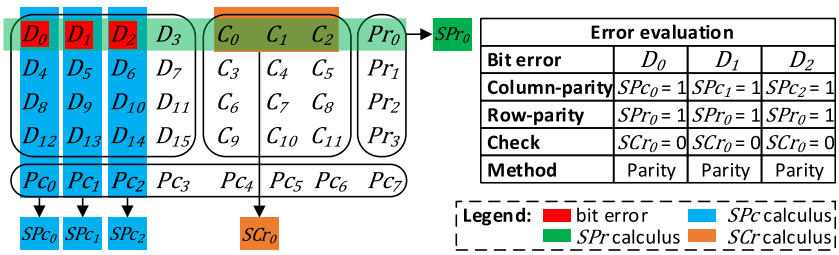


Fig. 5 Triple error corrected by CLC

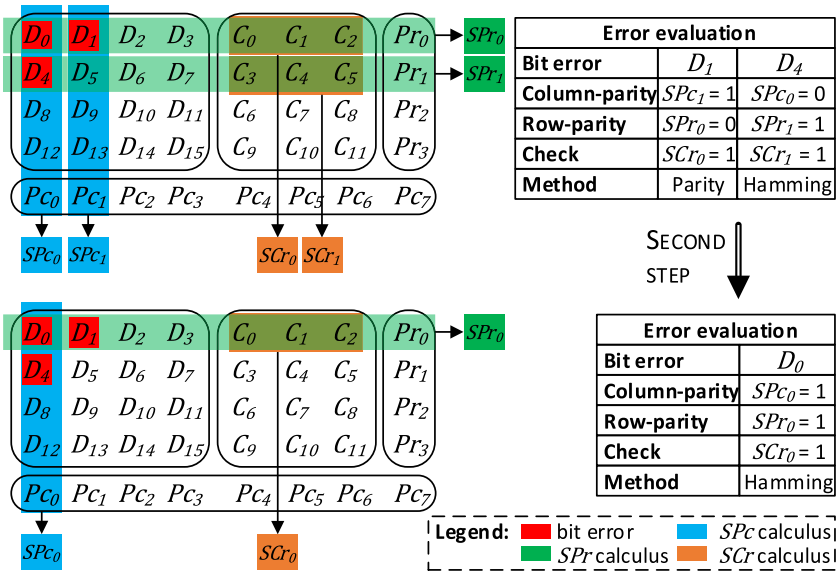


Fig. 6 Triple error corrected by CLC-E

Equation 23 computes the row-parity bits, and Equations from 24 to 27 calculate the column-parity bit, all equations, except Eq. 27, using q as index.

$$Pr_q = D_{2q} \oplus D_{2q+1} \oplus C_{3q} \oplus C_{3q+1} \oplus C_{3q+2} \quad \forall 0 \le q \le 7 \tag{23}$$

$$Pc_q = D_q \oplus D_{2+q} \oplus D_{4+q} \oplus D_{6+q} \oplus D_{8+q} \oplus D_{10+q} \oplus D_{12+q} \oplus D_{14+q} \quad \forall q \in \{0,1\} \tag{24}$$

$$Pc_{q+2} = C_q \oplus C_{3+q} \oplus C_{6+q} \oplus C_{9+q} \oplus C_{12+q} \oplus C_{15+q} \oplus C_{18+q} \oplus C_{21+q} \quad \forall q \in \{0,1,2\} \tag{25}$$

$$Pc_5 = Pr_0 \oplus Pr_1 \oplus Pr_2 \oplus Pr_3 \oplus Pr_4 \oplus Pr_5 \oplus Pr_6 \oplus Pr_7 \tag{26}$$

All formats of CLC apply the same algorithm to detect and correct bit flips, which consists in the analysis of each row of the codeword and the application of the correction of Table 2.

5 Redundancy Analysis

Previous sections described CLC(16,40), CLC(16,39) and CLC(16,54), which are three CLC formats using 16 data bits words, employing Extended Hamming(4,8), Extended Hamming(8,13) and Extended Hamming(2,6), respectively. These codes add a considerably quantity of redundancy bits; i.e., 150%, 143.75 and 237.5% in the formats CLC(16,40), CLC(16,39) and CLC(16,54), respectively. However, this section shows that as the number of data bits increases, the percentage of redundancy is reduced.

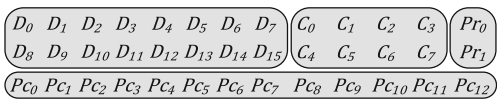


Fig. 7 CLC(16,39) codeword format

Let D_r be the number of data bits and R_r be the redundancy bits, both in a single row, than Table 3 provides the D_r and R_r of the Extended Hammings applied in this paper.

Let D_T be the total number of data bits codified by CLC than Eq. 27 provides the calculation of DR_T, meaning the sum of all data and redundancy bits of a CLC codeword.

$$DR_T = \left(\frac{D_T}{D_r} + 1\right) \times (D_r + R_r) \tag{27}$$

Where $\left(\frac{D_T}{D_r} + 1\right)$ is the number of rows the CLC 2D scheme have (note that "+1" represents the addition of the parity bits row). Besides, the total number of redundancy bits (R_T) is DR_T- D_T; therefore, the percentage of redundancy over data bits (R_A) is computed with Eq. 28.

$$R_A = \frac{R_T}{D_T} \times 100 \tag{28}$$

Table 4 shows the results R_A regarding the data bits inserted for each CLC scheme.

All CLC structures present redundancy reduction with the increase of data bits; however, CLC with 13 bits per row,

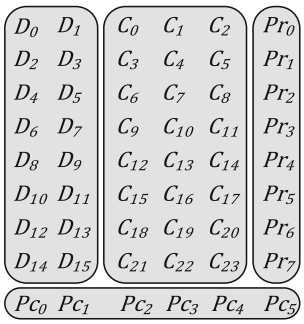


Fig. 8 CLC(16,54) codeword format

Table 3 Bits distribution of the Extended Hamming used on CLC formats

Redundancy type	D_r (bits)	R_r (bits)	Redundancy addition (%)
Extended Hamming (2,6)	2	4	200.0
Extended Hamming (4,8)	4	4	100.0
Extended Hamming (8,13)	8	5	62.5

which applies Extended Hamming(8,13), presented the lowest redundancy and the largest cutback in R_A rate from all CLCs proposed. For instance, CLC with 128 data bits has an R_A rate with almost half value of the 16 data bits version. Meanwhile, CLC with 6 bits per row is both the code with more redundancy bits and with lower R_A reduction. These results are explained by the fact that the Extended Hamming (8,13) presents from all applied the lowest redundancy addition rate, while it is quite the opposite for Extended Hamming (2,6).

6 Failure Injection Experiments

This section describes experiments comparing the CLC (16,39), CLC(16,40), CLC(16,54), Matrix and Reed-Muller (2,5) codes regarding correction and detection error estimation. Besides, the three CLC versions regard both correcting modes (i.e., CLC-E and CLC-S).

The experiments were made using a testbench in Matlab that pseudo-randomly inserts from 1 to 8 errors in adjacent memory cells to mimic the structure of MCUs, which tend to be physically close [23]. The testbench inserts errors in both data and redundant bits. Due to the number of code variations, we split the results into 2 groups comparison - Figs. 9 and 10 show the experimental results and correction analysis regarding the number of errors with standard CLCs and extended CLCs. Figures 9 and 10 demonstrate all methods are 100% effective for scenarios with a single error. When the errors increase, Matrix loses efficacy fast due to its lower number of redundancy bits. Reed-Muller loses efficacy abruptly for more aggressive error scenarios (i.e., 4 or more errors), which is explained by the majority logic applied in the Reed-Muller has a distance of eight [3], implying a sharp correction rate for only three errors. We emphasize as much more complex the Reed-Muller is applied, the more area and power requirements are necessary.

Table 4 CLC's 2D features for multiple sizes of data bits

D_T	R_T	D_r	R_r	Rows	Columns	R_A (%)
16	38	2	4	9	6	237.50
	24	4	4	5	8	150.00
	23	8	5	3	13	143.75
32	70	2	4	17	6	218.50
	40	4	4	9	8	125.00
	33	8	5	5	13	103.10
64	134	2	4	33	6	209.40
	72	4	4	17	8	112.50
	53	8	5	9	13	82.80
128	262	2	4	65	6	204.70
	136	4	4	33	8	106.30
	93	8	5	17	13	72.60

Figure 9 depicts all CLCs correct single and double errors and achieve better overall results than Matrix in both modes. All CLC-S cannot deal with all patterns of 3 errors, as explained in Section 3. Nonetheless, for more aggressive error patterns CLC codes present better results than Reed-Muller (2,5), especially, for CLC-S(16,54), which achieves more than 50% of efficacy from 1 to 5 errors since more than 70% of its codeword is composed by redundant bits. The correction efficacy of CLC-S(16,39) and CLC-S(16,40) are right behind the correction efficacy of CLC-S(16,54), presenting very close results.

Figure 10 shows the extended CLCs correct all 3-error patterns; also, the extended mode allows all CLCs boost their correction efficacy in most of the error patterns analyzed. CLC-E(16,54) achieved the best overall results, mainly because its higher quantity of redundancy bits compared to the other CLC versions, although its results for 7 and 8 errors were similar to the standard mode versions. Besides, the extended mode is the only code mode allowed the correction of some patterns with 8 errors.

Analyzing the correction error results regarding the 2D structure of CLCs, we infer from Fig. 7 CLC(16,39) allowed the fewer occurrence of MCUs affecting the same columns in the region of data bits, which means more errors occurred in the region of redundant bits. The more aggressive the error patterns are, the more columns are affected, making it more difficult to detect these errors by *Pr* bits, this allowed that both CLC(16,39) modes outperformed both CLC(16,40) modes for 6, 7 and 8 errors.

The 2D structure of CLC(16,59), shown in Fig. 8, has 8 rows with data bits, increasing the chances to have errors in the same columns and consequently the number of complex errors. Therefore, although CLC(16,59) has a higher quantity of redundant bits than CLC(16,39), the correction results for 6, 7 and 8 errors are very close.

Comparing Figs. 1, 2, 3, 4, 5, 6, 7, and 8, we notice only CLC(16,40) has a square 2D structure (i.e., a 4×4 matrix). Thus, as the complexity of the error patterns grows, more data

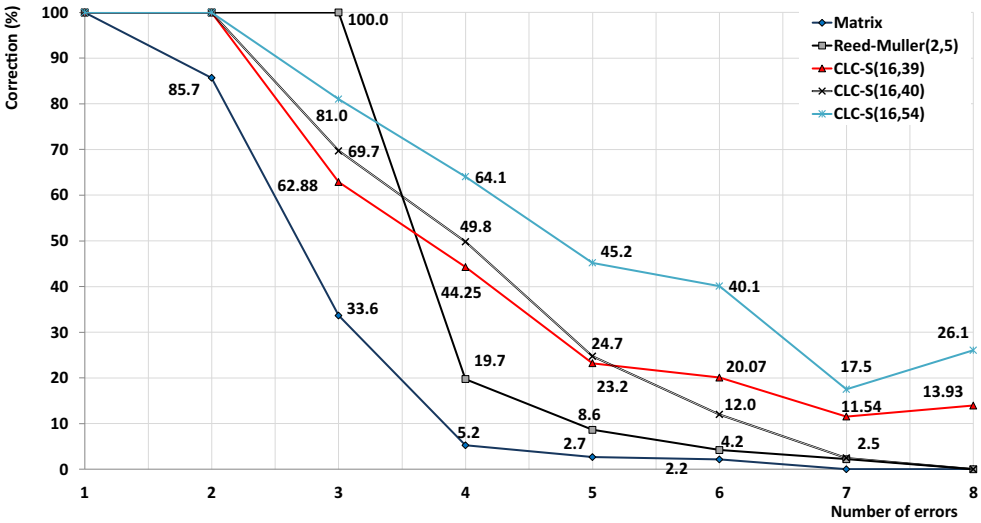


Fig. 9 Error correction rates comparison with standard CLCs

bits of CLC(16,40) are affected, reducing the code correction efficacy in comparison to the others.

Figure 11 shows all CLC versions reach better detection results than Matrix and Reed-Muller(2,5). Besides, both CLC modes have the same detection rates (so, the figure uses only CLC, instead of CLC-E or CLC-S), since the standard and extended modes only influence the correction performance.

One striking result is the increase in the CLC error detection rate when the injection of error rises from 5 to 6. This phenomenon is explained by the fact that CLC's algorithm applies extended Hamming for each row, allowing the detection of double errors meanwhile the code cannot differ an odd number of errors (i.e., 1, 3, 5).

Note that no code detected 8 errors; but, all CLCs in both modes, except CLC-S(16,40), correct some patterns of 8 errors. Our experiments considered a CRC only detected an error if the number of syndromes equivalent to this error was detected. An 8-bit error pattern is a very complex type, most of this errors in CLC's structures

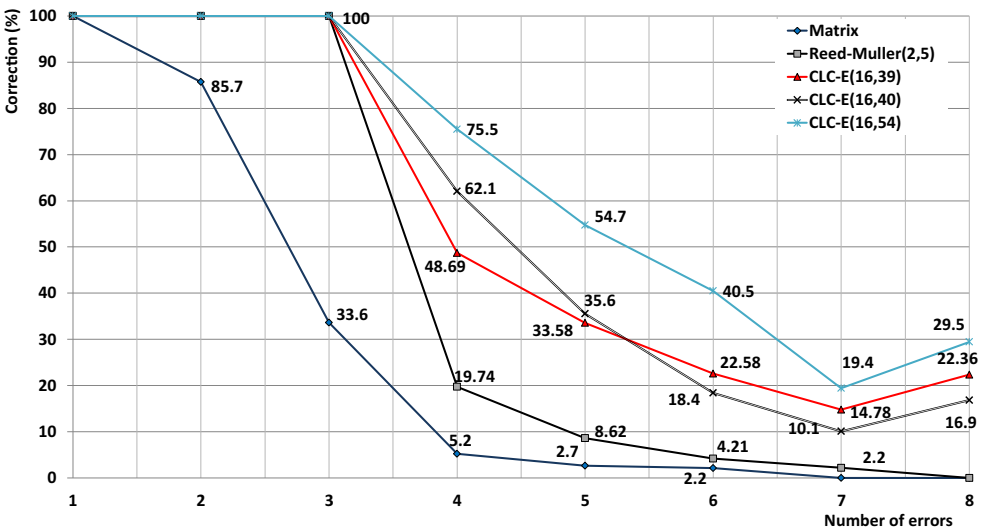


Fig. 10 Error correction rates comparison with extended CLCs

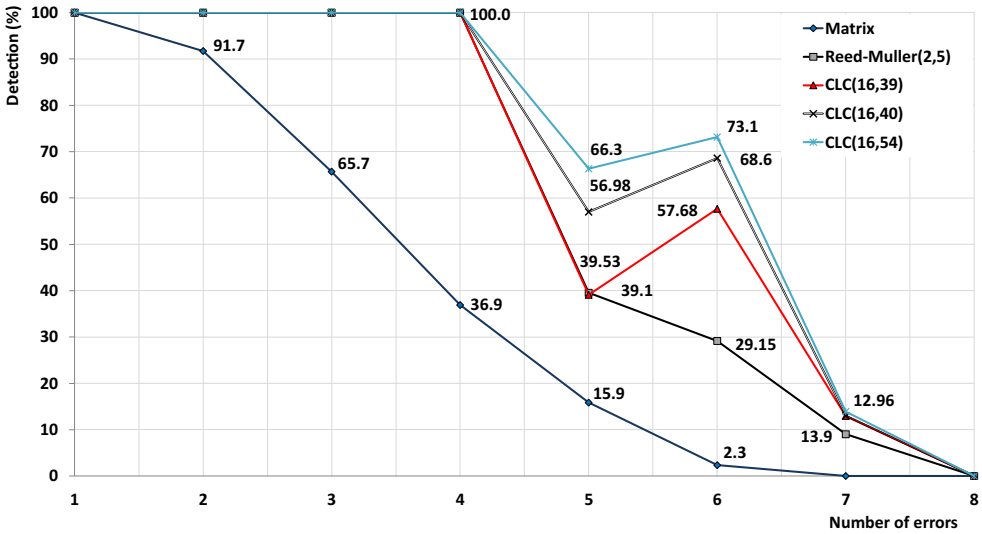


Fig. 11 Detection rates comparison with extended and standard CLCs

affected more redundant bits than data bits, this helps the correction of data bits, but not the detection, that depends on the redundant bits reliability. In situations of accumulated errors, the appliance of memory refresh techniques it is also required to correct transient and accumulated faults; however, this procedure increases the energy consumption [15]. Nonetheless, as presented in the faults experiments, the high correction rates of CLC-S and CLC-E diminish the necessity of memory refresh.

7 Reliability Estimation

This section shows a series of experiments using Matlab to evaluate the reliability of the proposed techniques.

7.1 Detection and Correction Capabilities

Similarly to the works [17, 21], this one assumes transient errors occur with a Poisson distribution and bit errors are statistically independent. Equation 29, which is based on Argyrides et al. [2], depicts the error detection ($E_{Detection}$) at each time (t) as a function of the probability of detecting errors (ED) in a possible faulty memory (MF) regarding Me.

$$E_{Detection}(t) = \sum_{i=1}^{Me} P(ED/MF) = \sum_{i=1}^{Me} P(ED/iE) \times P(iF/MF) \quad (29)$$

Where Me is the maximum number of errors that can arise during t, and iE is the probability of occurring errors in an MF.

Equation 30 depicts the probability of occurring errors (iE) in an MF, whereas, Eq. 31 provides the probability of occurring i errors in a word with $(d+p)$ bits.

$$P(iE/MF) = \frac{P(iE)}{P(MF)} \quad (30)$$

$$P(iE) = \binom{d+p}{i} \times \left(1-e^{-\lambda t}\right) \times e^{-\lambda(d+p-i)t} \quad (31)$$

Where the number of data bits is d, p denotes the protection bits, λ is the upset occurrence rate, and t is the time. Equation 32 gives the probability of the memory is faulty.

$$P(MF) = 1-e^{-\lambda(d+p)t} \quad (32)$$

Placing Eqs. 30–32 in Eq. 29, the error detection at each time is computed according to Eq. 33. The detection coverage considers the occurrence from one to eight flip bits. Also, $P(ED/iE)$ is obtained on Section 6 through statistical analysis regarding several error injections presented in Figs. 9, 10 and 11.

$$E_{Detection}(t) = \sum_{i=1}^{Me} P(ED/iE) \times \frac{\binom{d+p}{i} \times \left(1-e^{-\lambda t}\right) \times e^{-\lambda(d+p-i)t}}{1-e^{-\lambda(d+p)t}} \quad (33)$$

Figures 12 and 13 compare the correction coverage of Matrix, Reed-Muller(2,5), and the three formats of CLC-S and CLC-E, respectively. As the time rises, the probability of multiple errors occurrence also rises as well; thus, the curves of correction capability of all methods decrease.

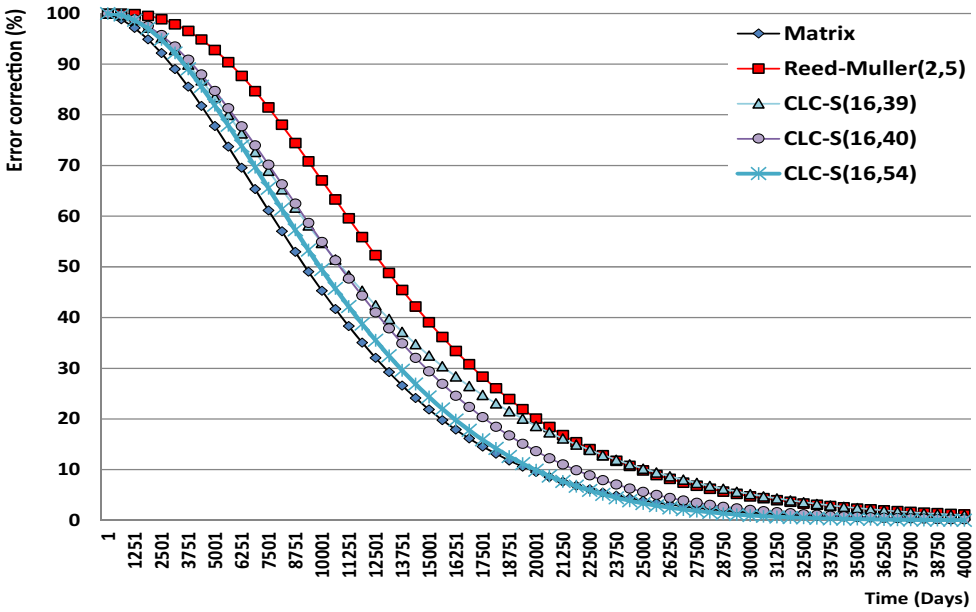


Fig. 12 Error correction for the three standard formats of CLC, Matrix and Reed-Muller(2,5) ($\lambda = 10^{-5}$ upsets/bit per and day 16-bit word size)

Figure 12 shows that despite the higher correction rates for all formats of CLC-S, for the majority of scenarios, Matrix and Reed-Muller(2,5) have the higher and lower execution time for error correction, respectively. Although more redundant bits allow dealing with more MCU patterns, a large number of these bits also impacts the error correction capability, raising the probability of bit-flips.

The formats of CLC-E improve the error correction significantly, allowing CLC-E(16,39) to have similar results in the early days and outperform Reed-Muller(2,5) from day 13,125 until the end of the experiment as is depicted in Fig. 13. However, the high redundancy downgraded the CLC(16,59) performance, being only ahead of Matrix code.

Figure 14 shows that even for detection analysis, Reed-Muller(2,5) achieved better results than the other codes. CLC(16,40) presents marginally better results than the other CLC versions; however, the difference increases from 1% (day 5000) to 72% (day 40,000). Again, the high redundancy of CLC(16,54) impacts in the experimental results, which allowed it to be surpassed by the Matrix code in day 12,501.

7.2 Reliability and MTTF Estimation

This section evaluates the reliability and MTTF of the ECCs regarding a memory containing 16 registers, each one with 16 data bits. Equation 34 computes the reliability of a register

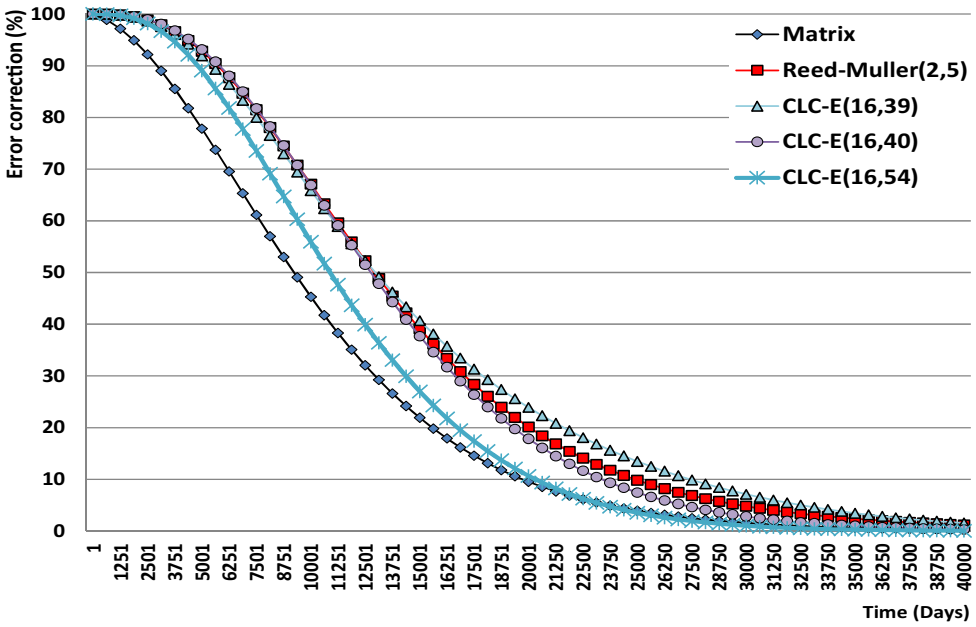


Fig. 13 The same of Fig. 12, exchanging CLC-S by CLC-E

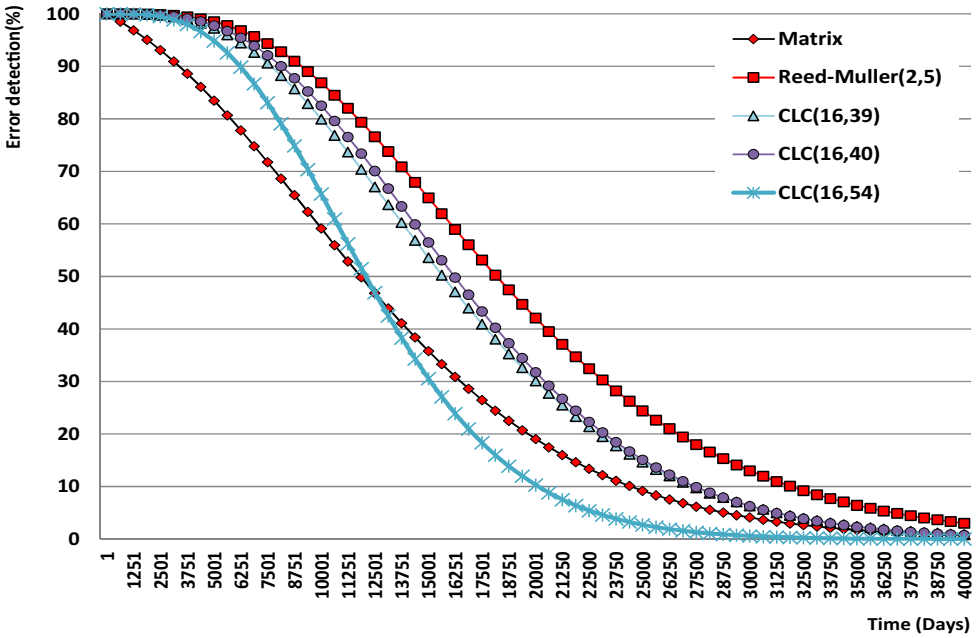


Fig. 14 Error detection capability for all codes according to the execution time ($\lambda = 10^{-5}$ upsets/bit per day, 16-bit word size)

over time. Where $1 - P(MF)$ is the probability of non-errors occurrence, *iCI* denotes *i* correctable errors, and *P(iCI)* the probability of *iCI* occurrence. Thus, $\frac{\sum_{i=1}^{Me} P(iF) \times P(iCI)}{Me}$ defines the probability of occurrence of errors that can be corrected by the code to be evaluated.

$$r(t) = 1 - P(MF) + \frac{\sum_{i=1}^{Me} P(iE) \times P(iCI)}{Me} \tag{34}$$

Let M be the number of registers, then the memory reliability $R(t)$ over time is calculated through the product of reliabilities of all registers. Subsequently, considering the reliabilities of registers having the same values, Eq. 35 calculates the reliability of a memory device.

$$R(t) = r(t)^M \tag{35}$$

Equation 36 illustrates that *MTTF* of the memory device protected by the ECC is obtained through the integration of the reliability function.

$$MTTF = \int_0^\infty R(t)\, dt \tag{36}$$

Figure 15 shows the reliability over time of a word with 16 bits for Matrix and Reed-Muller(2,5) codes and all formats of

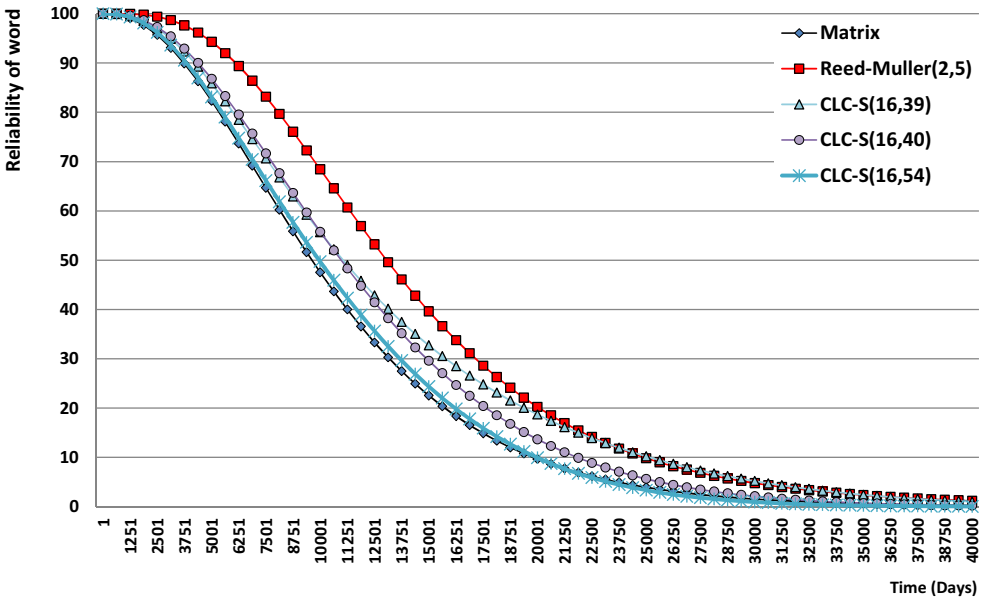


Fig. 15 Effect of time on the reliability of a 16-bit word for all formats of CLC-S, Matrix, and Reed-Muller(2,5) codes ($\lambda = 10^{-5}$ upsets/bit/day)

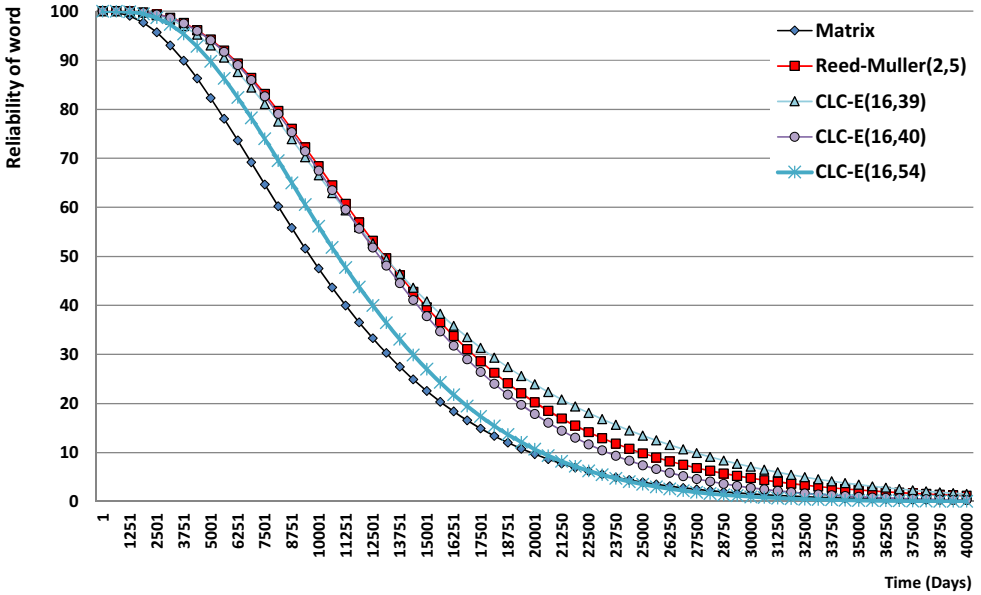


Fig. 16 Effect of time on the reliability of a 16-bit word for all formats of CLC-E, Matrix, and Reed-Muller(2,5) codes ($\lambda = 10^{-5}$ upsets/bit/day)

CLC-S, with an upset occurrence rate of 10^{-5} bits per day. This experiment proves that Reed-Muller(2,5) achieved the best results, followed by all CLC-S formats.

Figure 16 shows the same type of results of Fig. 15, but with all formats of CLC-E instead of CLC-S. These results show a reliability increase; with CLC-E(16,39) and CLC-E (16,40) very competitive compared to Reed-Muller(2,5). In fact, CLC-E(16,39) surpasses Reed-Muller(2,5) after 13,751 days, and the reliability difference between Reed-Muller(2,5) and CLC-E(16,40) is meaningless before 13,751 days.

Figure 17 compares the reliability of all formats of CLC-S, Matrix, and Reed-Muller(2,5) codes for a memory of 16 words. The rate of upsets occurrence (λ) is 10^{-5} bits per day. Though Reed-Muller(2,5) stands out relatively, the reliability of all codes drops highly through the days since the results reflect the composed reliability of all registers.

Figure 18 shows the same type of results of Fig. 17, but employing all formats of CLC-E instead of CLC-S. Despite the significant difference between CLC-E(16,40) and Reed-Muller(2,5) redundancies, the variation on reliabilities is less than 5%. The increase of correction capacity, provided by the extended CLCs, is the main reason for these results, although the vast redundancy of both modes of CLC(16,54) reduces the reliability of the memory meaningfully.

Table 5 shows the results of MTTF for memory devices protected by the Matrix, Reed-Muller(2,5) codes and both modes of CLC. The MTTFs of all formats and modes of

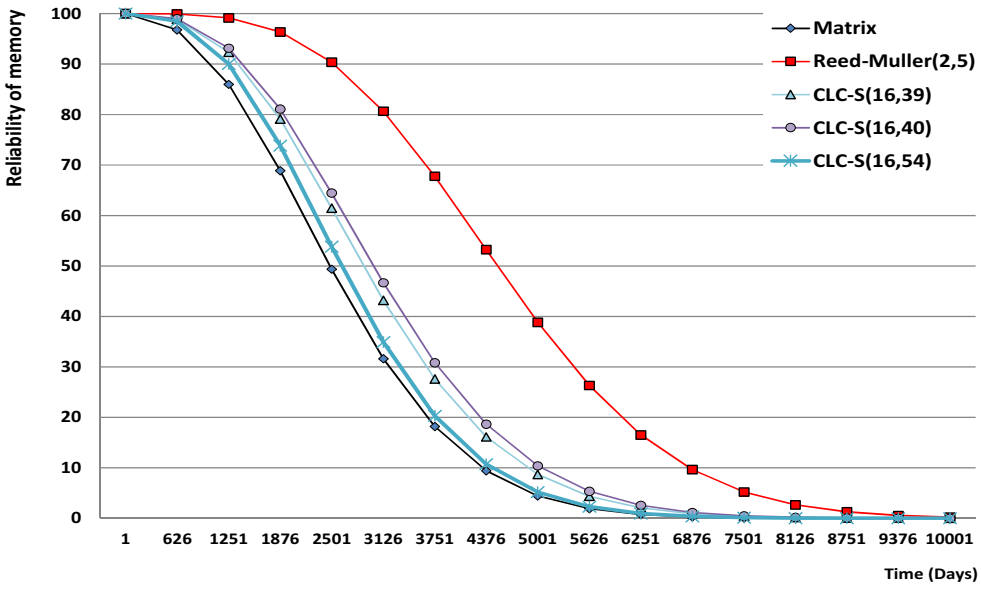


Fig. 17 Effect of time on the reliability of a 16-word memory ($\lambda = 10^{-5}$ upsets/bit/day, 16-bit word size)

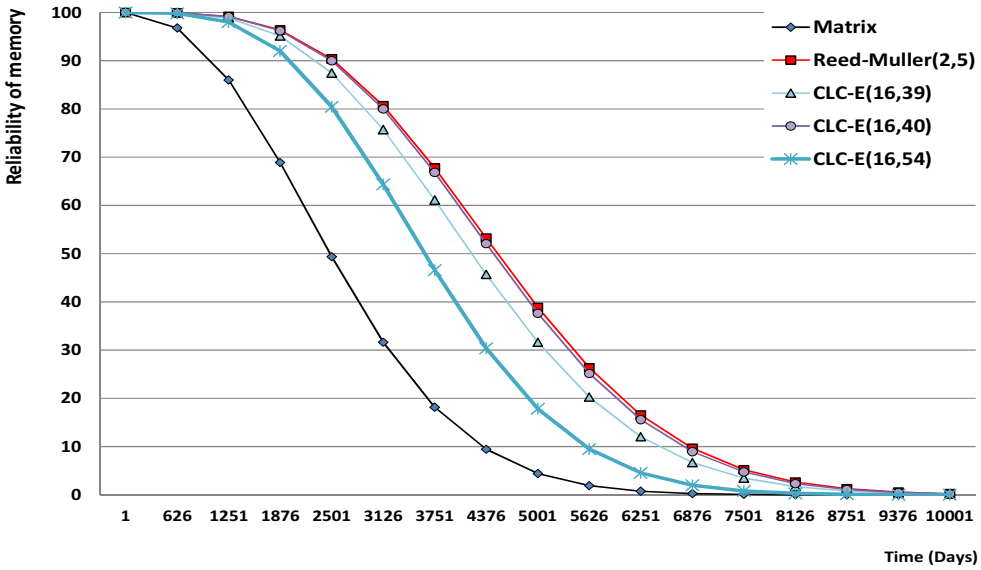


Fig. 18 Effect of time on the reliability of a 16-word memory ($\lambda = 10^{-5}$ upsets/bit per day, 16-bit word size)

CLC have an intermediate value between the Matrix and Reed-Muller(2,5) codes. Besides, the MTTF of Reed-Muller (2,5) is only 3.2 and 6.5% greater than the MTTF of CLC-E (16,40) and CLC-E(16,39).

8 Synthesis Results

This section presents and discusses the synthesis results of the implementations of all modes and formats of CLC, Reed-Muller(2,5) and Matrix. Tables 6 and 7 show the area consumption, power dissipation, and maximum delay, for the encoder and decoder implementation, respectively, which were obtained with Cadence's Encounter RTL Compiler for a 65 nm CMOS technology. Note that standard and extended CLC modes use the same encoders; thus, only the decoder of the extended mode has an additional synthesis cost. The CLC codes proposed, as well the other ECC used in this paper, are entirely combinational, being suitable for applications that require a low delay.

Table 6 displays Reed-Muller(2,5) and Matrix have the highest and lowest power dissipation and delay, respectively. CLC(16,54) achieved lower overall results in area and power consumption, even when comparing to CLC(16,39) and CLC (16,54) due to the extended Hamming applied in this code being less complex than the other two CLCs. Matrix applies 4-bits Hamming, implying a small difference (near to 8% higher) in comparison to CLC(16,54). CLC(16,39) and CLC (16,40) presented quite similar results, despite the Hamming applied in first required using less area than the second, reducing the power dissipation. The maximum delay was close in all formats of CLC, the extended Hamming complexity and number of data bits divisions are the main reasons for affecting the delay in each code.

Table 7 illustrates the Matrix code has the lowest of all costs because this code has a simpler algorithm in comparison to Reed-Muller(2,5), while when compared to CLCs, the appliance of extended Hamming in CLC's algorithm increases its redundancy, raising the overall cost. Both CLC-E(16,54) and Reed-Muller(2,5) decoders are highly area consumers, whereas CLC-S(16,39) and Matrix are the most economical ones. The power dissipation is one of the higher bottleneck of the Reed-Muller(2,5) decoder since it consumes more than 30% than CLC-E(16,54) and twelve

Table 5 MTTF of memory devices - 16-word memory, 16-bit word size, $\lambda = 10^{-5}$ upsets/bit/day

Code	Error rate	Code	Error rate
Matrix	261.18	CLC-S(16,40)	314.64
Reed-Muller(2,5)	461.87	CLC-E(16,40)	446.87
CLC-S(16,39)	303.66	CLC-S(16,54)	275.79
CLC-E(16,39)	431.93	CLC-E(16,54)	373.09

Table 6 Synthesis results of the encoders – absolute and normalized values – concerning Reed-Muller(2,5)

Method	Area		Power		Delay	
	(μm^2)	(%)	(mW)	(%)	(ns)	(%)
Reed-Muller (2,5)	504	100.0	0.037	100.0	0.74	100.0
Matrix	298	59.1	0.010	27.0	0.15	20.2
CLC(16,39)	402	79.7	0.023	62.1	0.39	52.7
CLC(16,40)	435	86.3	0.024	64.8	0.35	47.3
CLC(16,54)	258	51.2	0.011	29.7	0.41	55.4

Table 7 Synthesis results of the decoders – absolute and normalized values – according to Reed-Muller(2,5)

Method	Area		Power		Delay	
	(μm²)	(%)	(mW)	(%)	(ns)	(%)
Reed-Muller(2,5)	4312	100.0	0.737	100.0	2.12	100.0
Matrix	1090	25.2	0.050	6.7	1.01	59.5
CLC-S(16,39)	1194	27.7	0.085	11.5	1.30	61.3
CLC-E(16,39)	2612	60.5	0.268	36.3	2.55	120.2
CLC-S(16,40)	1351	31.3	0.076	10.3	1.33	62.7
CLC-E(16,40)	3360	77.9	0.331	44.9	2.50	118.0
CLC-S(16,54)	1665	38.6	0.096	13.0	1.35	63.6
CLC-E(16,54)	4418	102.4	0.490	66.4	2.53	119.3

Legend: cells marked in blue and red have the best and worst results, respectively

times more than Matrix, which is the most power-efficient code followed by CLC-S(16,40) and CLC-S(16,39). To implement the syndrome calculation to correct more aggressive error scenarios with substantial redundancy, the CLC-E(16,54) decoder has more than 2 times the delay of CLC-S(16,39). The analysis of these results shows that although Reed-Muller(2,5) and CLC-E(16,54) are the most efficient codes for error correction, CLC-S(16,39) and Matrix are the most efficient codes regarding area consumption, power dissipation, and delay. Finally, as a general conclusion regarding the CLC codes, the increase of rows of data bits impacts the synthesis cost for the decoders, since the error correction algorithm of standard and extended CLC modes verifies all rows to correct bit-flips.

9 Detection, Correction and Costs Analysis

Correlating high error correction rates with low overhead of silicon implementation allow choosing an efficient ECC for memory devices. Argyrides et al. [2] proposed Correction Coverage per Cost (CCC) and Detection Coverage per Cost (DCC) metrics to cover these combined requirements. However, by splitting up the total coverage into two metrics, one misses the overall efficiency picture of a given code. Therefore, a more appropriate metric should consider the two-coverage metrics together, because both detection and correction are critical phases for codes applied for fault tolerance reasons.

We propose in Eq. 37 the Total Coverage per Cost (*TCC*), which is a design metric that covers the ECC implementation cost (area consumption, power dissipation and maximum delay for the decoder modules). *TCC* highlights the importance of the primary design metric, which is error detection/correction performance, particularly for MCU errors. *TCC* is directly proportional to the *fault tolerance coverage*, which encompasses both error detection and recovery simultaneously. Additionally, *TCC* is inversely proportional to the costs for *data word coding/decoding process*, which covers area consumption, power dissipation, and delay. Thus, the higher *TCC* is, the better the method is.

$$TCC = \frac{Detection\ rate \times Correction\ rate}{Area \times Power \times Delay} \tag{37}$$

For each coding method, the parameters of Eq. 37 are obtained with the following rules: (i) *Correction rate* and *Detection rate* are the same values shown in Figs. 9, 10 and 11, which were extracted from the experiments described in Section 4; (ii) *Area*, *Power,* and *Delay* are the ones described in Table 7. Figure 19 shows the *TCC* produced by Eq. 37. Note that rising *Area*, *Power,* and *Delay*, minimizes *TCC*, and the opposite occurs with the increase of the *Correction rate* and *Detection rate*.

Figure 19 shows a tendency of reducing *TCC* with the increase of errors, which is a consequence of the correction and detection rates reduction. Additionally, Reed-Muller(2,5) and all formats of the extended CLC have insignificant *TCCs* due to their high combined synthesis costs. Finally, *TCCs* of all codes are 0 for eight errors because no ECC could detect errors in such scenarios.

Although works like [2, 7] use a similar method to compare ECCs, we comprehend that it is a qualitative metric without physical meaning. Consequently, we decide to normalize each parameter of Eq. 37 by its maximum value regarding all codes. Additionally, one can notice that correction and detection rates reduce drastically with the errors increase, but the synthesis values remain the same, and the designer can be interested in evaluating the best code for each error scenarios.

Figure 20 shows a new proposal of *TCC* that normalizes the values according to the most efficient code for each error scenario. This perspective of *TCC* highlights that although a code could have a low absolute *TCC*, comparatively it is the best one for dealing with MCUs for that error scenario; besides, in the case of the newest CMOS technologies employed in critical applications, the designers are interested in codes for dealing MCUs with several errors.

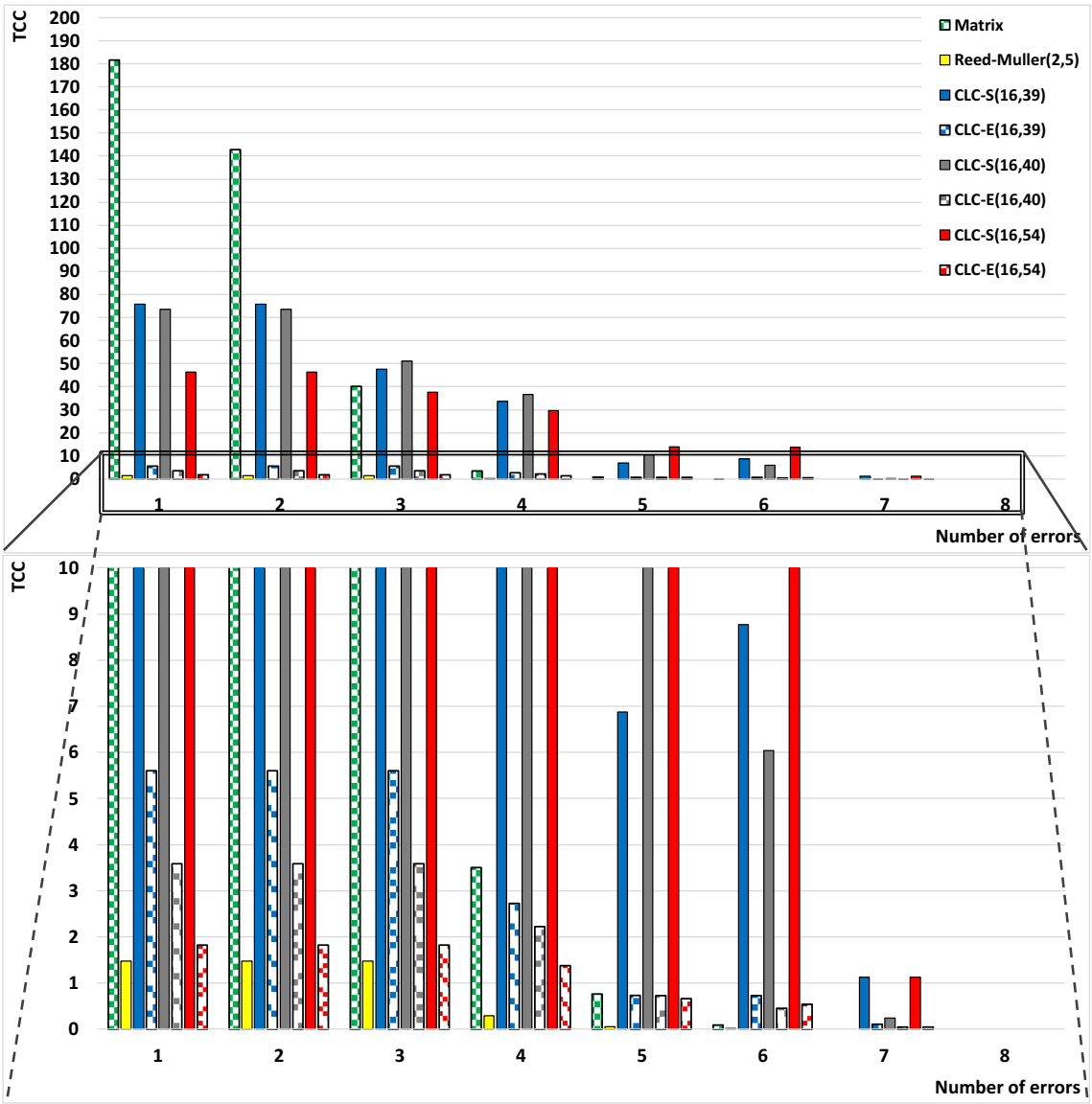
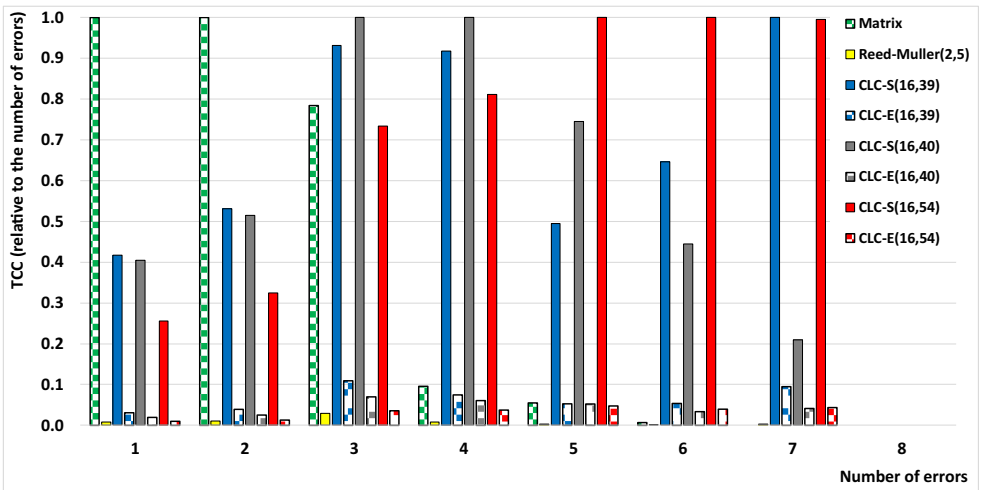


Fig. 19 TCC of all coding methods produced by Eq. 37

Fig. 20 *TCC* normalized for each error scenario



Regarding scenarios with a single error, all codes achieve 100% of error detection and correction; consequently, the *TCC* metric of all codes differs only on the combined synthesis cost, whose lowest value is achieved by the Matrix code implementation. Nevertheless, both CLC-S(16,39) and CLC-S(16,40) have low combined synthesis cost attributing them the second and third position.

Similar *TCC* results are obtained with two-error scenarios; however, the Matrix code does not reach 100% of error detection and correction; consequently, all other codes performs proportionally much better. The inefficacy of Matrix in detecting and correcting more than three errors reduces its *TCC* significantly. Besides, Reed-Muller(2,5) and the extended CLCs, which are the most efficacious correcting and detecting codes, reach very insignificant *TCC* results due to their high combined synthesis costs.

The high redundancy of CLC-S(16,54) makes increasing its *TCC* (proportionally to the other codes) with the increase of errors until reaching seven errors; then, its efficacy is exceeded by CLC-S(16,39), which is a lightweight code.

CLC-S(16,39) is more efficient than CLC-S(16,40) for the first two scenarios, due to the overall cost of the first is smaller than the second. CLC-S(16,39) is also more effective from six and seven errors scenario, these cases are justified by the correction and detection rates being better for such scenarios. These results indicate the benefits of modifying the format of the code to explore more parity bits instead of check bits. *TCC* of the CLC-E formats surpassed Reed-Muller(2,5) in all scenarios, due to the higher error coverage achieved by the CLC-E algorithm with lower cost. However, when compared to Matrix, the extended formats only exceeds it from five to seven errors. In all scenarios, CLC-S formats achieved better results than the CLC-E formats. Finally, except for single and double error scenarios (and in triple error scenario for CLC-S (16,54)), the CLC-S formats surpass all other codes.

10 Conclusion

This paper proposes a broader study of the CLC [3], an error detection/correction code for memory arrays subjected to MCUs. CLC can operate in standard (CLC-S) or extended (CLC-E) mode using parity and extended Hamming code to improve the detection and correction of various MCU patterns. CLC-E differs from CLC-S by applying twice the correction procedure to handle complex patterns of MCUs.

We evaluated in standard and extended modes the CLC (16,40) proposed in [3] with two other 2D schemes - CLC (16,39) and CLC(16,54). The first divides a 16-bit word into 2 rows of 8 bits, and the second divides the same word into 8 rows of 2 bits. The CLC codes presents scalability for different data bits sizes, diminishing the redundancy addition rate as well as the data bits applied grows. For instance, for a 128-bit data, the redundancy addition is near to 72.6%.

All CLC versions surpassed Matrix and Reed-Muller(2,5) codes in most of the error scenarios evaluated for correction and detection analysis. The experimental results show the extended modes of CLC correct more than 20% of errors compared to the standard mode for scenarios from 3 to 5 errors, in average. Although Reed-Muller(2,5) and all CLC formats have achieved 100% of error detection for 1 to 4 error scenarios, the Reed-Muller(2,5) error correction capability drops dramatically compared to the CLC codes. This conclusion is crucial to deal with MCUs, which are very likely to occur as technology dramatically scales down.

We also performed an MTTF analysis to evaluate the impact of each code in being susceptible to errors during the memory lifetime.

Finally, we proposed *TCC*, a new design metric employed to all ECCs to evaluate the tradeoff of high detection and correction error efficacy versus the decoder implementation costs (area consumption, power dissipation, and delay). *TCC* allows us concluding CLC is a well suitable option for embedded systems designed to operate in harsh environments, where MCUs are strongly expected to occur, allied with an excellent balance between error coverage and synthesis cost.

References

1. Argyrides C, Zarandi H, Pradhan D (2007) Matrix codes: multiple bit upsets tolerant method for SRAM memories. In: Proceedings of IEEE international symposium on defect and fault-tolerance in VLSI systems (DFT), pp 340–348
2. Argyrides C, Pradhan D, Kocak T (2011) Matrix codes for reliable and cost efficient memory chips. IEEE Trans Very Large Scale Integr VLSI Syst 19(3):420–428
3. Castro H, Silveira J, Coelho A, Silva F, Magalhães P, de Lima Jr O (2016) A correction code for multiple cells upsets in memory devices for space applications. In: Proceedings of IEEE NEWCAS, pp 1–6
4. Chugg A, Moutrie M, Jones R (2004) Broadening of the variance of the number of upsets in a read-cycle by MBUs. IEEE Trans Nucl Sci 51(6):3701–3707
5. Ferreyra P, Marques C, Ferreyra R, Gaspar J (2005) 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 52(1):494–500
6. Gherman V, Evain S, Auzanneau F, Bonhomme Y (2011) Programmable extended SEC-DED codes for memory errors. In: Proceedings of IEEE VLSI test symposium (VTS), pp 140–145
7. Guo J, Xiao L, Mao Z, Zhao Q (2014) Enhanced memory reliability against multiple cells upsets using decimal matrix code. IEEE Trans Very Large Scale Integr VLSI Syst 22(1):127–135
8. Hazucha P, Svensson C (2000) Impact of CMOS technology scaling on the atmospheric neutron soft error rate. IEEE Trans Nucl Sci 47(6):2586–2594
9. Hentschke R, Marques F, Lima F, Carro L, Susin A, Reis R (2002) Analyzing area and performance penalty of protecting different

digital modules with hamming code and triple modular redundancy. In: Proceedings of symposium on integrated circuits and systems design (SBCCI), pp 95–100
10. Hsiao M, Bossen D, Chien R (1970) Orthogonal Latin Square codes. IBM J Res Dev 4(4):390–394
11. Hsu C-L, Ho M-H, Lin C-F (2009) Novel built-in current-sensor-based testing scheme for CMOS integrated circuits. IEEE Trans Instrum Meas 58(7):2196–2208
12. Ibe E, Taniguchi H, Yahagi Y, Shimbo K, Toba T (2010) Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule. IEEE Trans Electron Devices 57(7):1527–1538
13. LaBel K, Barnes C, Marshall C, Johnston A, Reed R, Barth J, Seidleck C, Kayali S, O'Bryan M (2000) A roadmap for NASA's radiation effects research in emerging microelectronics and photonics. In: Proceedings of IEEE aerospace conference, pp 535–545
14. Liu S, Xiao Y, Mao G (2016) Extend orthogonal Latin Square codes for 32 bit data protection in memory applications. Microelectron Reliab 63:278–283
15. Lue S, Huang H (2017) Adaptive block-based refresh techniques for mitigation of data retention faults and reduction of refresh power. In: Proceedings of Test Conference in Asia (ITC-Asia), pp 101–106
16. Maestro J, Reviriego P (2008) Study of the effects of MBUs on the reliability of a 150 nm SRAM device. In: Proceedings of ACM/IEEE design automation conference (DAC), pp 930–935
17. Miremadi S, Zarandi H (2005) Reliability of protected techniques used in fault-tolerant cache memories. In: Proceedings of IEEE annual Canadian conference on electrical and computer engineering (CCECE), pp 776–779
18. Radaelli D, Puchner H, Wong S, Daniel S (2005) Investigation of multibit upsets in a 150 nm technology SRAM device. IEEE Trans Nucl Sci 52(6):2433–2437
19. Reviriego P, Argyrides C, Maestro J, Pradhan D (2011) Improving memory reliability against soft errors using block parity. IEEE Trans Nucl Sci 58(3):981–986
20. Reviriego P, Pontarelli S, Maestro J, Ottavi M (2013) Reducing the cost of single error correction with parity sharing. IEEE Trans Device Mater Reliab 13(3):420–422
21. Saleh A, Serrano J, Patel J (1990) Reliability of scrubbing recovery techniques for memory systems. IEEE Trans Reliab 39(1):114–122
22. Sanchez-Macian A, Reviriego P, Maestro J (2014) Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement. IEEE Trans Device Mater Reliab 14(1):574–576
23. Satoh S, Tosaka Y, Wender A (2000) Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAM's. IEEE Electron Device Lett 21(6):310–312
24. Toro D, Arzel M, Seguin F, Jezequel M (2014) Soft error detection and correction technique for radiation hardening based on C-element and BICS. IEEE Trans Circuits Syst Express Briefs 61 (12):952–956
25. Vargas F, Nicolaidis M (1994) SEU-tolerant SRAM design based on current monitoring. In: Proceedings of IEEE international symposium on fault tolerant computing, pp 106–115
26. Varghese B, Sreelal S, Vinod P, Krishnan A (2013) Multiple bit error correction for high data rate aerospace applications. In: Proceedings of IEEE conference on information communication technologies (ICT), pp 1086–1090

Felipe Gaspar Alan e Silva is a master's student at Federal University of Ceará since 2016. He achieved his bachelor degree in Teleinformatics Engineering in 2016 at Federal University of Ceará. His research interests are in Fault Tolerance, Error Correction Codes, Embedded Systems, Networks on Chip, Software & Hardware Testing, Real Time Systems. Currently, he works as a researcher at the Computer Systems Engineering Laboratory (LESC).

Ricardo Jardel Nunes da Silveira is an Assistant Professor at Teleinformatics Department (DETI), Federal University of Ceará (UFC), Brazil, since 2011. He received a Master degree in Teleinformatics Engineering from Federal University of Ceará, Brazil, in 2008. His research interests are in the areas of embedded systems on digital circuits, computer architecture, security, fault tolerance, and real-time systems. Currently, he is a doctorate student in DETI/UFC and participates in two research projects and he works as a researcher at the Computer Systems Engineering Laboratory (LESC).

Jarbas Aryel Nunes Silveira is an Adjunct Professor at Teleinformatics Department (DETI), Federal University of Ceará (UFC), Brazil, since 2009. He received his Ph.D. in Teleinformatics Engineering from Federal University of Ceará, Brazil, in 2015. His research interests are in the areas of embedded systems on digital circuits, computer architecture, on-chip communication architectures, fault tolerance, and real-time systems. Currently, he participates in three research projects and he works as a researcher at the Engineering Laboratory Computer Systems (LESC).

César Augusto Missio Marcon is professor at Graduate Program in Computer Science of Pontifical Catholic University of Rio Grande do Sul, Brazil, since 1995. He received his Ph.D. in Computer Science from Federal University of Rio Grande do Sul, Brazil, in 2005. Professor Marcon is member of IEEE. He has more than 100 papers published in prestigious journals and conference proceedings. Since 2005, prof. Marcon coordinated nine research projects in areas of telecom, and telemedicine. He is advisor of MsC. and Ph.D. graduate students in areas of NoC based MPSoC and Wireless Sensor Networks.

Fabian Vargas obtained his Ph.D. Degree in Microelectronics from the Institut National Polytechnique de Grenoble (INPG), France, in 1995. At present, he is Full Professor at the Catholic University (PUCRS) in Porto Alegre, Brazil. He holds 6 BR and international patents and published over 200 refereed papers. Prof. Vargas is associate researcher of the BR National Science Foundation since 1996. F. Vargas has experience in Computer Science, focusing on Computer Systems Architecture, acting on the following topics: fault-tolerant systems design for critical applications, design of on-chip sensors for reliability insurance, design for electromagnetic compatibility/ionizing radiation tolerance and on-line testing. Prof. Vargas is a Golden Core Member of the IEEE Computer Society and Senior Member of the IEEE.

Otávio Alcântara de Lima Jr. is an assistant professor at the Federal Institute of Technology of Ceara, Brazil. He received a Ph.D. in Microelectronics in 2015 from Jean Monnet University - France. He received his M.Sc. degree in Teleinformatics Engineering in 2012 from Federal University of Ceara, Brazil. His research interest includes embedded systems, MPSoCs and NoCs.