CLC-A: An Adaptive Implementation of the Column Line Code (CLC) ECC

[1] Felipe Silva, [1] Adahil Muniz, [1] Jarbas Silveira, [2] César Marcon

[1] Departamento de Engenharia de Teleinformática, Campus do Pici, Universidade Federal do Ceará, Fortaleza (UFC)
[2]Pontifical Catholic University of Rio Grande do Sul - PUCRS

{gaspar, adahil, jarbas}@lesc.ufc.br; cesar.marcon@pucrs.br;

Abstract–Column-Line-Code (CLC) is an Error Correction Code (ECC) designed to correct multiple errors in memory devices for critical applications. CLC has originally two decoder modes: CLC Standard (CLC-S) and CLC Extended (CLC-E). CLC-E presents higher efficacy in correcting multiple errors, although consuming more area, dissipating more power, and presenting higher delay than CLC-S, which impacts the overall system performance. This paper proposes the CLC Adaptive (CLC-A), an alternative CLC mode that introduces a Syndrome Analyzer circuit, allowing the system to decide if a second error checking is required to correct the data. The experimental results show that CLC-A reaches higher error correction rates than CLC-S and similar values of CLC-E. Besides, CLC-A obtained nearly half the overhead in area and power and about 27% less delay when compared to CLC-E, which is a significant reduction in the synthesis cost.

Keywords — Error Correction Code (ECC), Multiple Cell Upset (MCU), Transient Error, Reliable Memory, Critical Application.

I. INTRODUCTION

The reliability of electronic devices operating in an environment with high electromagnetic radiation is harmed by the incidence of Single Event Effects (SEE) [1]. This issue is potentiated as the size of these devices decreases [2][3][4]. Memories are electronic devices where the occurrence of SEEs corrupts stored data due to the incidence of transient faults [2].

Usually, Error Correction Codes (ECCs) are applied in memories to increase device reliability. For instance, ECCs with Single Error Correction – Double Error Detection (SEC-DED) capacity are widely applied in computational systems to protect against single-bit errors [5]. However, as the technology scales down, the sensitivity of integrated circuits to transient faults has grown significantly. The Multiple Cell Upsets (MCU) incidence in memories turned into a major problem for critical application [6][7], causing bitflips in nearby cells [8], which the standard utilization of SEC-DEDs cannot guarantee the data reliability. The combination of interleaving patterns is a strategy applied to explore SEC-DEDs use [9], enabling the correction of some MCU patterns by splitting multiple error patterns into single errors. However, this approach showed a lack of performance for Ternary Content Addressable Memories (TCAM), due to the tight physical coupling of memory [10]. ECCs with higher correction capacity are applied (e.g., Reed-Muller) [11] to solve this issue; but, these codes may increase the decoding complexity significantly, requiring more computational resources that increase the power dissipation and delay, decreasing the system performance. Thus, the ECC overall cost must be carefully analyzed to find a proper balance between computational cost and correction efficiency [12].

978-1-7281-9625-1/20/$31.00 ©2020 IEEE

Column-Line-Code (CLC) [13] is a 2D structured ECC that combines Extended Hamming [14] and Parity. Compared to well-known ECCs as Matrix [15] and Reed-Muller [11], CLC presented the best tradeoff between error correction capacity and synthesis cost. The main features of CLC are detailed in Section II.

In [16], the authors presented new CLC versions that explore different structures of Extended Hamming for performing tradeoffs on synthesis cost reduction or error correction rate, allowing the developers more flexibility to choose the most suitable CLC design for their applications. The main drawback of the CLC versions proposed until now is the number of redundant bits, particularly for a 16-bit word, although the authors [16] shown that this drawback may be reduced as the data word applied increases. Furthermore, these same authors also presented a new correction mode named CLC Extended (CLC-E), which increases the error correction capacity of CLC due to the double-checking of the codeword, without adding redundant bits. To make the distinction of this new correction mode easy, they called Standard CLC (CLS-S) the previous CLC version. Despite that, the cost increase of the CLC-E decoder is considerable, which motivates this paper proposal.

This paper proposes the CLC Adaptive (CLC-A), an adaptive version of the CLC decoder that decides in execution time whether the codeword is corrected through standard or extended correction mode. This feature allows CLC-A to present an equivalent correction capability and notable lower synthesis cost compared to CLC-E. Section III presents the concepts and structure of CLC-A.

Sections IV and V show the experiments and results of the CLC-A correction capability and synthesis cost, respectively. In the experiments, we compared the proposed approach with the CLC-S and CLC-E. Section VI finishes this work presenting the main conclusions achieved.

II. CLC FUNDAMENTALS

CLC is an ECC for correcting multiple adjacent errors through a logic that combines the principles of Extended Hamming and Parity. Figure 1 exemplifies a CLC code for 16-bit data that generates a 39-bit codeword.

The works [13][16] describe that the 39-bit codeword of CLC is divided into four sets of bits: 16 Data bits (D_0 ... D_{15}); eight Check bits generated through Hamming (C_0 ... C_7); 13 Parity bits (Pc_0 ... Pc_{13}); and two Extended Hamming Parity bits (Pr_0 and Pr_1). Each line encodes eight D bits producing four C bits and one Pr bit. The last line is composed only by Pc bits.

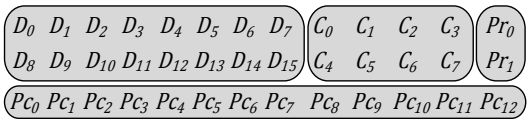

Figure 1. CLC(16,39) – 16-bit data [16].

The CLC algorithm begins estimating the syndrome of the redundant bits (C, Pc, and Pr), determining whether errors were detected. The syndrome estimation is the XOR ($\oplus$) operation between the stored value of a redundant bit (e.g., C) with the recalculated value of the same redundant bit (e.g., RC), whose value may differ with the occurrence of errors in a codeword. Equation 1 presents the operation to estimate a Check bit syndrome (SC). The same logic is applied to all other redundant bits.

$$SC = C \oplus RC \quad (1)$$

Then, through a correction table and syndrome analysis, every line is checked in the 2D structure for correct errors. Depending on the results of the syndrome calculation, an action of the correction table is taken. Table 1 presents the correction table of CLC.

Table 1. Correction table [16].

SC	*SPr*	*SPc*	Correction/Detection	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 / Parity

Table 1 allows correcting from one to three errors in the same line of CLC. Although CLC can cover up to 2-bit adjacent errors of the codeword, some 3-bit error patterns are not covered by the correction table, requiring data retransmission. Figure 2 presents an example of a 3-bit error pattern not corrected by the CLC-S algorithm.

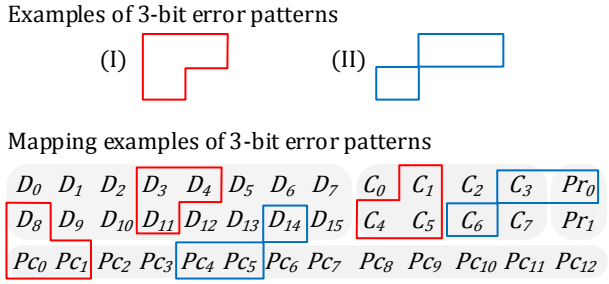

Figure 2. Examples of 3-bit error patterns and some mappings of these patterns on CLC(16, 39) that the CLC-S algorithm cannot correct.

To deal with these error patterns, [16] proposed the CLC Extended mode (CLC-E), which checks each line of the codeword twice, being a 2-step correction process. The CLC-E can correct all 3-bit adjacent errors, increasing the correction efficacy of the CLC Standard mode (CLC-S).

Despite the increase in correction efficacy of the CLC-E, particularly for 3-bit errors, the synthesis cost increased significantly. Table 2 compares the area consumption, total power dissipation (dynamic and leakage), and delay of the encoders implementing the CLC-S and CLC-E algorithms.

Table 2. CLC decoders synthesis results for 16-bit word versions.

ECC	Area		Power		Delay	
	(µm²)	(%)	(mW)	(%)	(ns)	(%)
CLC-S(16,39)	1194	100	0.085	100	1.37	100
CLC-E(16,39)	2612	218	0.268	315	2.55	186

CLC-E is a brute force algorithm that implements a double check in all error situations, even when there are only single errors. Thus, the implementation of CLC-E can imply a considerable impact on the system performance, without providing a significant increase in the correction efficacy. This performance cost is the main reason that boosts this work, which explores a more efficient error correction algorithm with a lower implementation cost.

III. CLC Adaptive Architecture

This paper proposes CLC-A, which is an Adaptive decoder for CLC that in execution time determines if the error correction procedure must use the standard or extended mode; therefore, enabling a tradeoff between correction efficacy (i.e., dealing with more complex error patterns) and system performance (i.e., delay and power consumption increases).

A. The 32-Bit CLC Structure and Computation

We expanded the CLC version described in Section II to a 32-bit data, following the description presented in [16], which results in a 65-bit codeword – CLC(32,65). This increase in the data field shows that CLC scaling implies a Redundant Rate (RR) reduction; for instance, while CLC(16, 39) has RR=(39-16)/16=1.4375, CLC(32,65) has RR=(65-32)/65=1.03125. Figure 3 illustrates the CLC(32,65) codeword structure.

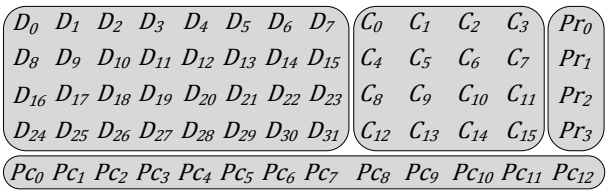

Figure 3. CLC (32,65) codeword structure.

The redundant bits of CLC(32,65) are computed according to Equations 2 to 9. Equations 2 to 5 describe how to calculate the C_q bits, which q represents the check bit index. The Pr_q bits are calculated as described in Equation 6. Equations from 7 to 9 describe how to calculate the Pc_q bits.

$$C_q = D_{8q} \oplus D_{8q+1} \oplus D_{8q+3} \oplus D_{8q+4} \oplus D_{8q+6} \quad \forall\, q \in \{0, 4, 8, 12\} \quad (2)$$

$$C_q = D_{8q} \oplus D_{8q+2} \oplus D_{8q+3} \oplus D_{8q+5} \oplus D_{8q+6} \quad \forall\, q \in \{1, 5, 9, 13\} \quad (3)$$

$$C_q = D_{8q+1} \oplus D_{8q+2} \oplus D_{8q+3} \oplus D_{8q+7} \quad \forall\, q \in \{2, 6, 10, 14\} \quad (4)$$

$$C_q = D_{8q+4} \oplus D_{8q+5} \oplus D_{8q+6} \oplus D_{8q+7} \quad \forall\, q \in \{3, 7, 11, 15\} \quad (5)$$

$$Pr_q = D_{8q} \oplus \ldots \oplus D_{8q+7} \oplus C_{4q} \oplus \ldots \oplus C_{4q+3} \quad \forall\, q \in \{0, 1, 2, 3\} \quad (6)$$

$$Pc_q = D_q \oplus D_{q+8} \oplus D_{q+16} \oplus D_{q+24} \quad \forall\, 0 \le q \le 7 \quad (7)$$

$$Pc_{q+8} = C_q \oplus C_{q+4} \oplus C_{q+8} \oplus C_{q+12} \quad \forall\, 0 \le q \le 3 \quad (8)$$

$$Pc_{12} = Pr_0 \oplus Pr_1 \oplus Pr_2 \oplus Pr_3 \quad (9)$$

B. CLC Adaptive Architecture

Figure 4 displays that CLC-A architecture encompasses two modules: Adaptive Control and Sub-Decoder. Adaptive Control is a Finite State Machine (FSM) that controls the execution of the Sub-Decoder, which analyzes the syndrome of redundant bits, deciding whether the error to be corrected requires a single-checking (similar to CLC-S) or double-checking (similar to CLC-E).

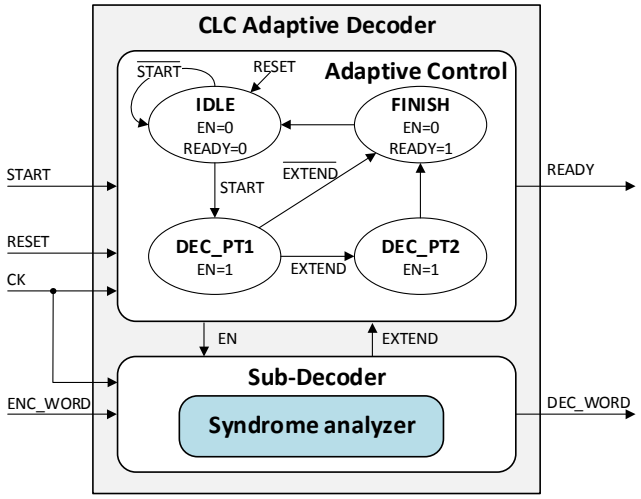

Figure 4. A high-level description of the CLC-A architecture.

The FSM of the Adaptive Control remains in the IDLE state while the command START=0; in this state, the CLC-A Decoder emits READY=0, informing that the decoder does not have a decoded word (DEC_WORD) ready to be read. When the CLC-A decoder is commanded to decode the input coded word (ENC_WORD), the FSM goes to the DEC_PT1 state, placing EN=1; consequently, commanding the Sub-decoder module to perform the first correction procedure, correcting simple errors detected in the ENC_WORD.

Sub-decoder implements the Syndrome Analyzer circuit, which assesses whether ENC_WORD has more than one line with errors and whether at least one line has a double error (as shown in Figure 2). In this case, the Syndrome Analyzer circuit informs the Adaptive Control, setting EXTEND=1, that it is necessary to perform a second correction procedure. On the subsequent clock pulse, the Adaptive Control reads the EXTEND signal, causing the FSM to go to the DEC_PT2 state. This last state generates an additional EN=1 command, which commands the Sub-decoder to perform another step of the CLC algorithm to correct new simple errors detected in ENC_WORD (see Section III.C for better understanding the double-error correction procedure).

If the Syndrome Analyzer circuit does not detect the need for a second correction step, Sub-decoder sets EXTEND=0, informing to the Adaptive Control that the ENC_WORD decoding is finished. Note that this does not necessarily imply that all errors have been corrected, but that there is no benefit in performing a second correction step. Subsequently, the Adaptive Control FSM goes to the FINISH state for a clock cycle, causing the CLC-A Decoder to emit READY=1, which means that the DEC_WORD is ready to be read. Finally, the Adaptive Control FSM goes to the IDLE state, waiting for decoding the next ENC_WORD.

C. Example of CLC-A Double Correction Procedure

The first CLC-A correction procedure can potentially correct a simple error, which is part of a double error, using a simple error correction method. Therefore, when applying a second correction procedure, the previous double error becomes a simple error to be corrected.

Figure 5 shows an example of a CLC(32,65) double correction procedure. In the first step, the algorithm detects that the codeword has a double error in the first line, but cannot correct this error, because the second line has an error, not allowing to know the double-error position. Also, the algorithm detects that the error on the second line is in D_{11}, which can be corrected with Hamming. The Syndrome Analyzer circuit reports the existence of this double error, requiring one more correction step. In the second step, the double error in the first line is detected again, but since the second line has no errors (corrected in the first step), the CLC algorithm can correct bits D_3 and D_4 through the column parity bits Pc_3 and Pc_4.

(I)

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	C_0	C_1	C_2	C_3	Pr_0
D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	C_4	C_5	C_6	C_7	Pr_1
D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	D_{21}	D_{22}	D_{23}	C_8	C_9	C_{10}	C_{11}	Pr_2
D_{24}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}	D_{30}	D_{31}	C_{12}	C_{13}	C_{14}	C_{15}	Pr_3
Pc_0	Pc_1	Pc_2	Pc_3	Pc_4	Pc_5	Pc_6	Pc_7	Pc_8	Pc_9	Pc_{10}	Pc_{11}	Pc_{12}

(II)

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	C_0	C_1	C_2	C_3	Pr_0
D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	C_4	C_5	C_6	C_7	Pr_1
D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	D_{21}	D_{22}	D_{23}	C_8	C_9	C_{10}	C_{11}	Pr_2
D_{24}	D_{25}	D_{26}	D_{27}	D_{28}	D_{29}	D_{30}	D_{31}	C_{12}	C_{13}	C_{14}	C_{15}	Pr_3
Pc_0	Pc_1	Pc_2	Pc_3	Pc_4	Pc_5	Pc_6	Pc_7	Pc_8	Pc_9	Pc_{10}	Pc_{11}	Pc_{12}

Figure 5. Example of a double-error correction procedure.

IV. Correction Analysis

This section presents the correction results achieved by CLC-A to be compared with CLC-S and CLC-E. The main objective of this analysis is to validate the proposed CLC-A, verifying the effectiveness of the Syndrome Analyzer to perform the second correction step. Therefore, the initial assumption is that CLC-A has at least correction efficacy equals to CLC-S, although this experiment also checks whether the correction rates of the proposed decoder match or surpass the correction rates of CLC-E.

We developed an error correction experiment using the Verilog language to evaluate CLC-A, CLC-S, and CLC-E, which were designed regarding 32-bit data. The script that controls the experiment and collects the data was built in SystemVerilog.

Along with the ECC Encoder and Decoders, we developed an error injector module that emulates MCUs in a memory device, resembling the experiments presented in [17][18]. The error injector module generates error patterns employing a one-bit distance between bitflips, which are the most probable situation, as stated by [18]. Figure 6 presents the four experiment phases.

This experiment, which is performed for CLC-S, CLC-E, and CLC-A, starts with a 32-bit word (Input data) encoded by the CLC encoder to produce the corresponding 65-bit codeword (I). Note that the CLC encoder is the same for all CLC types since the differences are only in the decoding algorithms. During Step II, an error injector module inserts random combinations from 1 to 8 errors into the codeword, emulating bitflips in a memory device. Next, the codeword with errors is inserted into the ECC decoder, producing the decoded 32-bit word (Output data) (III). Finally, the Input data is compared to the Output data. If both data are equal, then the error was successfully corrected; otherwise, it represents an uncorrected situation.

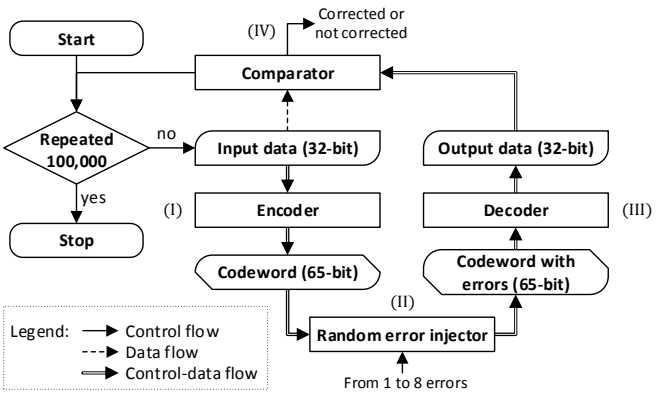

Figure 6. Steps of the error coverage experiment, which is performed for CLC-S, CLC-E, and CLC-A.

To evaluate the correction capacity of the ECCs from one to eight errors, we designed a set of ten thousand error patterns for each error scenario, which were used by the error injector to extract the correction efficacy of each ECCs. Figure 7 illustrates the results of this experiment, displaying that CLC-A achieved higher error coverage than CLC-S, which was expected since the decoder performs the extended correction when triggered by the Syndrome Analyzer. Besides, CLC-A also had similar correction results to CLC-E, meaning that the Syndrome Analyzer implemented inside the CLC-A decoder optimizes its correction performance, in most of the situations. However, CLC-E performs the extended correction mode for all error scenarios, and CLC-A uses this mode only when necessary, which represents substantial rationing of resources.

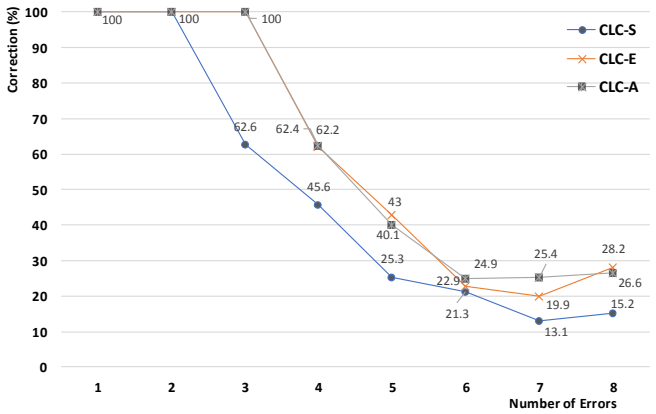

Figure 7. Error correction rates of CLC-S, CLC-E, and CLC-A.

V. Synthesis Results

This section presents the synthesis results of CLC-A, CLC-S, and CLC-E. We do not evaluate the encoder synthesis since for these three ECCs because the encoding module remains the same. Furthermore, we implemented a Coverage per Decoder Cost (CDC) analysis, which estimates the approximate tradeoff between error coverage and synthesis cost of the ECCs analyzed. CDC also was applied considering singly area, power, and delay.

A. Synthesis of the Decoders

The CLCs decoders were implemented in Verilog and synthesized with Cadence's *G.E.N.U.S* for 65 CMOS technology; the results are presented in Table 3.

Table 3. Decoder synthesis results.

ECC	Area		Power		Delay	
	(µm²)	(%)	(mW)	(%)	(ns)	(%)
CLC-A(32,65)	2463	102.5	0.423	116.8	2.0	142.8
CLC-S(32,65)	2403	100.0	0.362	100.0	1.4	100.0
CLC-E(32,65)	5283	219.8	1.040	287.2	2.7	192.8

Table 3 shows that CLC-A synthesis costs are very close to CLC-S, the FSM structure of the Adaptive Control, and Synthesis analyzer additions are the reasons for this minor difference. The delay, which increased by nearly 42.5% compared to CLC-S, was the most affected result due to the addition of the CLC-A features. On the other hand, the rise in area and power was nearly marginal, 2.5% and 16.8%, respectively.

When compared to CLC-E, CLC-A has a considerably lower cost. The area, power, and delay of CLC-A are respectively 53.4%, 59.4%, and 26% smaller than the CLC-E. These values are explained by the fact that the CLC-E decoder duplicates the CLC-S decoder execution, with the correction procedure occurring in the same clock pulse. CLC-A may require two clock pulses to perform the entire execution, impacting the CLC-A decoder delay.

B. Correction per Decoder Cost (CDC) Analysis

We performed the CDC analysis adapting the metric presented in [19], which considered the coverage of the proposed ECCs per the cost of the encoder and decoder. Our experiment does not consider the encoder cost since it is the same applied for all CLCs. Equation 10 presents the $Circuit\ Cost$ of each ECC decoder, which is the product of area consumption, power dissipation, and delay extracted from the synthesis results.

$$Circuit\ Cost(circuit) = Area \times Power \times Delay \qquad (10)$$

Equation 11 computes CDC as the division of the $Correction\ Rate$ (achieved by each ECC - Figure 7) per the $Circuit\ Cost$.

$$CDC = \frac{Correction\ Rate}{Circuit\ Cost} \qquad (11)$$

Figure 8 presents the CDCs for each CLC mode considering all error scenarios – the results are normalized, and the number 1.0 stands for the best tradeoff.

In general, CLC-S achieved the best CDC results; however, the gap to CLC-A from three to eight errors was marginal. This small gap can be explained by the fact that the correction results of CLC-A are considerably higher than CLC-S, compensating the higher CLC-A synthesis cost. Besides, for all the error patterns analyzed, the high correction

capacity of the CLC-E cannot counterbalance its huge synthesis cost, making CLC-E the code with the lower CDC.

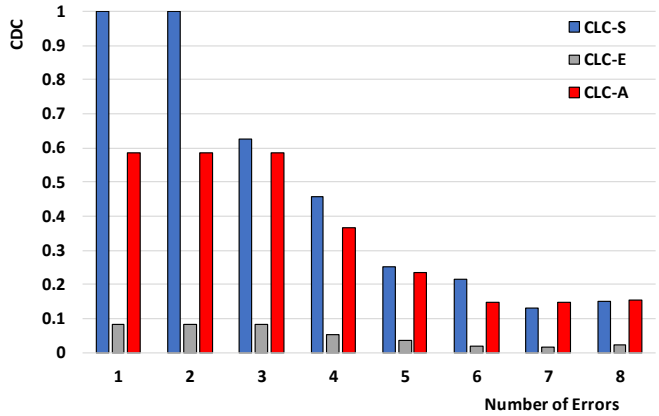

Figure 8. CDC results of CLC-S, CLC-E, and CLC-A.

We also expanded the analysis of the CDC by considering the isolated impact of each synthesis result. This experiment produced Figure 9 to Figure 11.

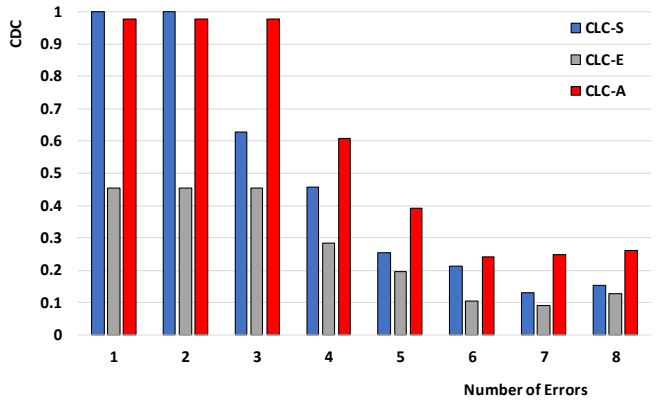

Figure 9. CDC considering only area consumption as the synthesis cost.

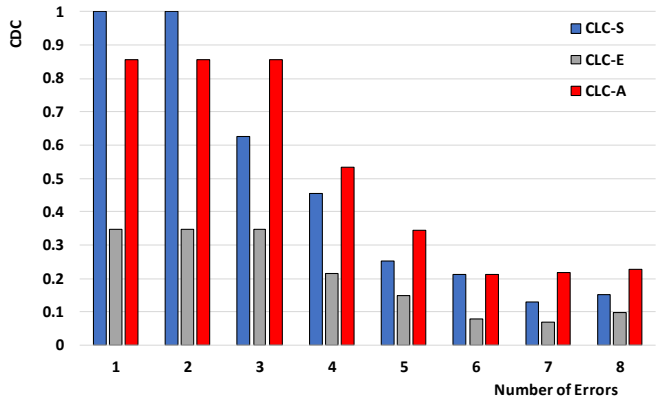

Figure 10. CDC considering only power dissipation as the synthesis cost.

On the one hand, Figure 9 and Figure 10 illustrate that, despite the first two error scenarios, CLC-A presented in most scenarios the best tradeoff between error correction capacity and area consumption or energy dissipation. On the other hand, Figure 11 shows that the delay significantly impacts the gain ratio between CLC-S and CLC-A, since the CLC-A decoder has almost 50% more delay than the CLC-S decoder. However, we stress that a rigorous delay analysis should require a more complex experiment, involving the implementation of the codes in an embedded system. Even because, to run the extended mode, the CLC-A requires two clock pulses, which were not considered for this experiment.

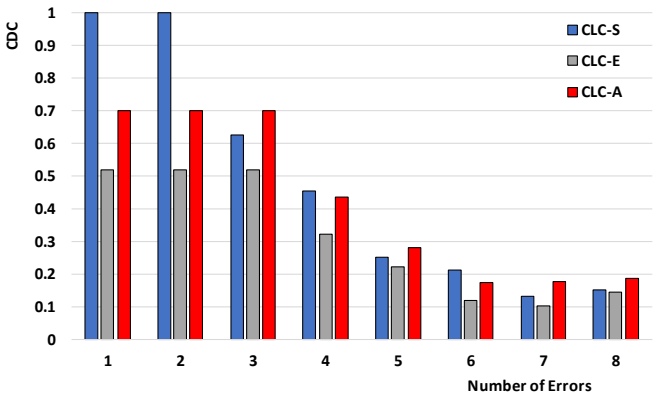

Figure 11. CDC considering only delay as the synthesis cost.

VI. CONCLUSION

This paper presented CLC Adaptive (CLC-A), an adaptive CLC mode that allows the decoder to decide whether to use the standard or extended correction modes during the execution time. The CLC-A decoder encompasses an FSM and a Syndrome Analyzer to perform its adaptive behaviour. The first one acts as a decoder controller, receiving the decision made by the Syndrome Analyzer to activate or not the extended correction mode. The decoder needs two clock pulses to perform the full codeword correction.

The proposed decoder not only maintains the high error coverage provided by the CLC-E but also considerably reduces the cost to implement a more robust version.

The CDC results showed that CLC-A represented a considerable upgrade compared to CLC-E, reducing the tradeoff gap against CLC-S and, particularly for the area- and power-driven systems, CLC-A presents the best tradeoff results for most of the error scenarios. The CDC results show that CLC-A provides a substantial improvement compared to CLC-E; besides, particularly for systems with requirements of reducing area consumption and energy dissipation, CLC-A presents the best tradeoffs for most error scenarios.

VII. ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

REFERENCES

[1] R. Baumann. **Radiation-induced soft errors in advanced semiconductor technologies**. *IEEE Transactions on Device and Materials Reliability*. v. 5, pp. 301-316, Sep. 2005.

[2] M. Nicolaidis. **Soft Errors in Modern Electronic Systems**, Springer Science & Business Media, Nov. 2012, 318pp.

[3] 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.

[4] A. Olazabal, J. Pleite. **Multiple Cell Upsets inside aircrafts. New Fault-Tolerant Architecture**. *IEEE Transactions on Aerospace and Electronic Systems*. v. 55, n. 1, pp. 332-342, Feb. 2019.

[5] V. Gherman, S. Evain, F. Auzanneau, Y. Bonhomme. **Programmable Extended SEC-DED Codes for Memory Errors**. *In Proceedings of IEEE VLSI Test Symposium (VTS)*, pp. 140-145, 2011.

[6] G. Tsiligiannis, L. Dilillo, A. Bosio, P. Girard, S. Pravossoudovitch, A. Todri, A. Virazel, H. Puchner, C. Frost, F. Wrobel, F. Saigné. **Multiple**

Cell Upset Classification in Commercial SRAMs. *IEEE Transactions on Nuclear Science*, v. 61, n. 4, pp. 1747-1754, Aug. 2014.

[7] 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.

[8] S. Satoh, Y. Tosaka, A. Wender. **Geometric Effect of Multiple-Bit Soft Errors Induced by Cosmic Ray Neutrons on DRAM's**. *IEEE Electron Device Letters*, v. 21, n. 6, pp. 310-312, Jun. 2000.

[9] S. Baeg, S. Wen, R. Wong. **SRAM Interleaving Distance Selection with a Soft Error Failure Model**. *IEEE Transaction on Nuclear Science*, v. 56, n. 4, pp. 2111-2118, Aug. 2009.

[10] S. Baeg, S. Wen, R. Wong. **Minimizing Soft Errors in TCAM Devices: A Probabilistic Approach to Determining Scrubbing Intervals**. *IEEE Transactions on Circuits Systems I: Regular Papers*, v. 57, n. 4, pp. 814-822, Apr. 2010.

[11] E. Weiss. **Generalized Reed-Muller codes**. *Information and Control*, v. 5, n. 3, pp. 213-222, Sep. 1962.

[12] 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**. *In Proceedings of Symposium on Integrated Circuits and Systems Design (SBCCI)*, pp. 95-100, 2002.

[13] H. Castro, J. Silveira, A. Coelho, F. Silva, P. Magalhães, O. Lima. **A Correction Code for Multiple Cells Upsets in Memory Devices for Space**. *In Proceedings of IEEE International New Circuits and Systems Conference (NEWCAS),* pp. 1-6, 2016.

[14] R. Hamming, **Error Detecting and Error Correcting Codes**, *The Bell System Technical Journal*, v. 29, pp. 147-160, Apr. 1950.

[15] C. Argyrides, H. Zarandi, D. Pradhan. **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, 2007.

[16] F. Silva, J. Silveira, J. Silveira, C. Marcon, F. Vargas, O. Lima Jr. **An Extensible Code for Correcting Multiple Cell Upset in Memory Arrays**. *Journal of Electronic Testing*, v. 34, n. 4, pp. 417-433, Jul. 2018.

[17] P. Rao, M. Ebrahimi, R.Seyyedi, M. Tahoori. **Protecting SRAM-based FPGAs against multiple bit upsets using erasure codes.** *In Proceedings of ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6, 2014.

[18] C. Ogden, M. Mascagni. **The Impact of Soft Error Event Topography on the Reliability of Computer Memories**. *IEEE Transactions on Reliability*, v. 66, n. 4, pp. 966-979, Dec. 2017.

[19] F. Silva, W. Freitas, J. Silveira, C. Marcon, F. Vargas. **Extended Matrix Region Selection Code: An ECC for adjacent Multiple Cell Upset in Memory Arrays**. *Microelectronics Reliability*, v. 106, pp. 1-9, Mar. 2020.