FEDERAL UNIVERSITY OF CEARA
CENTER OF TECHNOLOGY
TELEINFORMATICS ENGINEERING DEPARTMENT
TELEINFORMATICS ENGINEERING POSTGRADUATE PROGRAM

FELIPE GASPAR ALAN E SILVA

ERROR CORRECTION CODES BASED ON REGION SELECTION CODES

FORTALEZA

2023

FELIPE GASPAR ALAN E SILVA

ERROR CORRECTION CODES BASED ON REGION SELECTION CODES

Thesis presented to the Postgraduate Program in Teleinformatics Engineering at the Federal University of Ceará, as a partial requirement for obtaining the title of Doctor in Teleinformatics Engineering. Area of concentration: Signals and Systems.

Advisor: Prof. Dr. Jarbas Aryel Nunes da Silveira
Co-advisor: Prof. Dr. César Augusto Missio Marcon

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58e Silva, Felipe Gaspar Alan e.
Error Correction Codes Based on Region Selection Codes / Felipe Gaspar Alan e Silva. – 2023.
88 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2023.
Orientação: Prof. Dr. Jarbas Aryel Nunes da Silveira.
Coorientação: Prof. Dr. César Augusto Missio Marcon.

1. Error Correction Codes. 2. Fault Tolerance. 3. Memories Reliability. 4. Single Event Effects. I. Título.
CDD 621.38

FELIPE GASPAR ALAN E SILVA

ERROR CORRECTION CODES BASED ON REGION SELECTION CODES

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Doutor em Engenharia de Teleinformática. Área de concentração: Sinais e Sistemas.

Aprovada em:___/___/_____.

ACKNOWLEDGMENT

First to God, for my life and for being my strength in the most difficult times.

To my family, especially: To my parents João and Christina for the love and effort with which they raised me and allowed me to be where I am today, and my sisters Juliana and Marília, my first teachers. I would also like to thank my friends and co-workers for their love and support.

To professors Jarbas and César Macon, for their friendship, commitment and opportunities offered throughout my postgraduate career. You serve as an example to me and will always have my gratitude and respect.

To the professors participating in the committee for their time and suggestions provided for this work.

To LESC, which served as my second home for almost 10 years and where I graduated as a researcher and professional. Thanks to all teachers, engineers, assistants, colleagues, and friends for so many lessons.

To the DETI professors who continue their quest for excellence and pass it on to their students.

Finally, to CAPES, for the financial support with the maintenance of the scholarship.

“Let me tell you something you already know. The world ain’t all sunshine and rainbows. It’s a very mean and nasty place and I don’t care how tough you are it will beat you to your knees and keep you there permanently if you let it. You, me, or nobody is gonna hit as hard as life. But it ain’t about how hard ya hit. It’s about how hard you can get hit and keep moving forward. How much you can take and keep moving forward. That’s how winning is done!”
Rocky Balboa.

RESUMO

A continua diminuição em escala dos transistores impulsionou o advento de sistemas intra-chip, em inglês *System-on-Chips* (SoCs), permitindo a inserção de mais lógica computacional em um circuito integrado (CI), aumentando sua capacidade de processamento. No entanto, esse processo tornou os dispositivos eletrônicos mais susceptíveis a efeitos externos, principalmente radiação. Dentre os CIs modernos, memórias (e.g., SRAM e DRAM) são bastante susceptíveis a radiação, sendo inversão de múltiplos bits (MBUs) e corrupção da informação os tipos de falhas mais reincidentes. Códigos corretores de erros, em inglês *Error Correction Codes* (ECCs), têm sido amplamente utilizados para aumentar a confiabilidade dos dados armazenados em memórias. ECCs com formato linear e matricial se destacam em memórias de duas dimensões. Códigos de formato linear têm uma dimensão e são usados para proteger um conjunto de dados armazenados em uma dimensão (e.g. um endereço de memória). Códigos de formato matricial têm duas dimensões, protegendo uma matriz de dados (e.g. mais de um endereço de memória). Esta tese teve como objetivo desenvolver um conjunto de ECCs baseado em códigos de seleção de região, em inglês *Region Selection Code* (RSC), que consiste em separar os dados de memória em regiões, e através de operações lógicas e passos simples realizar correções de MBUs. O primeiro ECC desenvolvido foi o *Matrix Region Selection Code* (MRSC), um código de formato matricial capaz de corrigir erros adjacentes, conseguindo 100% de eficiêcia em até 2-bit erros. Outras abordagens foram desenvolvidas: *Extended Matrix Region Selection Code* (eMRSC) e *Triple Burst Error Corrector - Region Selection Code* (TBEC-RSC). O eMRSC é uma versão também de formato matricial, mas com dois formatos: uma com maior capacidade de correção de erros e outra com menor número de bits de redundância. O TBEC-RSC é uma versão proposta em formato linear, conseguindo corrigir até 3-bit erros em rajada. Todas as propostas foram comparadas com outros trabalhos da área considerando capacidade de correção, confiabilidade e custo de síntese. Por fim, os resultados coletados dos experimentos mostraram que os ECCs baseados na lógica RSC apresentaram excelentes taxas de correção de erros e de confiabilidade (e.g., TBEC-RSC corrigiu aproximadamente 40% de 8-bit burst erros) e também se caracterizaram por ter baixo custo de síntese (e.g., MRSC consumiu 91,2% menos potência que o código Reed-Muller), fazendo-os ter a melhor relação de cobertura de correção por custo de síntese dentre os ECCs comparados.

Palavras-chave: códigos corretores de erros; tolerância a falhas, confiabilidade de memórias, eventos de efeito único.

ABSTRACT

The continuous decrease in the scale of transistors spurred the advent of System-on-Chips (SoCs), allowing the insertion of more computational logic in an integrated circuit (IC), increasing its processing capacity and functionalities. However, this process made electronic devices more susceptible to external effects, mainly radiation. Among modern ICs, memory circuits (e.g., SRAM and DRAM) are very susceptible to radiation effects, and may present different types of failures, with multiple-bit inversion (MBUs) and stored information corruption being the most recurrent types. In this context, Error Correction Codes (ECCs) have been widely used to increase the reliability of data stored in memory. ECCs with linear and matrix format excel in two-dimensional memories. Linear format codes have one dimension and are used to protect a dataset in this single dimension. Array format codes are two-dimensional, protecting an array of data. This thesis had the goal of developing a set of ECCs based on Region Selection Code (RSC), which consists of separating memory data into regions, and through logical operations and simple steps to perform MBU corrections. The first ECC developed was the Matrix Region Selection Code (MRSC), a matrix format code that was developed for adjacent error correction, managing to correct 100% of 2-bit errors. Two other extension approaches were developed: Extended Matrix Region Selection Code (eMRSC) and Triple Burst Error Corrector - Region Selection Code (TBEC-RSC). eMRSC is also an extension of the matrix format, but with two configurations: one with greater error correction capability and another with a smaller number of redundancy bits. TBEC-RSC is a proposed extension to linear format, able to correct up to 3-bit burst errors. All proposals were compared with other works in the area considering correction capacity, reliability and synthesis cost. Finally, the results collected from experiments showed that ECCs based on RSC logic showed excellent error correction capability and good reliability rates (e.g., TBEC-RSC corrected approximately 40% of 8-bit burst errors), also characterized by low synthesis cost (e.g., MRSC consumed 91.2% less power than the Reed-Muller code), which made them have the best ratio of correction coverage per synthesis cost among the compared ECCs.

Keywords: error correction codes; fault tolerance, memory reliability, single event effects.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
2-D	2-Dimensional
CLC	Column Line Code
COTS	Commercial off-the-shelf
CMOS	Complementary Metal-Oxide-Silicon
CSC	Coverage per Cost
DMC	Decimal Matrix Code
DRAM	Dynamic Random Access Memory
ECC	Error Correction Code
XOR	Exclusive Or
eMRSC	Extended Matrix Region Selection Code
FIT	Failures in Time
FUEC	Flexible Unequal Control
IC	Integrated Circuit
MRSC	Matrix Region Selection Code
MTTF	Mean Time To Failure
MBU	Multiple Bit Upset
MCU	Multiple Cell Upset
NASA	National Aeronautics and Space Administration
NMOS	nFET Metal-Oxide-Silicon
OLS	Orthogonal Latin Square
RSC	Region Selection Code
SOI	Silicon-on-Insulator
SBU	Single Bit Upset
SEC-DED	Single Error Corrector-Double Error Detector
SEB	Single Event Burnout
SEDR	Single Event Dielectric Rupture
SEE	Single Event Effect
SEFI	Single Event Functional Interrupt
SEGR	Single Event Gate Rupture
SEHE	Single Event Hard Error

SEL	Single Event Latch-up
SESB	Single Event Snapback
SET	Single Event Transient
SEU	Single Event Upset
SER	Soft Error Rate
SRAM	Static Random Access Memory
SOC	System-on-Chip
TBEC–RSC	Triple Burst Error Corrector – Region Selection Code
TMR	Triple Modular Redundancy

TABLE OF CONTENTS

1 INTRODUCTION

Modern and complex Integrated Circuits (ICs) are characterized by high integration level and processing power, enabling them to enclose complex modules implemented by numerous and more extensive discrete components in the past. These enhancements emerged driven by the shrinkage of the CMOS technology to the nanoscale era, which brought together the increase in Single Event Effects (SEEs) induced by the impact of charged particles (Ferreyra et al., 2005). Nowadays, circuits are usually composed of memory devices whose occurrence of bitflips can lead to total operation failure; thus, dealing with SEE represents a vital concern for safety-critical applications (BAUMANN et al., 2005).

Bitflips induced by SEE commonly result from Single Bit Upset (SBU) or Multiple Cell Upset (MCU); the last one is the most dangerous for the systems since it produces multiple bitflips in a single event. Several approaches have been considered throughout the years to protect memory devices, although it is common sense that one of the leading solutions is to implement an Error Correction Code (ECC) (GHERMAN et al., 2011). Although the theory of ECC derives from telecommunication channel coding, the concepts are widely applied to the reliability of memory devices.

ECCs stand out from other approaches due to these features:

- Depending on the complexity and robustness of the technique, it may correct a vast amount of MCU patterns.
- Overall, it is a solution that brings low impacts on the systems in terms of re-design and financial cost.

Single Error Correction-Double Error Detection (SEC-DED) is a type of ECC widely applied. However, in aggressive environments such as space, MCUs become more likely to occur with the CMOS technology reduction. Typically, MCUs appear in memories as adjacent or burst error patterns produced by an energized particle that affects a group of neighbor cells, provoking either continuous (adjacent) or spaced (burst) errors by n bits in a word (OLAZABAL et al., 2019)(OGDEN et al., 2017). Since SEC-DEDs do not correct MCUs properly, the study of more efficient ECCs became a design trend.

With this context and due to the continuous enhancement of computer processing, several ECC approaches were introduced aiming to deal with MCUs. One of the oldest codes capable of correcting multiple bitflips is Reed-Muller, which despite being almost 70 years old, they are still relevant in many areas of coding theory (ABBE et al., 2020). However, implementing this code demands a significant quantity of computational resources (e.g., energy and delay) compared with Hamming code, which might not be indicated in systems with computing constraints (VARGHESE et al., 2013).

Some critical applications, such as avionics and space missions, have limited energy available for processing and delay, making the ECC choice not restricted by the correction capacity, but also requires considering the computational cost brought in encoding and decoding operations. In recent years, many authors have proposed ECC approaches to align error correction with low synthesis cost.

One of the first approaches focusing on this trade-off was proposed in Matrix Codes (ARGYRIDES et al., 2007), which organizes bits in a matrix format to correct MCUs in a 32-bits codeword. The code consists 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 compared to a more robust code like Reed-Muller, but with the advantage of consuming less area and energy. One reason for Matrix code achieving lower cost than Reed-Muller is related to the coding strategy applied. Matrix code uses a 2-D structure, while Reed-Muller is a linear block code. On the one hand, Matrix can combine codes with lower correction capability to compose the codeword, enabling error detection by matching the row and column error detectors. On the other hand, Reed-Muller uses a generator matrix and majority logic, which might require several linear equations to determine the correct value for each bit. From Matrix code, several other approaches were developed aiming to mimic and improve the 2-D scheme proposed by the authors, applying new ECCs and coding strategies. For instance, the Column-Line-Code (CLC) utilizes a similar structure implemented by Matrix codes, although improves de error correction by a small increase in the number of redundancy bits and adding a double-checking in the codeword (Extended mode) (SILVA et al., 2018).

In this context, one of the works that compose this thesis is the Matrix Region Selection Code (MRSC), an original, efficient, and low-cost approach capable of correcting adjacent MCUs. The code uses a 2-Dimensional (2D) or matrix format encoding scheme, applying simple logic equations to create the codeword. MRSC brought as its main contribution the Region Selection Algorithm (RSA), which classifies the data bits into regions, and the correction is made by detecting each one has more errors. MRSC is a 2-bit corrector that covers several adjacent error patterns with lower synthesis costs when compared to other robust ECCs (SILVA et al., 2017a).

MRSC is easily extensible to higher data words and this was explored by (SILVA et al., 2020) which it was proposed its extended version eMRSC. In that work, eMRSC was extended for a data word of 32 bits and provides ECC versions: one focused on the higher correction of errors and another with reduced redundancy bits in its encoding.

Despite the advantages brought by MRSC and eMRSC, the implementation of 2-D ECCs might be a concern for standard memory devices. 2-D codewords usually require that their data structure be kept when stored in memory. In that case, the lines of the matrix will be stored sequentially, requiring more write/read operations than the usual for linear format ECCs. With that perspective, an alternative version of MRSC was designed to use a linear format, abstracting the 2-D format. The new approach consisted of making minor changes in the original MRSC and adjusting the bit positioning to extract most of the correction capacity. This new approach corrects not only adjacent errors in the memory cells' neighborhood but also nonconsecutive adjacent errors, which are called burst errors. The code is called Triple Burst Error Corrector – Region Selection Code (TBEC-RSC) since its correction capability is up to three burst errors.

This thesis will present in detail the RSC codes, the encoding and decoding processes, examples of correction, experimentation setup, and results achieved and conclusions. The objectives and main contributions of this thesis are presented as follows.

1.1 THESIS OBJECTIVES AND MAIN CONTRIBUTIONS

This thesis presents the ECCs MRSC, eMRSC, and TBEC-RSC with their core concepts and features, such as codeword structure, encoding, and decoding procedures, with correction examples and known limitations.

The MRSC encodes the data using parity and bit positioning, composing a matrix-like codeword. For decoding it uses the RSA, an original approach that detects the codeword region with more errors and performs the correction. MRSC was first designed for a 16-bit word but is simple to be extended for bigger data sizes (e.g., 32-bit, 64-bit, etc). The extended version of MRSC for a 32-bit word was called eMRSC.

We implemented two versions of eMRSC: eMRSC (32,7,56) and eMRSC (32,3,64). The first one has a lower increase of redundancy and no regression in the correction capability, and the other one has a higher correction capability and no significant impact on synthesis cost.

Both MRSC and eMRSC were designed to deal with adjacent error patterns. Even though this type of error represents the majority of MCUs in memories (OGDEN et al., 2017), burst errors cannot be forgotten. Moreover, those approaches use a matrix format, hardening their utilization in regular memories without extra design steps, which may increase the delay response and, consequently, the data transmission. A burst error corrector and linear RSC version was developed to address these gaps.

The Triple Burst Error Corrector-Region Selection Code (TBEC-RSC) is a linear version that redistributes the MRSC bits to a linear format and implements a few modifications in the coding logic, which allows the ECC to correct burst errors. This version can correct 3-bit burst patterns with 100% efficacy and did not bring significant impacts in terms of algorithm complexity and synthesis cost.

Nonetheless, MRSC, eMRSC, and TBEC-RSC were submitted to experiments that followed the standards applied in publications regarding ECCs for memories, which consisted of Correction coverage analysis, Reliability analysis, and Synthesis cost and trade-off analysis. In those experiments, a set of published ECCs was also evaluated along with the proposed ones by this thesis. The following section presents how this thesis is organized.

1.2 THESIS ORGANIZATION

This thesis is organized as follows. Chapter 2 presents the theoretical foundations that provided context, motivation, and insights for the development of this work, with a brief review of the SEE theory and the concepts of linear and 2D ECCs. Chapter 3 presents a small literature review of the ECCs designed focusing on memory devices. Chapter 4 presents the methodology of MRSC, eMRSC, and TBEC-RSC, describing their features and presenting examples of correction. Chapter 5 presents the experimental details, results, and discussions regarding the proposed ECCs against other approaches. Finally, Chapter 6 depicts the main conclusions achieved with this work and some propositions for future work.

2 THEORETICAL BASIS

This chapter introduces the main concepts that represent the theory basis that motivated the proposal of this thesis: SEE and ECC. Those two usually are studied together in a reliability context, being the first the provoker of failures and the second the one that mitigates potential failures.

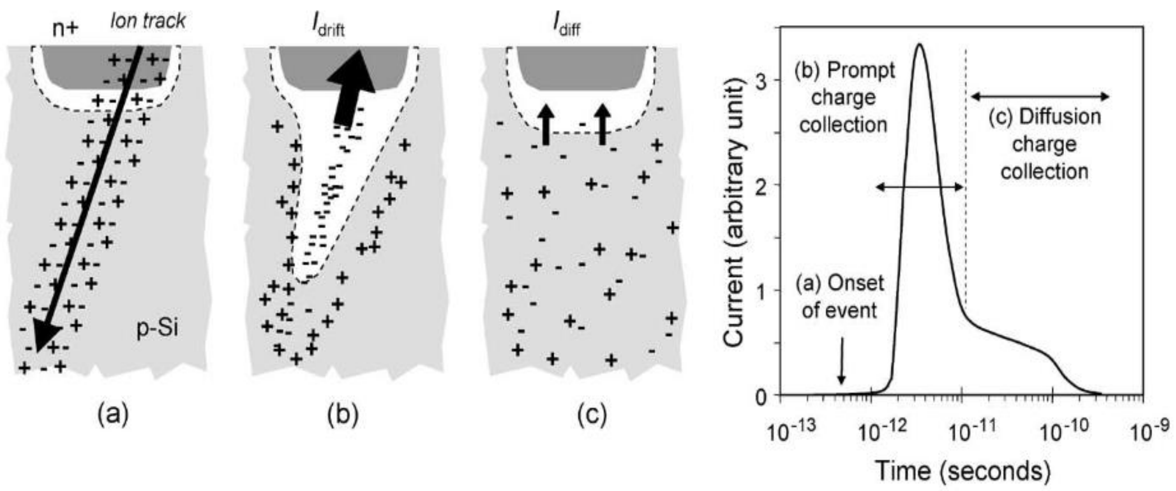
2.1 SINGLE EVENT EFFECT (SEE)

The continuous progress in scaling down integrated circuits provided the evolution and emergence of several electronic devices, making them more fragile to potential failures caused by radiation effects. One of the most common failure-causing effects in electronic circuits is SEE, an electrical disturbance that causes a change in the operation of a circuit. These effects are mainly caused by the passage of charged particles at the junctions of the transistors that compose the circuit. Despite these effects frequently occurring in space applications due to solar radiation and cosmic rays, neutrons produced by the interaction of cosmic rays with our atmosphere also cause SEE in electronic devices used in aviation and the terrestrial layer. This SEE can cause transient and permanent effects (BAUMANN, 2005). The following sections present how SEE can damage electronic circuits, the SEE types, which circuits are most susceptible to these effects, and how they are affected.

2.1.1 SEE: Types and Definitions

A SEE can be caused by the impact of a charged particle on the transistors of an integrated circuit. Regardless of several sources that can produce this effect, the behavior that transistors will present depends directly on how energized the ions will be at the time of impact with the junction. Figure 1 illustrates how a highly charged particle affects the junction of a transistor. Upon crossing the junction, a cylindrical beam of highly charged hole electron pairs is formed (Figure 1a). When the resulting ionization crosses or approaches the depletion region, the unbalance of charges induces the creation of a temporary funnel in which they are rapidly collected by the electric field (Figure 1b). Then the funnel is broken, and the remaining ions are balanced by diffusion (Figure 1c).

Figure 1 – Effect caused by the impact of a charged particle on a transistor.

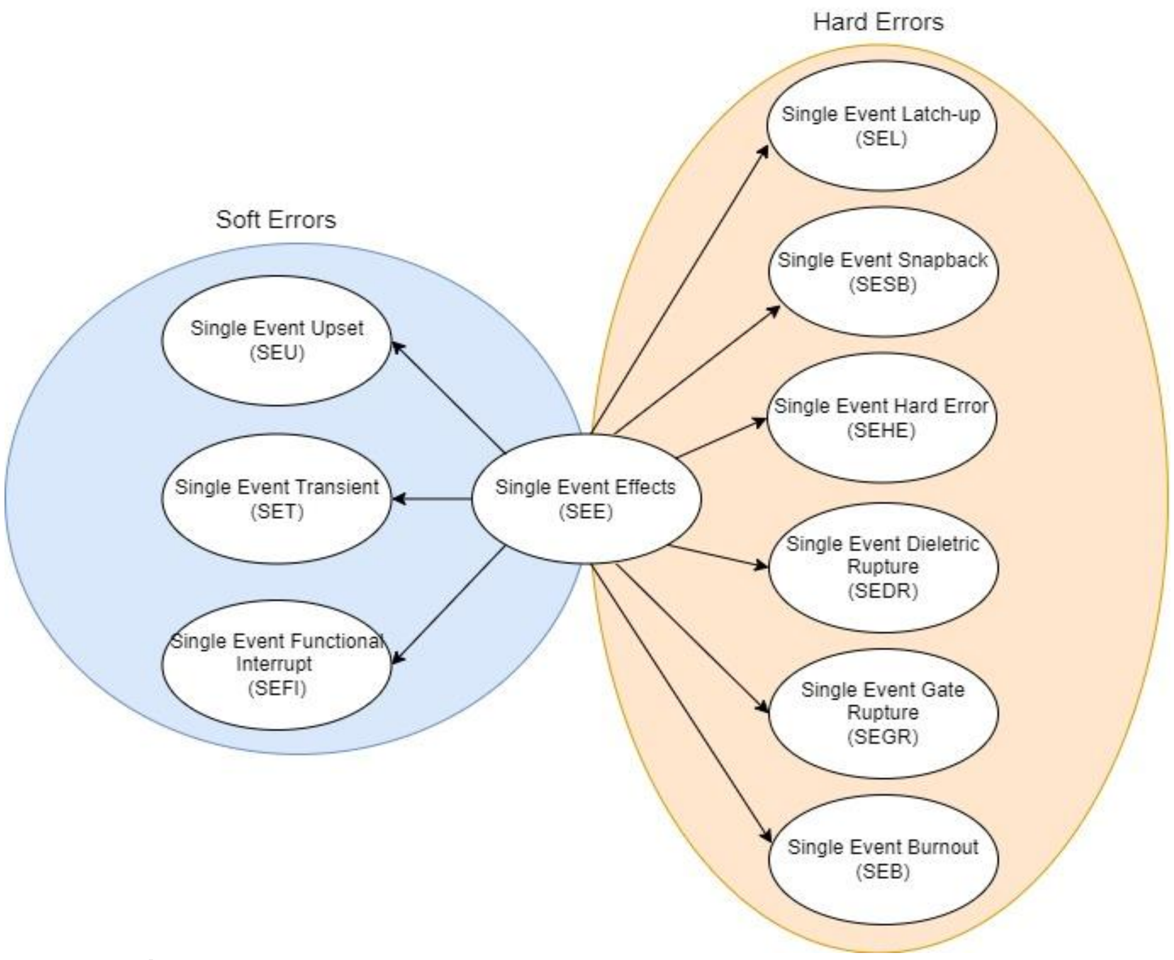


Source: (BAUMMAN, 2005)

A current spike can flip the transistor contents and, depending on the load intensity, can even damage the component. In space missions, these effects must be considered when designing the mission components and payload, particularly as space exploration and research have explored the use of Commercial Off-the-shelf (COTS) components (LABEL, 2018). For example, many COTS components meet performance and temperature specifications, which are critical to performing processes and collecting data. Small satellites such as CubeSats use COTS components and have gained much popularity in universities and space research centers due to their low manufacturing, launching, and operational costs (OGUT et al., 2017). However, radiation effects can trigger the malfunction of these circuits transiently or permanently, and they are presented as follows.

SEE can cause transient (Soft Errors) or permanent (Hard Errors) errors. The type of error caused by a charged particle depends on several factors, such as how much charged the particle is or what region of the integrated circuit is affected. Figure 2 shows the subdivisions of Soft Errors and Hard Errors, along with their definitions (BOSSER, 2017).

Figure 2 – Types of SEE.


Hard Errors
Soft Errors
Single Event Upset (SEU)
Single Event Transient (SET)
Single Event Functional Interrupt (SEFI)
Single Event Effects (SEE)
Single Event Latch-up (SEL)
Single Event Snapback (SESB)
Single Event Hard Error (SEHE)
Single Event Dieletric Rupture (SEDR)
Single Event Gate Rupture (SEGR)
Single Event Burnout (SEB)

Source: Author.

Soft Errors:

- Single Event Upset (SEU) is the change in the state of an electronic device caused by a single charged particle. They are common in critical application memories and can affect one (Single Bit Upset - SBU) or more bits (Multiple Cell Upset - MCU). If the particle affects multiple bits in the same stored information or instruction, the error is called a Multiple Bit Upset (MBU). This effect will be presented in more detail later in this work.
- Single Event Transient (SET) occurs when the movement of charges from a charged particle causes a voltage spike (glitch). External attackers can also provoke this type of error in cyber-attacks. Large companies have conducted studies identifying potential ways to mitigate these situations (BITTNER et al., 2021).

- Single Event Functional Interrupt (SEFI) occurs when a disturbance between the state registers affects the circuit's regular operation, causing it to enter unscheduled operating modes and even causing lockups. SEFIs are types of SEUs, as they occur in control parts of the circuit. Its correction requires a circuit reset.

***Hard Errors*:**

- Single Event Latch-up (SEL) is caused by a highly charged particle that can generate an abnormal operating current in the circuit. The transistor junction might break down if the circuit power is not normalized in time. SEL is a type of permanent failure, usually catastrophic to the system. This effect occurs mainly in Complementary Metal-Oxide-Silicon (CMOS) components; however, circuits with Silicon-on-Insulator (SOI) technology, which insulates the transistor junction with a silicon layer, manage to prevent the occurrence of SEL (KASTENSMIDT, CARRO, REIS, 2006).
- Single Event Snapback (SESB) has a very similar effect to SEL in what it causes in the circuit, but they are more common in nFET Metal-Oxide-Silicon (NMOS) circuits.
- Single Event Hard Error (SEHE) is a fault that corrupts memory cells permanently, ending the capability of having its value changed, even after performing a refresh process – reading part of the memory content and rewriting its stored value.
- Single Event Dielectric Rupture (SEHE) is an event that causes current leakage from the transistor affected by the particle to the primary power circuit of the chip.
- Single Event Burnout (SEB) is caused by a highly charged particle that generates a high current state in a part of the circuit. This effect also has catastrophic consequences for the device.
- Single Event Gate Rupture (SEGR) is an effect where a highly charged particle generates a conductive path through the gate of the transistor. It is usually seen in conjunction with SEB.

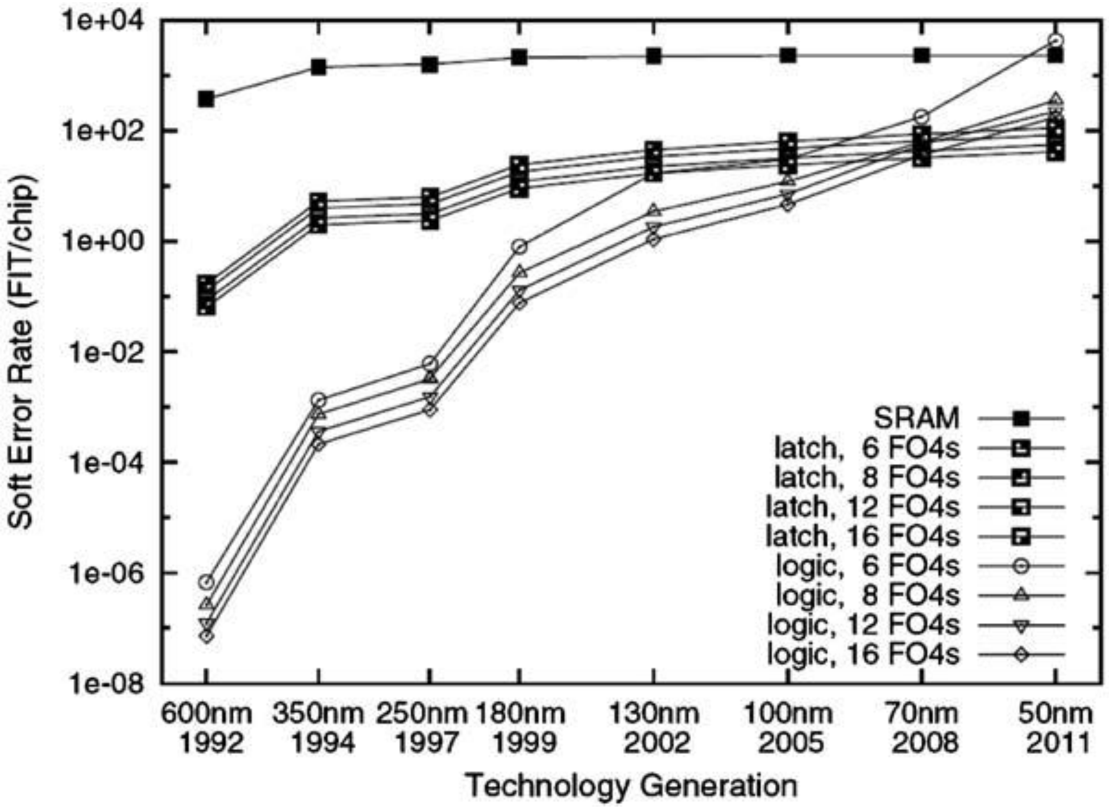
A critical factor for increasing these effects is the technology scale applied on the chips affected by radiation effects. One of the ways to measure how sensitive an integrated circuit is to radiation effects is through the Soft Error Rate (SER), a rate presented in more detail in the following subsection.

2.1.2 Soft Error Rate (SER)

SER is the rate at which a device or system is likely or predicted to suffer from Soft Errors. This rate is measured in units of failures in time (FIT), where 1 FIT means one failure per billion hours of device operation (approximately 114,077 years). The standard range for electronic devices is between 100 and 100,000 FIT (HEIJMEN, 2011).

With the miniaturization of integrated circuits, the SER of devices has grown significantly, making chips with SER above 50,000 FIT quite common. Figure 3 presents a graph of SER evolution for different logics about applied technology. The aggressive growth of the SER of combinational circuits is justified by the reduction of the applied technology that allowed the creation of highly interconnected circuits, improving the processing capacity but making them more fragile.

Figure 3 – SER by technology applied.



Source: (GARG et al., 2009)

Memory circuits, such as SRAM, already had a very high SER from higher technologies. Despite the SER in memories without much evolution in the graph, current applications demand much more storage space to perform operations, such as saving photos, videos, and data in general. Thus, these devices have their probability of failure increased.

Therefore, memories are susceptible circuits to SEE, and these components are crucial for several critical applications. Since one of the objectives of this thesis is to present a new ECC family capable of correcting the occurrence of multiple errors in data stored in these memories, it is necessary to understand the most common types of failures in these circuits.

2.1.3 SEEs in Memories

As discussed, memories are quite sensitive to radiation effects, which makes SEE a constant threat for applications exposed to charged particles, especially space missions. Therefore, designers of such applications need to be aware of the most likely effects on these circuits, their impact on the mission, and how to mitigate these effects, aiming to avoid operational problems. This section presents the most common radiation effects on memories.

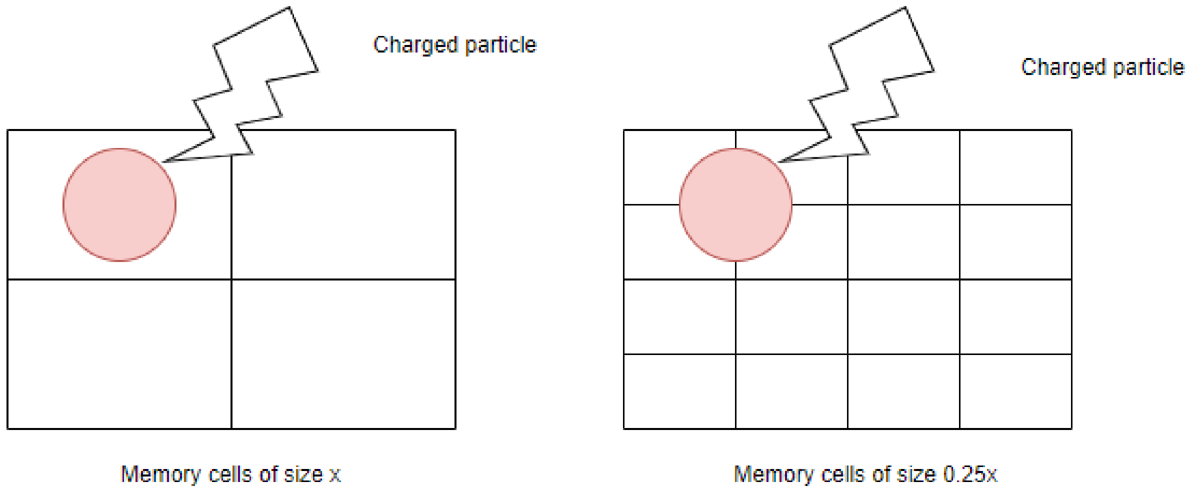
SEE has three types of effects: SEU, SET, and SEFI. Although the last two are also dangerous in memory circuits, the occurrence of SEU and its diffusion in MBU or MCU has been gaining significant relevance in the scientific community (OGDEN, 2017).

The occurrence of SEU in memories is a problem that has been observed since the first space applications (GUENZER et al., 1981). However, for many years, these effects have already been reported in applications at atmospheric levels (JOHANSSON, 1998).

The miniaturization of integrated circuits increased the frequency of these effects. It made them more likely to present SEU's more harmful variation, MCUs that occur since the memory cells started to decrease in size, increasing affected logic by a charged particle. Figure 4 illustrates this situation.

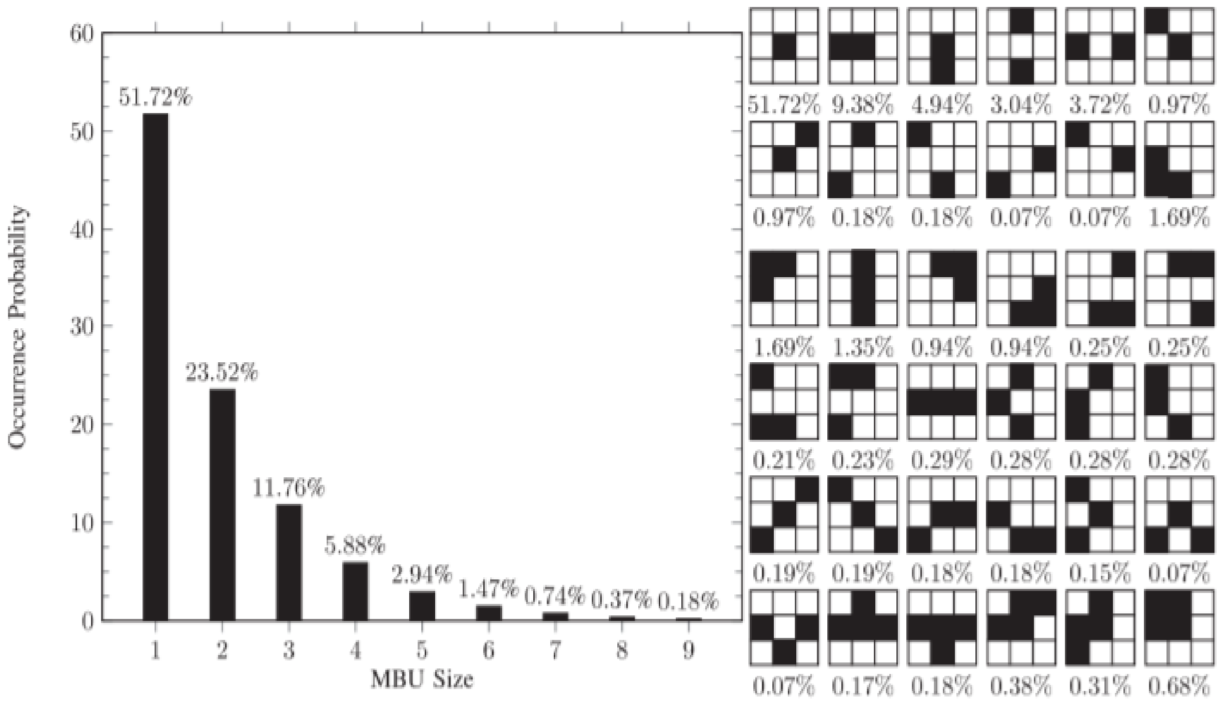
Figure 4 displays that with technology reduction by four times, there will now be four instead of a single affected cell. However, the particle will need more energy to invert the bit stored in each cell, and the energy distribution will be proportional. Figure 5 shows the results of a SEE simulator for memories. Although SBU is predominant, MBUs represented approximately half of the occurrences, with double and triple errors being cases above 10% of all simulations. Another point to note is that this experiment considered the radiation level in the terrestrial environment; thus, space missions expect a high number of MBUs.

Figure 4 – Impact of radiation on miniaturized technology.



Source: Author.

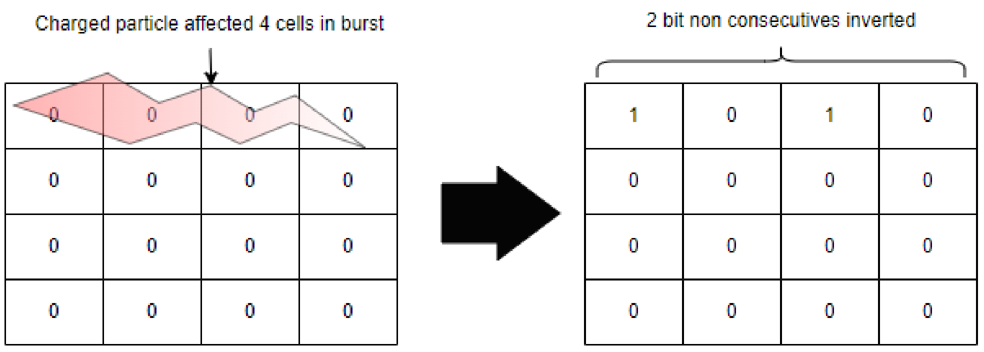
Figure 5 – Most common patterns of MBUs.



Source: (OGDEN, 2017).

Another type of in-memory MBU is called burst errors. These errors affect at least two bits, but not necessarily consecutive ones. This error is more likely to occur when the radiation reaches the memory chip as a non-direct impact but in the form of a burst, passing through multiple bits (GRACIA-MORÁN, 2018). Figure 6 presents an example of a burst error.

Figure 6 – Burst errors in memories.


Charged particle affected 4 cells in burst
2 bit non consecutives inverted
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
1 0 1 0
0 0 0 0
0 0 0 0
0 0 0 0

Source: Author.

We can observe in Figure 6 cells of the same row that had a passage of a charged particle, however, note that the bits were not close. The distance between the first and last inverted bit is called the burst error length, which in the example will equal three.

Therefore, we have seen that systems exposed to radiation have the information transmitted internally constantly threatened and corrupted by charged particles, which may eventually cause system failure if the data read from their memories are not protected or recovered. Note that the biggest problem is not the inversion of the bits themselves, this can happen, and the information returns to the original state before being processed. Otherwise, the system must have the tools to recover the original value of the information or at least inform the system that an error has occurred and that the data to be processed is no longer reliable.

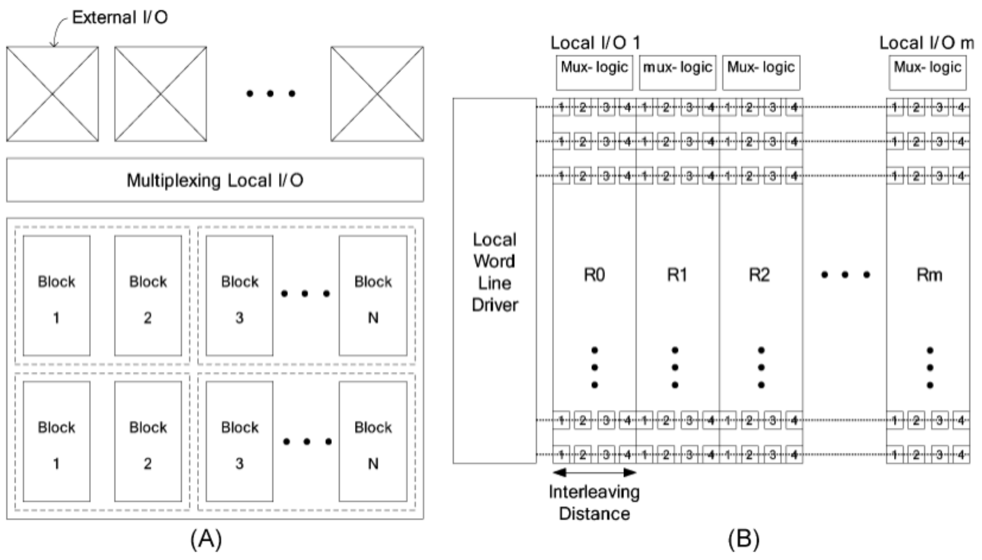
2.1.4 Techniques Utilized to Mitigate SEEs in Memories

Critical applications, such as space and military, are characterized by having a large amount of processing power to carry out and exchange information and processes. Memories in these systems are also essential to maintain performance at a high level. Thus, these must be reliable in most system operations to avoid catastrophic failures. Among numerous studies throughout the years in memory reliability, we highlight bit interleaving, Silicon on Insulator (SOI), triple modular redundancy (TMR), and ECC.

2.1.4.1 Bit Interleaving

Bit interleaving consists of the physical spacing bits of corresponding data in different storage regions. Figure 7 presents an example of interleaving architecture.

Figure 7 – Example of interleaving structure.



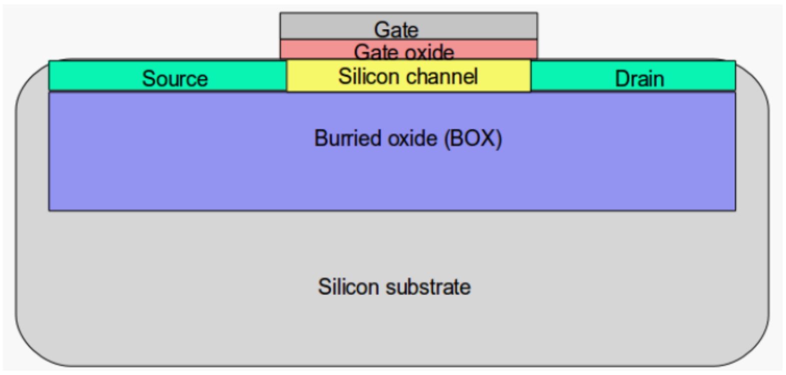
Source: (BAEG et al., 2009)

Figure 7 (B) illustrates that each internal memory Register (R) contains part of the data for each block. The interleaving distance is the number of bits between two bits of the same block. Commonly, bit interleaving is applied with an SEC-DED ECC, allowing the correction of MCUs. However, this technique implementation can lower the performance in read and write operations since the multiplexing in those processes might be time-consuming.

2.1.4.2 Silicon On Insulator (SOI)

The SOI technology is a technique that covers the silicon substrate of a transistor with a layer of SiO2 (Silicon Dioxide), which is also known as the buried oxide (BOX) layer. Figure 8 presents an example of this approach.

Figure 8 – Example of SOI in a transistor.



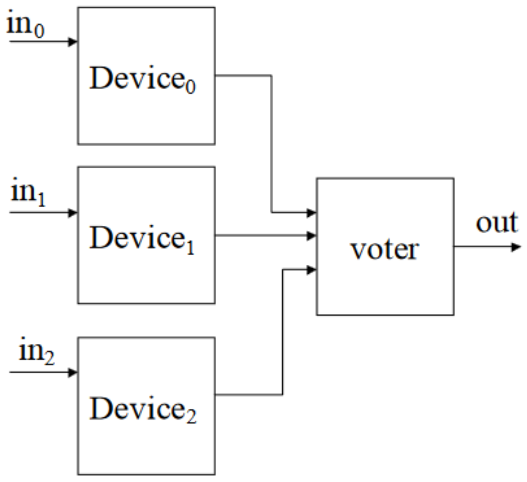
Source: (KUMAR et al., 2018)

SOI effectively avoids the process described in Figure 1 by shielding the silicon substrate from charged particles that might affect the transistor. However, shielding all transistors in a memory device is expensive and not viable for COTS devices.

2.1.4.3 Triple Modular Redundancy (TMR)

TMR is a technique that will triplicate a circuit or device and submit the resultant logic output to a voter to decide the outcome. Figure 9 displays a TMR example.

Figure 9 – Example of TMR implementation.



Source: (KASTENSMIDT et al., 2006)

This approach might apply to situations with transient and permanent faults since the voter will receive three results to decide the correct information to pass. Although, along with the cost increase brought by the implementation of TMR, the voter process may affect the performance. Moreover, if two of three devices present failures, the wrong output might be selected. In aerospace and critical systems, the cost increase of applying TMR is justifiable for components in which the occurrence of a single bit upset might not be acceptable.

2.1.4.4 Error Correction Code (ECC)

In recent years, ECCs have been the primary approach to dealing with bitflips in memories (SUN et al., 2022). They are extensively used to fix bitflips caused by noise interference in communication channels but are also present in several industries, such as aerospace and automotive. One of the main qualities of ECCs is that their use in a system does not significantly increase the cost of the system and they are simple to integrate in systems. For those reasons, ECCs are preferable for critical applications. The following section presents an analysis of the evolution and application of ECCs in critical applications.

2.2 ERROR CORRECTION CODES: LINEAR AND 2-D APPROACHES

In modern computing, it is challenging (and expensive) to completely isolate a system from external effects or anomalies that cause transient or permanent errors in its internal circuits and components. As we saw in the previous chapter, radiation is an agent that can attack electronic devices, especially memories, with variable intensity (transient and permanent) and scalable (one or multiple blocks of logic). When placed in the context that some critical applications are built using COTS, it becomes even more necessary to use strategies to protect the information processed by a system. This section discusses two ECCs formats that the scientific community has extensively studied: Linear Codes and 2-Dimensional (2-D) Codes.

2.2.1 Linear codes

The idea of creating a way to correct errors in data transmission in modern communications systems started with the scientists C. E. Shannon and R. W. Hamming. Shannon developed the mathematical basis that contributed to the development of information theory, making it possible to determine the source's entropy and the communication channel's transmission capacity. This knowledge is still fundamental in transmission channels. Although Shannon's initial contributions were not in developing error correction algorithms, they showed that building a reliable channel is possible.

R. W. Hamming was one of the first to conduct studies to develop the first algorithms to correct transmission channel errors. Hamming's discovery was motivated by persistent glitches in the computers on which he performed his experiments. Thus, the first version of the Hamming code emerged, encoding four data bits into three check bits. Through a truth table, it was possible to identify the position of an error among the seven bits (four of data + three of redundancy) and correct the information, avoiding retransmissions (HAMMING et al., 1950). Hamming's code still serves as a foundation for developing current linear codes. A brief description of the Hamming code based on (NEUBAUER, FREUDENBERGER, and KÜHN, 2007) (MORREIRA and FARREL, 2006) is presented below.

2.1.1.1 Hamming Code

The Hamming code (M,N) was one of the first ECCs developed for application in computer systems, intending to correct simple errors. The following fundamentals characterize a Hamming code (M,N).

- It is a linear code because its coding results from a linear system of equations and a truth table to identify the position of the error.
- M and N represent the number of data and the total bits, respectively.
- N is defined by the equation $N = M + k$; k is the number of Hamming bits.
- A Hamming Code needs to respect the following relations 2.1 e 2.2.

$$2^k \geq k + M + 1 \qquad (2.1)$$

$$2^M \geq \frac{2^N}{N+1} \qquad (2.2)$$

The encoding of the Hamming code (4,7) starts with the calculation of its generating matrix G, which is found in Equation 2.3

$$G = [\, I_{2^k-k-1}\, Q] \qquad (2.3)$$

Where I_{2^k-k-1} is the squared identity matrix of order $2^k - k - 1$, Q is the matrix composed of $2^k - k - 1$ columns with two or more vectors of weight two, and k is the number of redundancy bits. The possible Hamming bit vectors are 000, 001, 010, 011, 100, 101, 110, and 111. The four column vectors that form the matrix Q are 011, 101, 110, and 111.

$$G(4{,}7) \;=\; \begin{matrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}$$

After calculating the matrix G, the encoding process for the Hamming code (4,7) consists of a linear transformation. The encoded vector N(1,7) is obtained through the product between vector M(1,4) and matrix G(4,7). Using a simple example, we have for a $M =$ [1 0 0 0], the codeword is:

$$N \;=\; M\,x\,G \qquad (2.4)$$

$$N = 1\,0\,0\,0 \quad x \quad \begin{matrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}$$

$$N = [1\,0\,0\,0\,0\,1\,1]$$

The Decoding process consists of checking the message for errors; Equation 2.6 describes the syndrome calculation, which is the product between parity matrix H, with Equation 2.5, and the coded message N.

$$H = [Q^T\ I_k] \tag{2.5}$$

Thus, the matrix H is:

$$H = \begin{matrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}$$

$$M_{Decoded} = H\ x\ N^T \tag{2.6}$$

If the result of this product is a null vector, then the transmitted message did not detect errors. Otherwise, the message N shows an error; an operation is performed XOR with the correction vectors (Vc) presented in Table 1. After the error correction performed by Equation 2.7, the message is ready to be transmitted.

Table 1 – Correction position by the syndrome.

$\boldsymbol{H\ x\ N^T}$	$\boldsymbol{Vc}$
[0 0 0]	[0 0 0 0 0 0 0]
[0 0 1]	[0 0 0 0 0 0 1]
[0 1 0]	[0 0 0 0 0 1 0]
[0 1 1]	[1 0 0 0 0 0 0]
[1 0 0]	[0 0 0 0 1 0 0]
[1 0 1]	[0 1 0 0 0 0 0]
[1 1 0]	[0 0 1 0 0 0 0]
[1 1 1]	[0 0 0 1 0 0 0]

Source: Author

$$M = N\,XOR\,Vc \tag{2.7}$$

The Extended Hamming code $(M, N+1)$ is formed by adding a bit calculated from the parity between the bits in the message. Adding parity increases the Hamming distance to 4, which means the code will fix one error and will be able to catch two errors. The check results of the Extended Hamming code $(M, N+1)$ may have the following states shown in Table 2.

Table 2 – Extended Hamming $(M, N+1)$ states.

Syndrome of Hamming bits	**Parity**	**State**
No error	No error	No error
No error	Error	Simple error in Parity
Error	No error	Simple error
Error	Error	Double error

Source: Author.

Starting from the example presented in the previous subsections, the following considerations are made:

- Calculating the parity of the seven bits of M, we found a bit of value 1, that is added at the end of M to compose the message [1 0 0 0 0 1 1 1];
- In the occurrence of two errors in M, e.g. [**0 1** 0 0 0 1 1 1], the parity syndrome is 0, which initially indicates the absence of errors. However, if operation 2.6 is repeated for the first seven bits of M, it is observed that the product is different from 0, and both errors cannot be corrected. Thus, there are errors, but they cannot be corrected.

More than 70 years after its discovery, the Hamming code remains in current microcontrollers as the TMS570LS0432. This high-performance microcontroller can be used in Safety-Critical applications such as the automotive industry. In the TMS570LS0432, Hamming code is used to protect SRAM and Flash memories, using the code's double error detection capability to trigger a system interrupt, warning of an undetectable fault. However, the decrease in chip technology made multiple memory errors much more likely, making the Hamming code a not-so-good option to protect critical application information.

Thanks to the evolution of computing and the decrease in integrated circuits, several new ECC were inserted in the literature. For example, another well-known linear ECC that has been widely used in space missions is the Reed-Muller code. This ECC was key to the National

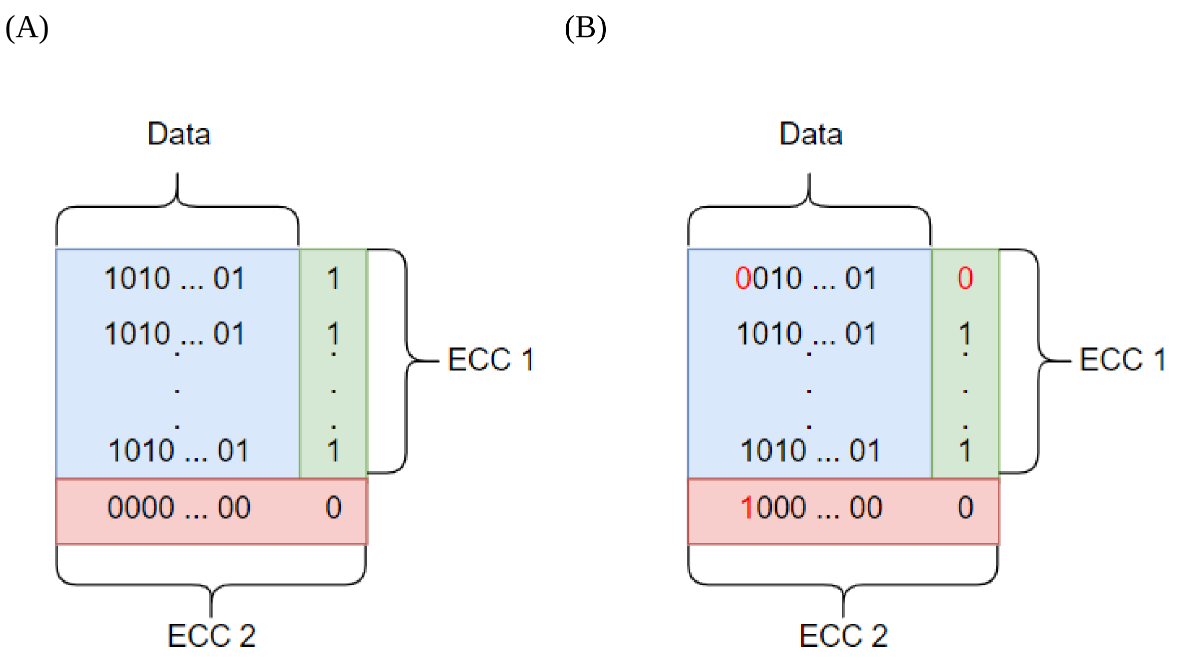
Aeronautics and Space Administration (NASA) Mariner missions, enabling the transmission of the first (colorless) photos of the surface of Mars. However, due to the complexity of this code being much greater than that of the Hamming code, especially in its decoding process, its implementation can be expensive, as shown by (SILVA et al., 2017).

In this context, computational cost and correction capacity became decisive factors in the research and development of an ECC. Thus, the 2-D code development methodology combined high correction capacity with low computational cost.

2.2.2 2-D codes

The development of 2-D codes was a thought-out approach aimed at combining low-cost codes to create a codeword format more resilient to multiple adjacent errors. The matrix format allows ECCs to be used together to detect and correct errors. Figure 10 (A) shows an example of the organization of a typical matrix code.

Figure 10 – 2-D ECC Matrix structure and Error detection in a 2-D ECC



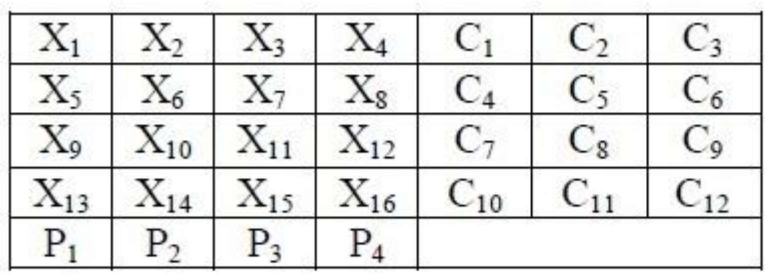
Source: Author.

As in a battleship game, errors are corrected by matching the error detected by ECC 1 and ECC 2. Figure 10 (B) shows an example of when the two ECCs detect an error (the red bit). Once errors are detected, each matrix code's correction procedure is specific. The Matrix code is one of the first ECCs that adopt this approach (ARGYRIDES, 2007).

2.2.2.1 Matrix code

The Matrix code is an ECC that uses Hamming code (4,7) and Parity to form a matrix structure code, as shown in Figure 11.

Figure 11 – Matrix code structure for 16 bits.



X_1	X_2	X_3	X_4	C_1	C_2	C_3
X_5	X_6	X_7	X_8	C_4	C_5	C_6
X_9	X_{10}	X_{11}	X_{12}	C_7	C_8	C_9
X_{13}	X_{14}	X_{15}	X_{16}	C_{10}	C_{11}	C_{12}
P_1	P_2	P_3	P_4			

Source: (ARGYRIDES et al., 2007).

This format was applied to correct multiple error patterns with a lower synthesis cost than more robust codes. The Matrix (16,32) encoding process consists of dividing the 16 bits of data to apply the Hamming code (4,7), which will compose the first four lines of the matrix. The fifth row is formed by calculating the parity of the first four bits of the first four columns of the matrix, forming the 32-bit codeword.

X, C and P are the data, check and parity bits, respectively. Using X = [1000 1010 0101 1111] as a data input, the resulting encoded message is:

$$\begin{array}{ccccccc}
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & & &
\end{array}$$

For decoding, all check bits C and parity P are recalculated, then the syndrome with the original bits presented in equations 2.8 and 2.9 is performed.

$$SC_i = C_i \oplus RC_i \tag{2.8}$$

$$SP_i = P_i \oplus RP_i \tag{2.9}$$

In which RC_i and RP_i are the recalculated bit values for each of the check and parity bits, respectively. Equation 2.10 is applied to correct the wrong bits from the syndrome values.

$$X_{i\_correct} = (X_{i\_wrong} \oplus O_i) \oplus (DED_j \cdot SP_i) \tag{2.10}$$

Where $X_{i\_correct}$ is the value of the corrected bit, X_{i_wrong} represents the bit to be corrected, O_i is the element of the Hamming correction vector presented in Table 1, DED_j represents the double error detection bit on the line j, and S_p is the parity syndrome corresponding to the corrected bit column.

The ECC selection depends on several factors, such as the type of error sought to be corrected, which is more suitable for recent memories, and the implementation cost. Linear codes are generally best for physical memory implementation, inserting a codeword per address. On the other hand, matrix codes tend to have a lower implementation cost than robust linear codes, as they combine low-cost techniques to form an adapted approach with more error coverage. In recent years, several works have been produced to correct adjacent or burst errors, reduce implementation costs, or improve existing versions of traditional codes. The next chapter reviews the Linear and Matrix ECCs focused on memories that served as a reference base to produce this Thesis.

3 REVIEW OF ECCS FOCUSED ON MEMORIES

In recent years, the number of ECCs developed has grown mainly due to the increase in transient failures in computing systems. These failures have generated significant interest in the scientific community to understand these effects better and develop algorithms that combine correction capacity and low computational cost. Another factor that contributed a lot was the evolution of computing itself, allowing the development of highly accurate simulations and managing to measure the correctness of a code in a relatively short term. This chapter reviews publications of ECCs applied in memories between 2016 and 2022 inspired by linear coding or matrix format.

3.1 DISCUSSION AND SEARCH RESULTS

To carry out this research, creating a standard search string involving common keywords from works already known in the ECC area but focused explicitly on memories was necessary. The code format applied was not inserted in this String. However, the primary way to differentiate linear (or based on the linear approach) and matrix codes is that the first needs a truth table and a generator matrix to encode the data. The second is usually the combination of other linear codes to form a matrix of bits. Based on the keywords presented in the works (SILVA, 2018), (ARGYRIDES, 2007) and (GRACIA-MORÁN, 2018), in addition to synonyms and correlated terms, we constructed the standard search string shown in Table 3.

Table 3 – Search string.

Keywords	Synonyms	Correlated terms
“Error Correction Code(s)”	“Error Correcting Codes” and “ECC”	“Fault Detection”, “Fault Correction” and “Parity Check Codes”
Memory (Memories)	-	-
“Decoding” and “Encoding”	-	-
Search String	((Error Correction Codes) OR (Error Correcting Codes) OR “ECC” OR “Fault Correction” OR “Fault Detection” OR (Parity Check Codes)) AND “Memory” AND (“Decoding” OR “Encoding”)	

Source: Author.

The research sources chosen for this search were the most used in the scientific community: Web of Science, IEEExplore, and Scopus. Studies that do not present correction results or synthesis costs using FPGAs were not included, as the most referenced works in the area present both correction data and synthesis evaluation for a given technology (e.g., 65nm, 45nm, or 28nm). Finally, works that were not written in English or not published in international journals or congresses were excluded, as these works hardly have validation by the international scientific community.

Table 4 presents academic works collected from 2016 to 2022. To create it, we seek to classify the works according to the following criteria: Identifier index (ID), Type that the code is based on (Linear or Matrix), Ability to correct multiple errors (Yes or No), error pattern (Burst or Adjacent), number of versions displayed.

Table 4 – State of Art Review 2016-2022.

Reference	ID	Type	Synthesis technology	Error pattern	N° of versions
LIU et al., 2016a	01	Linear	65nm	Burst	2
LIU et al., 2016b	02	Linear	65nm	Burst	4
SILVA et al., 2017a	03	Matrix	45nm	Adjacent	1
LIU et al., 2017	04	Matrix	-	Adjacent	1
REVIRIEGO et al., 2017	05	Linear	65nm	Adjacent	2
TAMBATKAR et al., 2017	06	Matrix	45nm	Adjacent	1
GRACIA-MORÁN et al., 2018	07	Linear	45nm	Burst	3
SILVA et al., 2018	08	Matrix	45nm	Adjacent	3
LIU et al., 2018a	09	Linear	65nm	Burst	1
LIU et al., 2018b	10	Linear	65nm	Burst	3
LI et al., 2018a	11	Linear	65nm	Burst	1
LI et al., 2018b	12	Linear	65nm	Burst	1
LI et al., 2019a	13	Linear	65nm	Burst	3
CHOI et al., 2019	14	Linear	65nm	Adjacent	1
LI et al., 2019b	15	Linear	65nm and 28nm	Adjacent	3
DAS and TOUBA, 2019	16	Linear	45nm	Burst	6
SILVA et al., 2020	17	Matrix	65nm	Adjacent	2
DAS and TOUBA, 2020	18	Linear	45nm	Burst	15
FREITAS et al., 2021	19	Matrix	45nm	Adjacent	1
GARCIA-HERRERO et al., 2021	20	Linear	45nm	Adjacent	2

Source: Author.

As a result of this study, 20 works were collected. Analyzing them in detail, 6 of them use the matrix or 2-D approach. Those ECCs (03,04,06,08,17,19) usually combine two low-cost ECCs (e.g., HAMMING), resembling what Matrix code (ARGYRIDES et al., 2007) has implemented. An example of this case is work number 08, which presents the Column-Line-Code (CLC), which encodes the data bits through extended Hamming and parity in a 2-D structure. Some of them were implemented considering their application in 3D memories, being the case of works 06 and 19. Overall, those approaches usually presented lower synthesis costs when compared with linear codes.

The remaining collected ECCs applied the linear structure. Those codes usually utilize the linear operation concepts employed in Hamming code, although miscellaneous detection and correction approaches were proposed. Some works involved refactoring or inserting additional steps on the Orthogonal Latin Square code (01, 02, 05, 19, 20), a traditional ECC developed on IBM research facility that utilizes majority logic to detect and correct errors (HSIAO et al., 1970).

Other linear codes (07, 09, 10, 11, 12, 13, 15) were proposed by increasing the syndromes that detect unique error patterns, such as those found for Hamming code. For example, work number 07 employs the Flexible Unequal Control (FUEC) methodology, which allows generating matrices capable of producing syndromes to correct multiple errors without adding too many redundancy bits. However, this approach ends up significantly adding to the synthesis cost of the ECC produced.

Although all works manage to correct multiple errors, some of them are limited to specific error patterns, as presented in work 18. Others focused on adjacent error patterns, such as 17 and 19, which most studies chose as the error pattern.

Also, as seen in works 01, 02, 05, 07, 08, 10, 13, 15, 16, 17, 18, and 20, the authors were concerned with proposing alternative versions of codes, varying the number of redundancy bits and correction capacity to provide flexibility in implementing the ECC. However, we also saw that most works have a fixed format; that is, they are presented as a matrix or linear code, not proposing a way to adapt the formatting of the codeword or the detection strategy. This thesis comprises the work presented in 17 (SILVA et al., 2020), which is an extended version of the work 03 (SILVA et al., 2017a), the Matrix Region Selection Code (MRSC). This code presents an ECC matrix approach, subdividing the bits into regions and correcting the regions through low-complexity logic operations. In (SILVA et al., 2020), the two possible versions of the code for 32-bit words are presented, which as contributions, show that the MRSC is a code that, in addition to correcting multiple errors and having low cost, is also expandable, allowing variation of produced redundancy and correction capability. Furthermore, an alternative linear format proposal was developed to correct burst errors without increasing implementation costs or significant changes in the original algorithm.

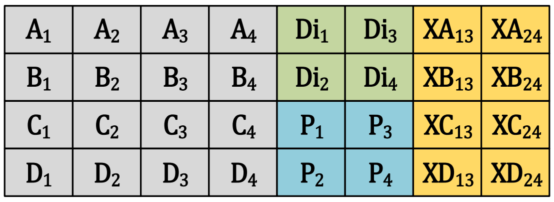
4 METHODOLOGY: MRSC, EMRSC AND TBEC-RSC

The MRSC (SILVA et al., 2017a) ECC was developed with a matrix structure combining two correction techniques, enabling us to correct multiple adjacent errors using codes of low complexity and correction capacity, forming a matrix that allows the precise error location. The protected information is split into regions, where the algorithm detects the number of errors in the region. MRSC corrects adjacent errors with the distance between errors less than two cells. From MRSC, other two ECCs were proposed: eMRSC (SILVA et al., 2020) and TBEC-RSC (SILVA et al., 2023). The first one explores the scalability of the MRSC, extending the ECC to 32-bit data and presenting two ECC formats, that differ in performance, redundancy, and synthesis cost. The second is a linear version of MRSC, which extends the correction capability to burst errors. MRSC, eMRSC and TBEC-RSC are presented in the following.

4.1 MRSC (16, 32): ENCODING AND DECODING

Figure 12 shows the first MRSC (16, 32) version that encodes 16 data bits into a 32-bit word, performing four groups A, B, C, and D of four data bits, each and 16 redundancy bits split into (i) four diagonal bits Di_1, Di_2, Di_3, Di_4; (ii) four parity bits P_1, P_2, P_3, P_4; and (iii) eight check bits $XA_{13}, XA_{24}, XB_{13}, XB_{24}, XC_{13}, XC_{24}, XD_{13}, XD_{24}$. Table 5 presents equations for calculating redundancy bits – $\oplus$ represents the XOR (or-exclusive) logic operation.

Figure 12 – Codeword structure of MRSC (16, 32).



A_1	A_2	A_3	A_4	Di_1	Di_3	XA_{13}	XA_{24}
B_1	B_2	B_3	B_4	Di_2	Di_4	XB_{13}	XB_{24}
C_1	C_2	C_3	C_4	P_1	P_3	XC_{13}	XC_{24}
D_1	D_2	D_3	D_4	P_2	P_4	XD_{13}	XD_{24}

Source: SILVA et al., 2020.

Table 5 – MRSC (16, 32): Encoding equations.

Di bits		P bits	
$Di_1 = A_1 \oplus B_2 \oplus C_1 \oplus D_2$	(4.1)	$P_1 = A_1 \oplus B_1 \oplus C_1 \oplus D_1$	(4.5)
$Di_2 = A_2 \oplus B_1 \oplus C_2 \oplus D_1$	(4.2)	$P_2 = A_2 \oplus B_2 \oplus C_2 \oplus D_2$	(4.6)
$Di_3 = A_3 \oplus B_4 \oplus C_3 \oplus D_4$	(4.3)	$P_3 = A_3 \oplus B_3 \oplus C_3 \oplus D_3$	(4.7)
$Di_4 = A_4 \oplus B_3 \oplus C_4 \oplus D_3$	(4.4)	$P_4 = A_4 \oplus B_4 \oplus C_4 \oplus D_4$	(4.8)
X bits			

$XA_{13} = A_1 \oplus A_3$	(4.9)	$XC_{13} = C_1 \oplus C_3$	(4.13)
$XA_{24} = A_2 \oplus A_4$	(4.10)	$XC_{24} = C_2 \oplus C_4$	(4.14)
$XB_{13} = B_1 \oplus B_3$	(4.11)	$XD_{13} = D_1 \oplus D_3$	(4.15)
$XB_{24} = B_2 \oplus B_4$	(4.12)	$XD_{24} = D_2 \oplus D_4$	(4.16)

Source: Author.

The MRSC decoding process is divided into three steps:

I. Estimation of redundancy bits syndrome

The syndrome (S) estimation, as shown in chapter 2, is the process that consists of the operation XOR between the value of the original bit read and the value of the recalculated bit (RDi, RP, e RX). Therefore, the syndrome bit values are calculated as follows:

$$SDi = Di \oplus RDi \tag{4.17}$$

$$SP = P \oplus RP \tag{4.18}$$

$$SX = \mathrm{X} \oplus RX \tag{4.19}$$

II. Checking the decoding conditions

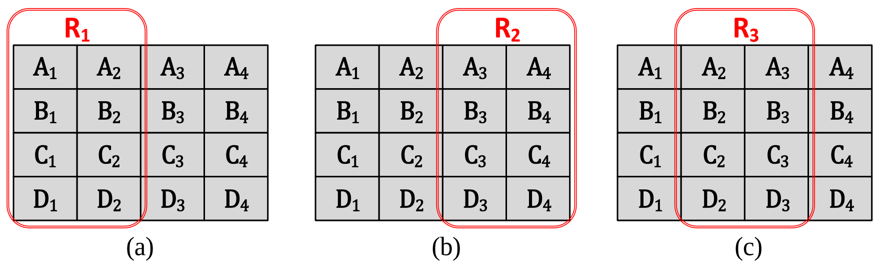
After the syndromes are calculated, one of the conditions shown must be satisfied for the correction process to start:

- Both syndromes SDi and SP need to have at least one of their values equal to 1.
- More than one Syndrome SX needs to have a value equal to 1 and SDi or SP are not null.
- These conditions were generated by analyzing the adjacent error patterns that the MRSC code seeks to correct. These conditions allow all possible adjacent errors to be detectable by the MRSC.

III. Wrong data region selection and correction

Once errors are detected by Step II, a specific region is selected to be corrected. The regions of this version of MRSC are divided, as shown in Figure 13.

Figure 13 – MRSC Regions (16, 32).



Source: SILVA et al. 2020.

For region selection, Table 6 was created to select the region with the most errors; it uses the results of Step I described previously and is shown below.

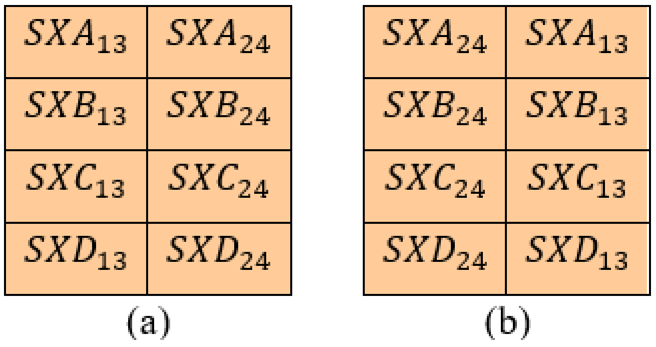
Table 6 – MRSC (16, 32): Region selection equation.

Chosen Region	**Region Selection Equations**
R1	$(SDi_1 + SDi_2 + SP_1 + SP_2) > (SDi_3 + SDi_4 + SP_3 + SP_4)$
R2	$(SDi_1 + SDi_2 + SP_1 + SP_2) < (SDi_3 + SDi_4 + SP_3 + SP_4)$
R3	$(SDi_1 + SDi_2 + SP_1 + SP_2) = (SDi_3 + SDi_4 + SP_3 + SP_4)$

Source: Author.

After choosing the region, the matrix formed by SX is used to correct the bits of the data region, shown in Figure 14, but its use depends on the region chosen. Regions R1 and R2 use matrix (a), and Region R3 uses matrix (b), which is (a) mirrored. It is important to emphasize that the R3 region is only selected in case of a tie (disregarding 0 to 0) in the number of errors detected between the syndromes SDi e SP.

Figure 14 – MRSC(16, 32): SX matrices used for region correction.



SXA_{13}	SXA_{24}
SXB_{13}	SXB_{24}
SXC_{13}	SXC_{24}
SXD_{13}	SXD_{24}

(a)

SXA_{24}	SXA_{13}
SXB_{24}	SXB_{13}
SXC_{24}	SXC_{13}
SXD_{24}	SXD_{13}

(b)

Source: Author.

The following subsection presents examples of the MRSC decoding process, from error detection to bit correction using the correction matrix.

4.1.1 MRSC (16, 32): Correction examples

The correction examples shown below are intended to facilitate an understanding of the logic proposed by the MRSC. The matrix of Figure 15 displays the result of the 16-bit word 1000000011111010 encoded by MRSC.

Figure 15 – Example of MRSC (16, 32) codeword.

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

Source: Author.

Figure 16 to Figure 20 display double-error pattern scenarios covered by the MRSC algorithm, which was designed to cover all double-error cases. The errors in the matrix and the recalculated syndromes are represented by red bits.

Figure 16 – Codeword with double error in region R2.

								SXs	
1	0	**1**	**1**	0	1	1	0	1 ⊕ 0 = 1	0 ⊕ 1 = 1
0	0	0	0	0	0	0	0	0 ⊕ 0 = 0	0 ⊕ 0 = 0
1	1	1	1	1	0	0	0	0 ⊕ 0 = 0	0 ⊕ 0 = 0
1	0	1	0	1	1	0	0	0 ⊕ 0 = 0	0 ⊕ 0 = 0

SDi $SDi_{1,2,3,4}$	0010 ⊕ 0001 = 0011
$SP_{1,2,3,4}$	1101 ⊕ 1110 = 0011
Chosen region	**0+0+0+0 < 1+1+1+1**

➡ R_2

Source: Author.

The conditions (a) and (b) of Step II are respected so that the correction can proceed to determine the region. More syndromes are found by calculating the selection of regions, and errors are detected in the R2 region so that it is selected for correction.

Figure 17 – Codeword with double error in region R1.

								SXs	
0	0	0	0	0	1	1	0	1 ⊕ 0 = **1**	0 ⊕ 0 = **0**
1	0	0	0	0	0	0	0	0 ⊕ 1 = **1**	0 ⊕ 0 = **0**
1	1	1	1	1	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
1	0	1	0	1	1	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**

$SDi_{1,2,3,4}$	0010 ⊕ 1110 = **1100**
$SP_{1,2,3,4}$	1101 ⊕ 1101 = **0000**
Chosen region	**1+1+0+0 > 0+0+0+0**

➡ R_1

Source: Author.

In this other case, the conditions (a) and (b) of Step II are also respected, proceeding to the region determination. More syndromes are found by calculating the region selection, and errors are detected in the R1 region, so it is selected for correction.

Figure 18 – Codeword with double error in region R3.

								SXs	
1	1	0	0	0	1	1	0	1 ⊕ 1 = **0**	0 ⊕ 1 = **1**
0	0	1	0	0	0	0	0	0 ⊕ 1 = **1**	0 ⊕ 0 = **0**
1	1	1	1	1	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
1	0	1	0	1	1	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**

$\boldsymbol{SDi}_{1,2,3,4}$	0010 ⊕ 0111 = **0101**
$\boldsymbol{SP}_{1,2,3,4}$	1101 ⊕ 1011 = **0110**
Chosen region	**0+1+0+1 = 0+1+1+0**

$\Rightarrow$ R_3

Source: Author.

In this case, again (a) and (b) of step II are respected, proceeding to determine the region. A tie was determined between the error detection of the two regions, resulting in the decision to correct the R3 region, using the mirror correction matrix.

Figure 19 – Codeword with double error between region R2 and redundancy bits.

								SXs	
1	0	0	1	1	1	1	0	1 ⊕ 1 = **0**	0 ⊕ 1 = **1**
0	0	0	0	0	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
1	1	1	1	1	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
1	0	1	0	1	1	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**

$\boldsymbol{SDi}_{1,2,3,4}$	1010 ⊕ 0011 = **1001**
$\boldsymbol{SP}_{1,2,3,4}$	1101 ⊕ 1100 = **0001**
Chosen region	**1+0+0+0 < 0+0+1+1**

$\Rightarrow$ R_2

Source: Author.

This case shows the importance of the conditions presented in step II. Note that only one of the syndrome bits *SX* is equal to 1, violating the condition (b). However, condition (a) is satisfied, so the decoding can proceed to determine the region to be corrected, which will be R2.

In situations where errors occurred entirely in the redundancy bits, the decoding conditions could avoid wrong corrections for up to 2 errors, as the following example shows.

Figure 20 – Codeword with double error in redundancy bits.

								SXs	
1	0	0	0	0	**0**	**0**	0	0 ⊕ 1 = **1**	0 ⊕ 0 = **0**
0	0	0	0	0	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
1	1	1	1	1	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
1	0	1	0	1	1	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**

$SDi_{1,2,3,4}$	0000 ⊕ 0010 = **0010**
$SP_{1,2,3,4}$	1101 ⊕ 1101 = **0000**
Chosen region	**0+0+0+0 < 0+0+1+0**

***SP* and *SXs* equal to 0**
Detection blocked

Source: Author.

In this case, none of the decoding conditions (II) were met, causing the detection to be blocked.

So far, all the situations shown refer to double errors in which the data is corrected, or there is no error in the correction, which would be the undue inversion of some data bits. Figure 21 shows an example where correction is not achieved.

Figure 21 – Codeword with triple error in regions R1 and R2.

								SXs	
0	**1**	**1**	0	0	1	1	0	1 ⊕ 1 = **0**	0 ⊕ 1 = **1**
0	**1**	0	0	0	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 1 = **0**
1	1	1	1	1	0	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
1	0	1	0	1	1	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**

$SDi_{1,2,3,4}$	0010 ⊕ 1100 = **1110**
$SP_{1,2,3,4}$	1101 ⊕ 0111 = **1010**
Chosen region	**1+0+1+1 > 1+1+0+0**

➡ R_1

Source: Author.

Note that all conditions are satisfied, and the algorithm chose the region with more errors; however, the correction matrix is only applied in R1, and the error in R2 persists when the decoding process ends. Therefore, the word is partially correct. This case represents a situation where errors occurred between two regions but did not result in a tie.

The case shown in Figure 21 only represents some possibilities of MRSC miscorrection. However, if multiple errors occur within the same data region are corrected by the MRSC, as shown in Figure 22. All syndrome bits detected by the correction matrix correct the inverted bits in Region 1, delivering the correct data.

Figure 22 – Codeword with triple error in region R1.

								SXs	
0	0	0	0	0	1	1	0	1 ⊕ 0 = **1**	0 ⊕ 0 = **0**
1	**1**	0	0	0	0	0	0	0 ⊕ 1 = **1**	0 ⊕ 1 = **1**
0	1	1	1	1	0	0	0	0 ⊕ 1 = **1**	0 ⊕ 0 = **0**
1	0	1	0	1	1	0	0	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**

***SDi*$_{1,2,3,4}$**	0010 ⊕ 1110 = **1100**
***SP*$_{1,2,3,4}$**	1101 ⊕ 0001 = **1100**
Chosen region	**1+1+1+1 > 0+0+0+0**

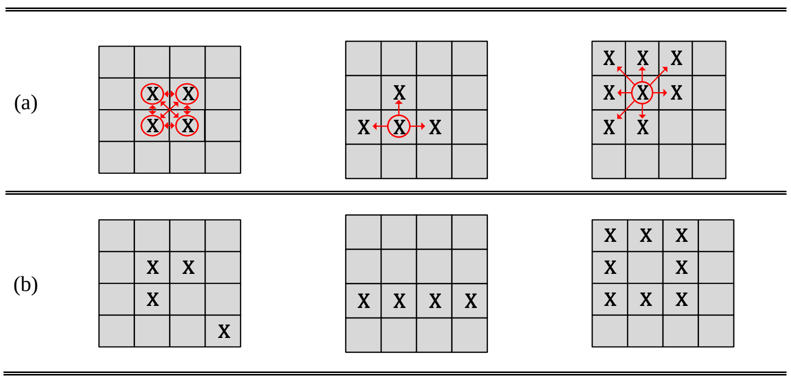
$\Rightarrow$ ***R*$_1$**

Source: Author.

4.1.2 MRSC (16, 32): Final considerations

The examples show that the MRSC decoding logic is simple and efficient, allowing to correct multiple errors in a data region and even covering situations in which part of the errors occurred in the redundancy bits, avoiding wrong corrections. MRSC(16, 32) can correct 100% of cases with up to two adjacent errors; although the algorithm detects other error cases, they will not be 100% covered. Furthermore, the code was built focused on adjacent errors, which are the most common types of MCUs in critical application memories (OGDEN et al., 2017). Figure 23 – (a) and (b) show sizable adjacent and non-adjacent error sets, respectively.

Figure 23 – (a) Adjacent and (b) non-adjacent multiple error patterns.



Source: Author.

Splitting data into regions facilitates error detection and correction in the same region. Besides, it allows correcting patterns of 3 and 4 adjacent errors when concentrated in the same data region. For errors in which data between two regions were compromised and did not result in a tie, there will be no correction (see Figure 21), as the correction matrix will only be applied in one of the regions. Note that Figure 18 can also be characterized as an error between two regions but results in a tie and correction of region R3. The MRSC codeword was constructed with the matrix format to resemble the 2-D memory structure; the codeword storage must follow the same spatial distribution to maintain coding efficacy. Inserting MRSC into commercial memories requires adjusting the input and output logic and requesting more read and write accesses (four accesses if the memory input bus is eight bits long).

As a potential improvement, the ECC Column-Line-Code (CLC) presented in (CASTRO, 2016) and (SILVA, 2018) shows a potential improvement to allow corrction of multiple regions. The decoding process is applied twice in the extended version of CLC (CLC-E). Although this made the CLC-E achieve higher correction rates than the standard version (CLC-S) for more aggressive errors, this strategy does not present an exciting cost-benefit since the cost of synthesis of the CLC-E decoder is much higher than the CLC-S decoder.

We saw that multiple errors encapsulated within the same region can be corrected with MRSC. This indicated that by increasing the number of data bits contained in the regions, the ECC would be able to correct more multiple error patterns. With this, the incidence of cases like the one in Figure 21 will be statistically lower.

Another point to consider about MRSC is its number of redundancy bits. Note that the 16-bit version produces 16 bits of redundancy, increasing 100% of the data bits. Chapter 3 shows that several authors have proposed techniques by varying the number of bits generated to increase the concentration of valuable data stored in memories and to give more flexibility to using the proposed techniques. In addition, a reduction in redundancy bits for some ECCs will cause a decrease in logical operations and, consequently, a reduction in the cost of synthesis and correction capacity, as we can see in (GRACIA-MORÁN et al., 2018) and (LIU et al., 2016).

The 16-bit MRSC version does not support larger regions (each region is a 4×2 matrix) or reducing the number of bits without significantly reworking its original logic. Therefore, we proposed an extension of the MRSC (eMRSC) to 32 data bits to explore those assumptions collected from MRSC results.

4.2 EMRSC: AN EXTENSION OF MRSC

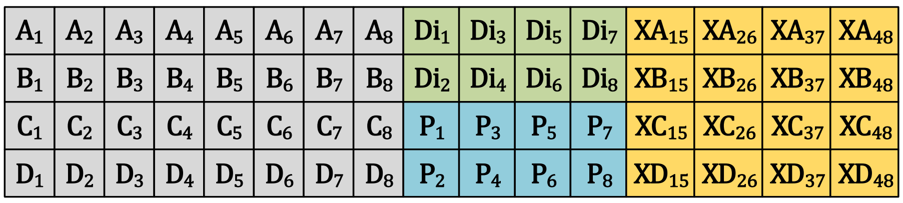
Let m be the number of data bits, r be the number of regions, and n be the total bits produced by encoding, then eMRSC (m, r, n) represents the new code format using the same region logic as the MRSC with a larger data area. We produced two 32-bit versions of eMRSC:

- eMRSC (32, 3, 64) keeps the redundancy rate at 100%, the number of regions at 3, and corrects more error patterns than the 16-bit version.
- eMRSC (32, 7, 56), which reduces the redundancy rate to 75%, the number of regions grows to 7, and the correction rate is lower than the eMRSC (32, 3, 64) to that achieved by the 16-bit version.

4.2.1 eMRSC (32, 3, 64)

The eMRSC (32, 3, 64) uses coding logic like the MRSC presented in the previous chapter, keeping the redundancy rate at 100% (32 bits generate 64 bits). Figure 24 shows the eMRSC codeword structure (32, 3, 64). Note that the same types of redundancy bits are considered in the encoding, but they are all duplicated. Table 7 presents the equations used to calculate the redundancy bits of this version.

Figure 24 – eMRSC codeword structure (32, 3, 64).



A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	Di_1	Di_3	Di_5	Di_7	XA_{15}	XA_{26}	XA_{37}	XA_{48}
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	Di_2	Di_4	Di_6	Di_8	XB_{15}	XB_{26}	XB_{37}	XB_{48}
C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	P_1	P_3	P_5	P_7	XC_{15}	XC_{26}	XC_{37}	XC_{48}
D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	P_2	P_4	P_6	P_8	XD_{15}	XD_{26}	XD_{37}	XD_{48}

Source: Author.

Table 7 – eMRSC (32, 3, 64): Equations for encoding.

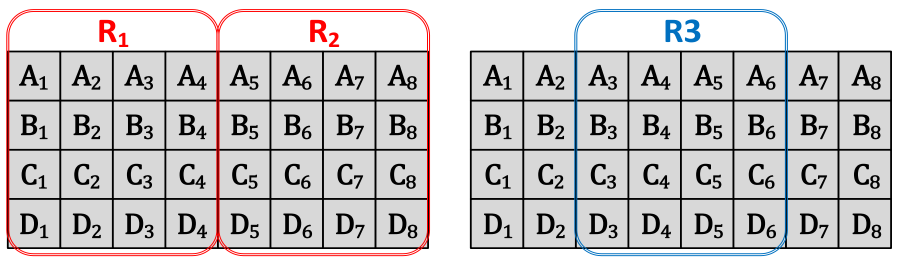
Di bits		P bits	
$Di_1 = A_1 \oplus B_2 \oplus C_1 \oplus D_2$	(4.20)	$P_1 = A_1 \oplus B_1 \oplus C_1 \oplus D_1$	(4.28)
$Di_2 = A_2 \oplus B_1 \oplus C_2 \oplus D_1$	(4.21)	$P_2 = A_2 \oplus B_2 \oplus C_2 \oplus D_2$	(4.29)
$Di_3 = A_3 \oplus B_4 \oplus C_3 \oplus D_4$	(4.22)	$P_3 = A_3 \oplus B_3 \oplus C_3 \oplus D_3$	(4.30)
$Di_4 = A_4 \oplus B_3 \oplus C_4 \oplus D_3$	(4.23)	$P_4 = A_4 \oplus B_4 \oplus C_4 \oplus D_4$	(4.31)
$Di_5 = A_5 \oplus B_6 \oplus C_5 \oplus D_6$	(4.24)	$P_5 = A_5 \oplus B_5 \oplus C_5 \oplus D_5$	(4.32)
$Di_6 = A_6 \oplus B_5 \oplus C_6 \oplus D_5$	(4.25)	$P_6 = A_6 \oplus B_6 \oplus C_6 \oplus D_6$	(4.33)
$Di_7 = A_7 \oplus B_8 \oplus C_7 \oplus D_8$	(4.26)	$P_7 = A_7 \oplus B_7 \oplus C_7 \oplus D_7$	(4.34)

$Di_8 = A_8 \oplus B_7 \oplus C_8 \oplus D_7$	(4.27)	$P_8 = A_8 \oplus B_8 \oplus C_8 \oplus D_8$	(4.35)
X bits			
$XA_{15} = A_1 \oplus A_5$	(4.35)	$XC_{15} = C_1 \oplus C_5$	(4.44)
$XA_{26} = A_2 \oplus A_6$	(4.37)	$XC_{26} = C_2 \oplus C_6$	(4.45)
$XA_{37} = A_3 \oplus A_7$	(4.38)	$XC_{37} = C_3 \oplus C_7$	(4.46)
$XA_{48} = A_4 \oplus A_8$	(4.39)	$XC_{48} = C_4 \oplus C_8$	(4.47)
$XB_{15} = B_1 \oplus B_5$	(4.40)	$XD_{15} = D_1 \oplus D_5$	(4.48)
$XB_{26} = B_2 \oplus B_6$	(4.41)	$XD_{26} = D_2 \oplus D_6$	(4.49)
$XB_{37} = B_3 \oplus B_7$	(4.42)	$XD_{37} = D_3 \oplus D_7$	(4.50)
$XB_{48} = B_4 \oplus B_8$	(4.43)	$XD_{48} = D_4 \oplus D_8$	(4.51)

Source: Author.

The eMRSC (32, 3, 64) decoding follows the same steps of Section 4.1: (I) Calculation of the redundancy syndrome, (II) Verification of decoding conditions, and (III) Selection and correction of the wrong data region. Figure 25 shows the eMRSC (32, 3, 64) regions. Note that for the eMRSC (32, 3, 64), the region is a 4×4 matrix, while the MRSC (16, 32) is 4×2. The region selection and correction logic of the eMRSC (32, 3, 64) follows the same principle as that presented by the MRSC (16, 32), with the addition of the extra syndrome bits, and the SX matrix was adjusted to a 4×4 matrix. Table 8 shows the equations for the region selection; Figure 26 shows the eMRSC (32, 3, 64) decoding steps, and Figure 27 displays the syndrome calculation process.

Figure 25 – eMRSC regions (32, 3, 64).


R_1
R_2
R3
A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8
B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8
C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8
D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8

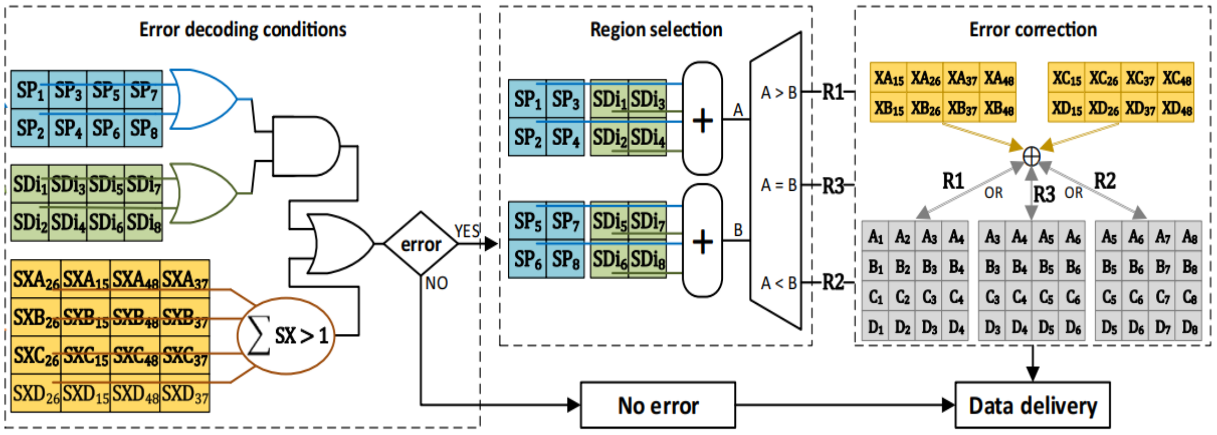
Source: Author.

Table 8 – eMRSC (32, 3, 64): Region Selection Equations.

Chosen Region	**Region Selection Equations**
R1	$\sum_{n=1}^{4} SDi_n + \sum_{n=1}^{4} SP_n > \sum_{n=5}^{8} SDi_n + \sum_{n=5}^{8} SP_n$
R2	$\sum_{n=1}^{4} SDi_n + \sum_{n=1}^{4} SP_n < \sum_{n=5}^{8} SDi_n + \sum_{n=5}^{8} SP_n$
R3	$\sum_{n=1}^{4} SDi_n + \sum_{n=1}^{4} SP_n = \sum_{n=5}^{8} SDi_n + \sum_{n=5}^{8} SP_n$

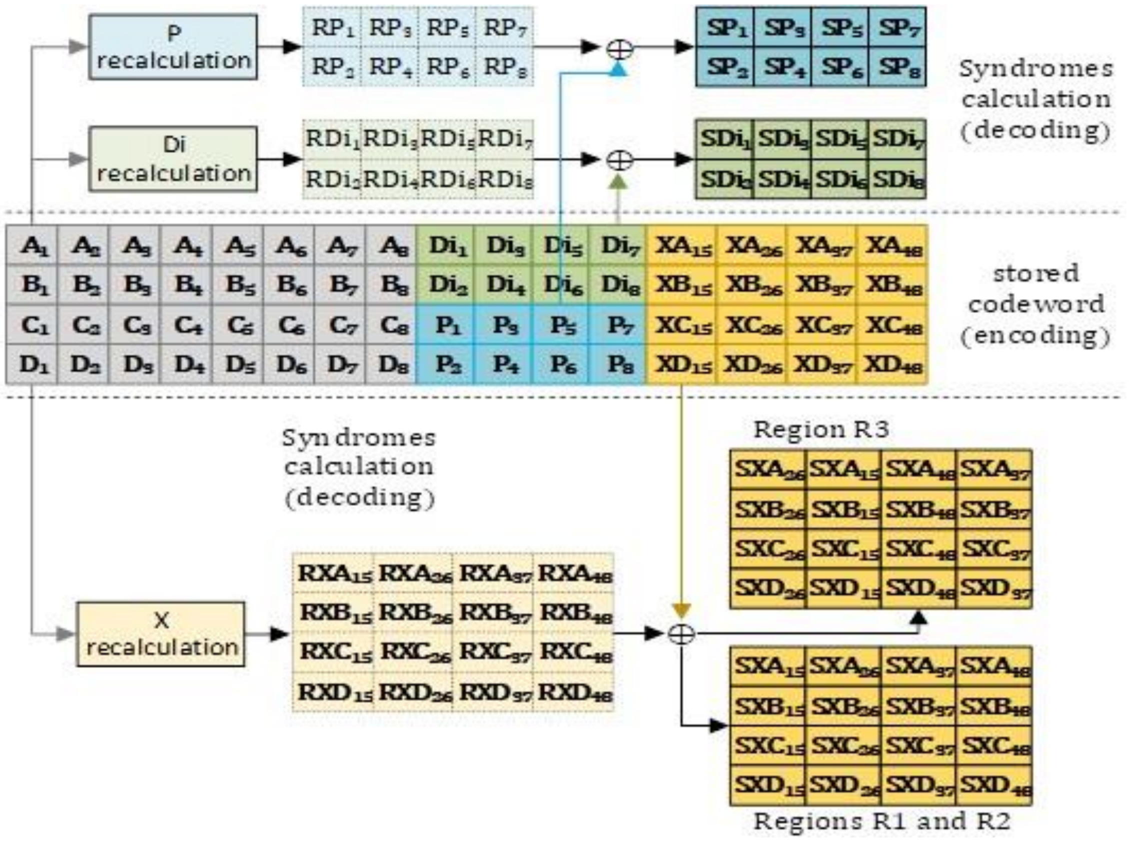
Source: Author.

Figure 26 – eMRSC (32, 3, 64): Decoding steps.



Source: Author.

Figure 27 – eMRSC (32, 3, 64): Syndrome calculation process.


P recalculation
Di recalculation
X recalculation
Syndromes calculation (decoding)
stored codeword (encoding)
Syndromes calculation (decoding)
Region R3
Regions R1 and R2

Source: Author.

4.2.2 eMRSC (32, 7, 56)

Figure 28 displays the codeword format of eMRSC(32, 7, 56) that has as one of its main features the reduction of the redundancy bit rate. Note that the redundancy number of this code is 24 bits, 8 bits less than the eMRSC (32, 3, 64), having a redundancy rate of 75%. However, the region considered for this version maintained the dimensions adopted in the MRSC (16, 32).

Figure 28 – eMRSC Codeword Structure (32, 7, 56).

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	Di_1	Di_3	Di_5	Di_7	XA_{1357}	XA_{2468}
B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	Di_2	Di_4	Di_6	Di_8	XB_{1357}	XB_{2468}
C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	P_1	P_3	P_5	P_7	XC_{1357}	XC_{2468}
D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	P_2	P_4	P_6	P_8	XD_{1357}	XD_{2468}

Source: Author.

Table 9 shows the equations used to create the eMRSC codeword, the Di and P bits have the same eMRSC (32,3,64) equations and the X bits have some additional XOR operations, as they cover more regions.

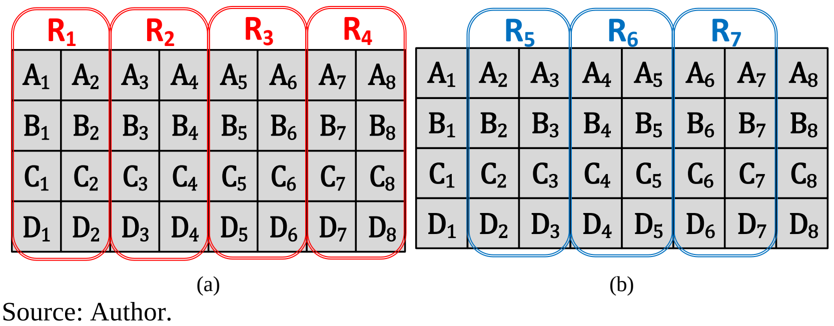
Table 9 – eMRSC (32, 7, 56): Equations for encoding.

Di bits		P bits	
$Di_1 = A_1 \oplus B_2 \oplus C_1 \oplus D_2$	(4.52)	$P_1 = A_1 \oplus B_1 \oplus C_1 \oplus D_1$	(4.60)
$Di_2 = A_2 \oplus B_1 \oplus C_2 \oplus D_1$	(4.53)	$P_2 = A_2 \oplus B_2 \oplus C_2 \oplus D_2$	(4.61)
$Di_3 = A_3 \oplus B_4 \oplus C_3 \oplus D_4$	(4.54)	$P_3 = A_3 \oplus B_3 \oplus C_3 \oplus D_3$	(4.62)
$Di_4 = A_4 \oplus B_3 \oplus C_4 \oplus D_3$	(4.55)	$P_4 = A_4 \oplus B_4 \oplus C_4 \oplus D_4$	(4.63)
$Di_5 = A_5 \oplus B_6 \oplus C_5 \oplus D_6$	(4.56)	$P_5 = A_5 \oplus B_5 \oplus C_5 \oplus D_5$	(4.64)
$Di_6 = A_6 \oplus B_5 \oplus C_6 \oplus D_5$	(4.57)	$P_6 = A_6 \oplus B_6 \oplus C_6 \oplus D_6$	(4.65)
$Di_7 = A_7 \oplus B_8 \oplus C_7 \oplus D_8$	(4.58)	$P_7 = A_7 \oplus B_7 \oplus C_7 \oplus D_7$	(4.66)
$Di_8 = A_8 \oplus B_7 \oplus C_8 \oplus D_7$	(4.59)	$P_8 = A_8 \oplus B_8 \oplus C_8 \oplus D_8$	(4.67)
X bits			
$XA_{1357} = A_1 \oplus A_3 \oplus A_5 \oplus A_7$	(4.68)	$XC_{1357} = C_1 \oplus C_3 \oplus C_5 \oplus C_7$	(4.72)
$XA_{2468} = A_2 \oplus A_4 \oplus A_6 \oplus A_8$	(4.69)	$XC_{2468} = C_2 \oplus C_4 \oplus C_6 \oplus C_8$	(4.73)
$XB_{1357} = B_1 \oplus B_3 \oplus B_5 \oplus B_7$	(4.70)	$XD_{1357} = D_1 \oplus D_3 \oplus D_5 \oplus D_7$	(4.74)
$XB_{2468} = B_2 \oplus B_4 \oplus B_6 \oplus B_8$	(4.71)	$XD_{2468} = D_2 \oplus D_4 \oplus D_6 \oplus D_8$	(4.75)

Source: Author.

The eMRSC (32, 7, 56) decoding process also follows the same rules presented in section 4.1; however, choosing the region to be corrected has become more complex. The complexity in selecting regions increased as more error situations between two regions became possible. Because of this, two groups of regions were considered for determining regions, shown in Figure 29.

Figure 29 – eMRSC (32, 7, 56) regions: group 1(a) and group 2(b).



Source: Author.

The regions *R₁*, *R₂*, *R₃*, and *R₄* are placed in group 1, where errors occurred only in their placeholders. If adjacent errors affect two regions of group 1, the regions *R₅*, *R₆*, and *R₃* that encompass group 2 are analyzed. For example, B_1 and B_2 bit errors make the chosen region the *R₁*; the region *R₅* is selected if the errors are in B_2 and B_3. Table 10 presents the equations for selecting the first group.

Table 10 – eMRSC (32, 7, 56): Region selection equations for group 1.

Chosen Region	**Region Selection Equations**
R_1	$\sum_{i=1}^{2} SDi_i + SP_i > \sum_{i=3}^{4} SDi_i + SP_i, \sum_{i=5}^{6} SDi_i + SP_i, \sum_{i=7}^{8} SDi_i + SP_i$
R_2	$\sum_{i=3}^{4} SDi_i + SP_i > \sum_{i=1}^{2} SDi_i + SP_i, \sum_{i=5}^{6} SDi_i + SP_i, \sum_{i=7}^{8} SDi_i + SP_i$
R_3	$\sum_{i=5}^{6} SDi_i + SP_i > \sum_{i=3}^{4} SDi_i + SP_i, \sum_{i=3}^{4} SDi_i + SP_i, \sum_{i=7}^{8} SDi_i + SP_i$
R_4	$\sum_{i=7}^{8} SDi_i + SP_i > \sum_{i=1}^{2} SDi_i + SP_i, \sum_{i=3}^{4} SDi_i + SP_i, \sum_{i=5}^{6} SDi_i + SP_i$

Source: Author.

If none of the conditions in Table 10 have been met, it is necessary to analyze the equations for group 2 presented in Table 11.

Table 11 – MRSC (32, 7, 56): Region selection equations for group 2.

Chosen region	**Equations to select region**
R₅	$\sum_{i=1}^{2} SDi_i + SP_i = \sum_{i=3}^{4} SDi_i + SP_i$
R₆	$\sum_{i=1}^{2} SDi_i + SP_i \neq \sum_{i=3}^{4} SDi_i + SP_i$ AND $\sum_{i=3}^{4} SDi_i + SP_i = \sum_{i=5}^{6} SDi_i + SP_i$
R₇	$\sum_{i=1}^{2} SDi_i + SP_i \neq \sum_{i=3}^{4} SDi_i + SP_i$ AND $\sum_{i=3}^{4} SDi_i + SP_i \neq \sum_{i=5}^{6} SDi_i + SP_i$ AND $\sum_{i=5}^{6} SDi_i + SP_i = \sum_{i=7}^{8} SDi_i + SP_i$

Source: Author.

For the composition of the equations in Table 11, error situations were considered only in adjacent regions. Therefore, these equations will not cover widely spaced error cases. However, as shown in (OGDEN, 2017) and discussed in chapter 2, spaced error situations represented a tiny portion of the failures observed in memories of applications exposed to radiation. The matrices used to correct these regions are those shown in Figure 30.

For eMRSC (32, 3, 64), the region size adjustment allows this code to correct more complex error patterns, increasing the correction capacity of the initial version. Moreover, for eMRSC (32, 7, 56), decreasing the number of bits does not reduce the original MRSC correction capability, keeping 100% correction for up to two errors.

Figure 30 – eMRSC (32, 7, 56): *SX* matrices for error correction.

SXA_{1357}	SXA_{2468}
SXB_{1357}	SXB_{2468}
SXC_{1357}	SXC_{2468}
SXD_{1357}	SXD_{2468}

(a) Regions R_1, R_2, R_3 and R_4

SXA_{2468}	SXA_{1357}
SXB_{2468}	SXB_{1357}
SXC_{2468}	SXC_{1357}
SXD_{2468}	SXD_{1357}

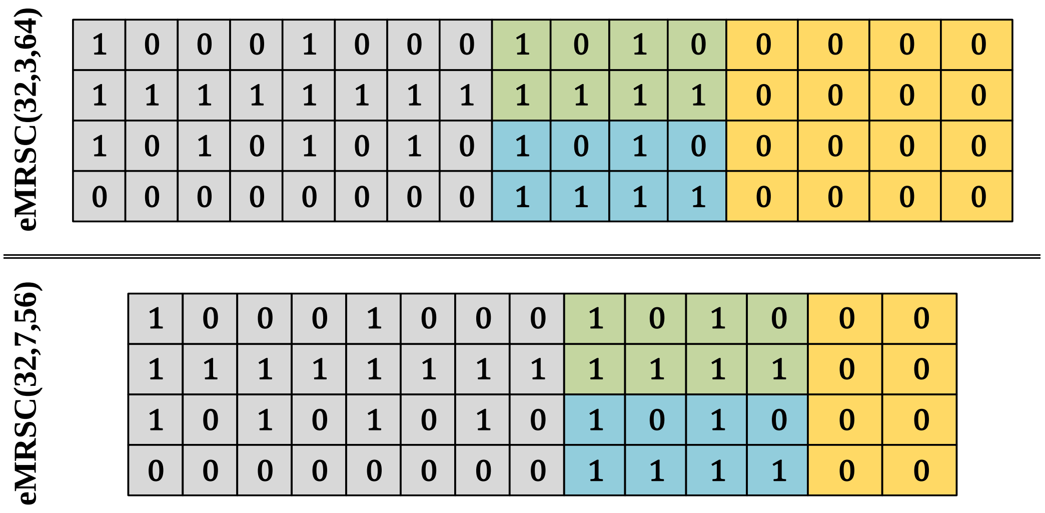
(b) Regions R_5, R_6 and R_7

Source: Author.

4.2.3 eMRSC (32, 3, 64) and eMRSC (32, 7, 56): Correction examples

This section presents some correction examples using the extended versions of the MRSC. Consider the word “10001000 11111111 10101010 00000000”, with their respective encodings for eMRSC (32, 3, 64) and eMRSC (32, 7, 56) shown in Figure 31.

Figure 31 – Encoding words for eMRSC (32, 3, 64) and eMRSC (32, 7, 56).



eMRSC(32,3,64)

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

eMRSC(32,7,56)

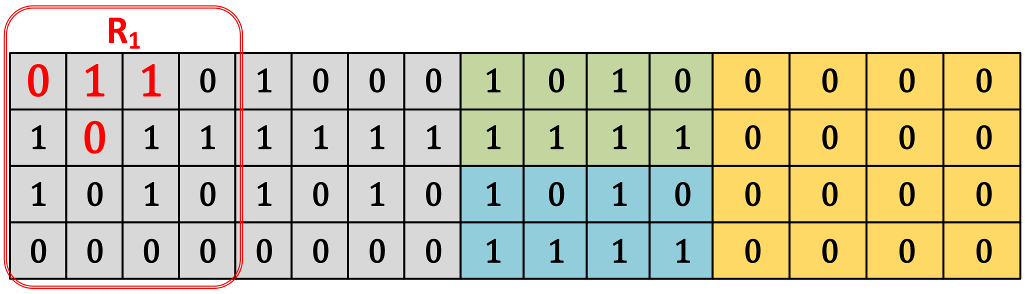
1	0	0	0	1	0	0	0	1	0	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0

Source: Author.

Figure 32 and Figure 33 present detectable error situations specifically for eMRSC versions. Remember that the central logic of the decoding algorithm of the versions shown was kept, except for the adjustments in eMRSC (32, 3, 64) and eMRSC(32, 7, 56).

Figure 32 displays how increasing the region contributed to correcting more complex error patterns. In this example, eMRSC (32, 3, 64) can correct an error pattern that the MRSC base version cannot correct.

Figure 32 – eMRSC (32, 3, 64): Fixed four errors in the R1 region.



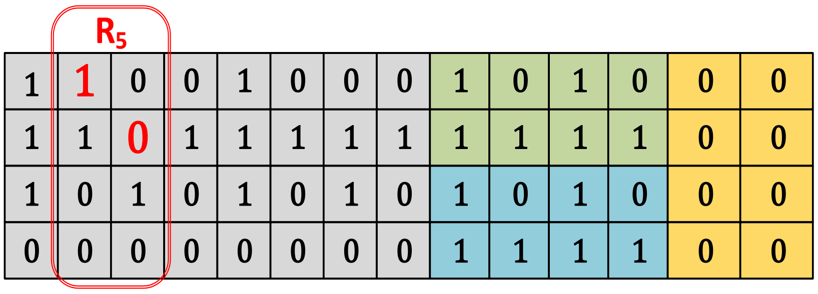
0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

$\boldsymbol{SDi} = 11011101 \oplus 01111101 = \mathbf{10100000}$				
$\boldsymbol{SP} \ = 11011101 \oplus 10111101 = \mathbf{01100000}$				
$\sum_{i=1}^{4} SDi_i + SP_i > \sum_{i=5}^{8} SDi_i + SP_i$				
Chosen region: *R₁*				
$\boldsymbol{SX}$	$0 \oplus 1 = \mathbf{1}$	$0 \oplus 1 = \mathbf{1}$	$0 \oplus 1 = \mathbf{1}$	$0 \oplus 0 = \mathbf{0}$
	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 1 = \mathbf{1}$	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$
	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$
	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$

Source: Author.

In Figure 21, the same error pattern could not be corrected by the 16-bit version of MRSC. In its extended version, the error was entirely enclosed by region *R₁* and corrected by matrix SX. In Figure 33, despite reduced the number of bits and the increase in regions, the eMRSC logic will maintain the ability to correct double errors.

Figure 33 – eMRSC (32, 7, 56): Fixed double errors in R5 region.



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

$\boldsymbol{SDi} = 11011101 \oplus 10001101 = \mathbf{01010000}$		
$\boldsymbol{SP} \ = 11011101 \oplus 10111101 = \mathbf{01100000}$		
$\sum_{i=1}^{2} SDi_i + SP_i = \sum_{i=3}^{4} SDi_i + SP_i$		
Chosen region: *R₅*		
$\boldsymbol{SX}$	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 1 = \mathbf{1}$
	$0 \oplus 1 = \mathbf{1}$	$0 \oplus 0 = \mathbf{0}$
	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$
	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$

Source: Author.

Note that the error shown in Figure 33 is located between the regions *R_1* and *R_2*; therefore, the equations in Table 10 match, leading to the analysis of the equations in Table 11. There will be a tie between errors in the region *R_1* and *R_2*, and no other, resulting in region *R_5* selection. Moreover, the matrix SX used is mirrored, as shown in Figure 30(b). Both codes presented had their correction capacity estimated, and the results are presented in chapter 5.

4.2.4 eMRSC (32, 3, 64) e eMRSC (32, 7, 56): Final considerations

This section illustrated eMRSC (32, 3, 64) and eMRSC (32, 7, 56), which both provided interesting alternatives for the use of MRSC. Compared to the MRSC base version, eMRSC (32, 3, 64) brought improvements in correcting aggressive multiple error patterns, without changes in logical basis and increasing the redundancy rate. The second version maintained the matrix structure of the region of the 16-bit version, which reduced the number of X bits and the redundancy rate.

The eMRSC did not cause significant changes to the original logic, especially for the eMRSC (32, 3, 64). For this version, there was only an increase in the data region structure; the base logic of the three regions was maintained. For eMRSC (32, 7, 56), the complexity of selecting the region with errors slightly increases when reducing the X bits, since an additional table of equations is required to detect errors between the two regions (see Figure 29).

MRSC extensions are not limited to what has been presented in section 4.2. It can be inferred that the code can be extended to even larger data words (e.g., 64-bit and 128-bit), especially if keeping the number of regions to three. The extended versions of MRSC still use some of the assumptions of the base code: the first is that only adjacent errors were considered when building the code, which is the most likely pattern in memory circuits, even though these errors can end up being more spaced out, as in cases of burst errors; The second is that the encoding matrix needs to be inserted into specific memories, making it difficult to use in commercial memories (generally requiring four read/write processes).

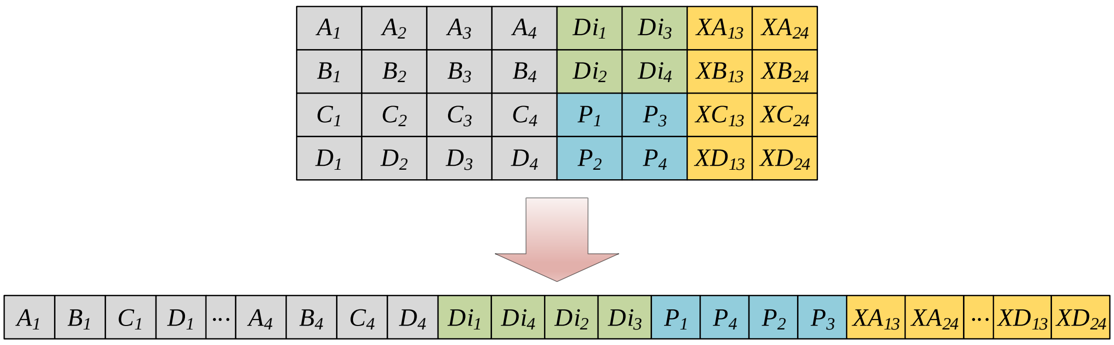
The next step to improve MRSC was to develop a version that correct burst errors without increasing computational cost or abruptly changing the base region selection logic. Additionally, the version must be more accessible to commercial memories. For this, a linear MRSC version was developed following the RSA concepts.

Furthermore, with minor adjustments in the positioning of data bits and the addition of specific logic conditions, it was possible to increase the range of detectable error patterns compared to the matrix model. This new version of the MRSC is presented in the following section.

4.3 TBEC-RSC: TRIPLE BURST ERROR - REGION SELECTION CODE

This proposal was designed to be a linear ECC format using the 16-bit MRSC logic and a version that can correct burst errors. The previous examples have shown that some complex patterns of multiple errors are detectable if they are concentrated in the same data region (see Figure 22). Therefore, the idea of linearizing the technique starts not only to enable the code to be easily applicable in commercial memories, which is a problem for matrix codes in general but also to increase the correction capacity concerning the original proposal. The code can correct burst errors of up to three bits with minor adjustments to the existing MRSC logic and with the repositioning of some bits. Due to this correction capability, this version was named TBEC-RSC (Triple Burst Error Corrector – Region Selection Code) (Silva, 2023). Figure 34 shows the proposed structure for the TBEC-RSC (16, 32).

Figure 34 – TBEC-RSC (16, 32): Codeword structure.


A_1 A_2 A_3 A_4 Di_1 Di_3 XA_{13} XA_{24}
B_1 B_2 B_3 B_4 Di_2 Di_4 XB_{13} XB_{24}
C_1 C_2 C_3 C_4 P_1 P_3 XC_{13} XC_{24}
D_1 D_2 D_3 D_4 P_2 P_4 XD_{13} XD_{24}
A_1 B_1 C_1 D_1 ··· A_4 B_4 C_4 D_4 Di_1 Di_4 Di_2 Di_3 P_1 P_4 P_2 P_3 XA_{13} XA_{24} ··· XD_{13} XD_{24}

Source: Author.

Note that this model reduces the probability of multiple errors occurring in bits from different regions or generating a tie in error detection (same number of errors detected in adjacent regions) between regions R_1 and R_2. Therefore, this model aims to ensure that when multiple errors occur, they are more probable to be in the same region, avoiding situations like the one shown in Figure 21.

Adjusting the bit position will improve the error spacing and consequently ease the correction, but more was needed to allow the code to correct triple errors. The equations for generating the TBEC-RSC (16,32) code redundancy bits are shown in Table 12.

Note that the equations for bits Di and X, are the same as those shown in Table 5; however, changes have been made to the equations in P bits.

Table 12 – TBEC-RSC (16, 32): Equations for coding.

Di bits		P bits	
$Di_1 = A_1 \oplus B_2 \oplus C_1 \oplus D_2$	(4.76)	$P_1 = A_1 \oplus A_2 \oplus B_1 \oplus B_2$	(4.80)
$Di_2 = A_2 \oplus B_1 \oplus C_2 \oplus D_1$	(4.77)	$P_2 = C_1 \oplus C_2 \oplus D_1 \oplus D_2$	(4.81)
$Di_3 = A_3 \oplus B_4 \oplus C_3 \oplus D_4$	(4.78)	$P_3 = A_3 \oplus A_4 \oplus B_3 \oplus B_4$	(4.82)
$Di_4 = A_4 \oplus B_3 \oplus C_4 \oplus D_3$	(4.79)	$P_4 = C_3 \oplus C_4 \oplus D_3 \oplus D_4$	(4.83)
X bits			
$XA_{13} = A_1 \oplus A_3$	(4.84)	$XC_{13} = C_1 \oplus C_3$	(4.88)
$XA_{24} = A_2 \oplus A_4$	(4.85)	$XC_{24} = C_2 \oplus C_4$	(4.89)
$XB_{13} = B_1 \oplus B_3$	(4.86)	$XD_{13} = D_1 \oplus D_3$	(4.90)
$XB_{24} = B_2 \oplus B_4$	(4.87)	$XD_{24} = D_2 \oplus D_4$	(4.91)

Source: Author.

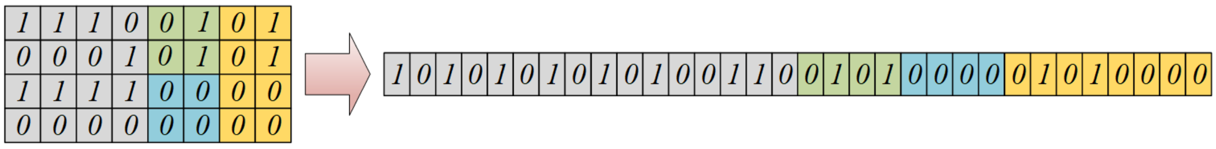
The changes in the equations of P allowed the detection of adjacent and non-adjacent triple errors. Considering a situation where errors occur in bits A_1 and C_1, if the equations for the bit P were the same as presented in Table 5, the syndrome of P_1 errors would detect an error. However, using Table 12, these errors are detected by syndromes P_1 and P_2.

Another change made for this version was the addition of a special condition in Step II of the decoding process (Checking the decoding conditions). When analyzing the triple error patterns in TBEC-RSC (16,32), it was found that condition ***b*** is met for a triple error in the bits P_3, XA_{13} and XA_{24}, performing an erroneous correction in the region R_1. We avoided this miscorrection by adding the ($SP = 1$ AND $SDi = 1$ AND $SX = 2$) condition. The subsequent examples illustrate the proposed changes to TBEC-RSC compared to the original MRSC (16, 32) version.

4.3.1 TBEC-RSC (16, 32): Correction Examples

Consider, for example, the 16-bit word "1110000111110000". Figure 35 shows how the conversion of the word encoded by MRSC to TBEC-RSC would be.

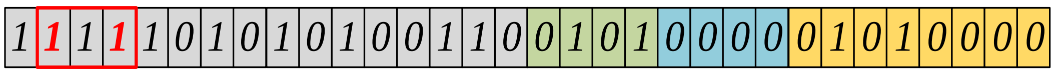
Figure 35 – TBEC-RSC (16, 32): Codeword example.



Source: Author.

Figure 36 shows an example where the need to change the equation for bits P was discussed to detect and correct non-adjacent error patterns or burst errors.

Figure 36 – Non-adjacent errors corrected by TBEC-RSC (16, 32).

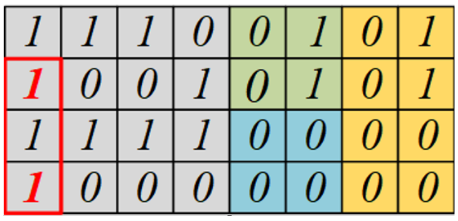


$\boldsymbol{SDi} = 0011 \oplus 0011 = \mathbf{0000}$		
$\boldsymbol{SP} \ \ = 0000 \oplus 1100 = \mathbf{1100}$		
$\sum_{i=1}^{2} SDi_i + SP_i > \sum_{i=3}^{4} SDi_i + SP_i$		
Region selected: *R₁*		
$\boldsymbol{SX}$	$0 \oplus 0 = \mathbf{0}$	$1 \oplus 1 = \mathbf{0}$
	$0 \oplus 1 = \mathbf{1}$	$1 \oplus 1 = \mathbf{0}$
	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$
	$0 \oplus 1 = \mathbf{1}$	$0 \oplus 0 = \mathbf{0}$

Source: Author.

Figure 37 shows the same pattern as Figure 36 affecting bits on the vertical; bits P in Figure 37 cannot detect which region has errors because two bitflips are present in the same syndrome for both SDi and SP, which cancel both in XOR operations. Comparing the approaches shows that the new P equations (Figure 36) increase the code detection potential, enabling the identification of errors in R_1.

Figure 37 – Non-adjacent errors corrected by MRSC (16, 32)



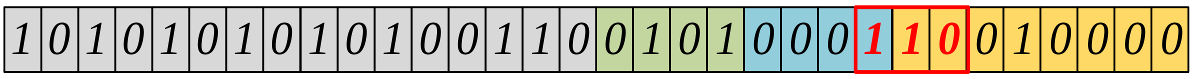
1	*1*	*1*	*0*	*0*	*1*	*0*	*1*
1	*0*	*0*	*1*	*0*	*1*	*0*	*1*
1	*1*	*1*	*1*	*0*	*0*	*0*	*0*
1	*0*	*0*	*0*	*0*	*0*	*0*	*0*

$\boldsymbol{SDi}$ = 0011 ⊕ 0011 = **0000**		
$\boldsymbol{SP}$ = 0000 ⊕ 0000 = **0000**		
$\sum_{i=1}^{2} SDi_i + SP_i = \sum_{i=3}^{4} SDi_i + SP_i$		
Selected region: None		
$\boldsymbol{SX}$	0 ⊕ 0 = **0**	1 ⊕ 1 = **0**
	0 ⊕ 1 = **1**	1 ⊕ 1 = **0**
	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**
	0 ⊕ 1 = **1**	0 ⊕ 0 = **0**

Source: Author.

Figure 38 presents the special case mentioned at the beginning of the TBEC-RSC (16, 32) description, in which a triple error affected the bits P_3, XA_{13} and XA_{24}. The correction will proceed if the special condition maintains the same value because **more than one SX syndrome will have a value 1 (condition b)**.

Figure 38 – Triple error activating special condition of the TBEC-RSC (16, 32).



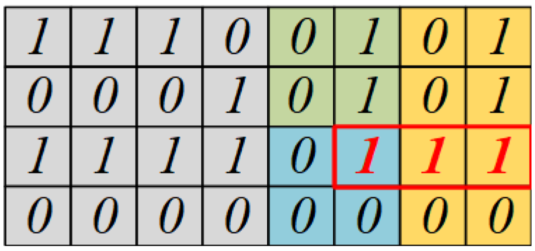
1	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	***1***	***1***	***0***	*0*	*1*	*0*	*0*	*0*	*0*

$\boldsymbol{SDi}$ = 0011 ⊕ 0011 = **0000**		
$\boldsymbol{SP}$ = 0000 ⊕ 0010 = **0010**		
$\sum_{i=1}^{2} SDi_i + SP_i > \sum_{i=3}^{4} SDi_i + SP_i$		
Region selected: Correction stopped due to the special condition triggered		
$\boldsymbol{SX}$	0 ⊕ 1 = **1**	1 ⊕ 0 = **1**
	0 ⊕ 0 = **0**	1 ⊕ 1 = **0**
	0 ⊕ 0 = 0	0 ⊕ 0 = **0**
	0 ⊕ 0 = **0**	0 ⊕ 0 = **0**

Source: Author.

Figure 39 shows the same pattern applied in the original version of the MRSC (16, 32). Following the MRSC decoding logic, the displayed error will not be corrected and will eventually cause unnecessary inversion of data bits. Note that MRSC was not designed to account for this special condition or correct triple errors. Adding this condition to the MRSC logic and gathering the other X neighboring bits increase the code correction power.

Figure 39 – Triple error causing patch error in MRSC (16, 32).



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

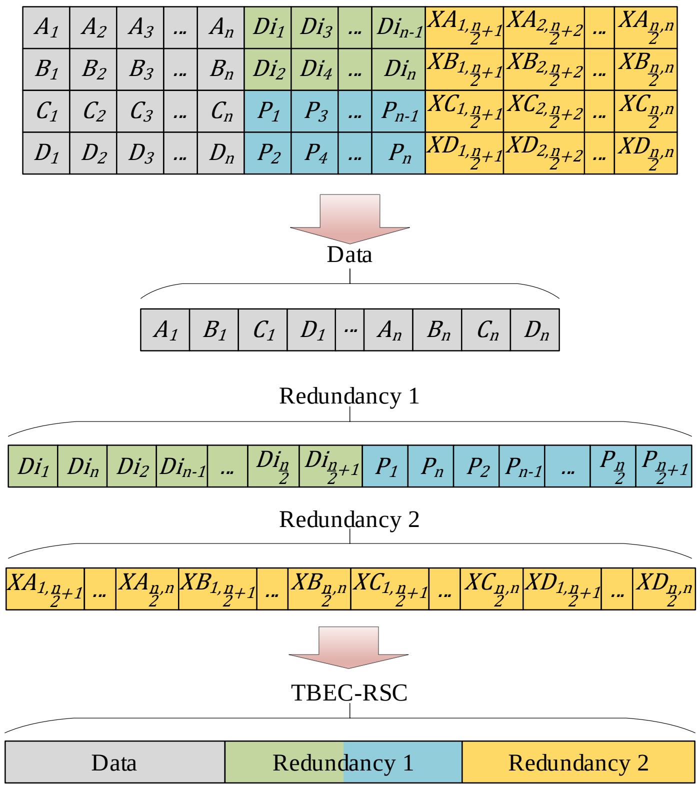
$\boldsymbol{SDi} = 0011 \oplus 0011 = \mathbf{0000}$		
$\boldsymbol{SP} = 0000 \oplus 0010 = \mathbf{0010}$		
$\sum_{i=1}^{2} SDi_i + SP_i = \sum_{i=3}^{4} SDi_i + SP_i$		
Region selected: R_2		
$\boldsymbol{SX}$	$0 \oplus 0 = \mathbf{0}$	$1 \oplus 1 = \mathbf{0}$
	$0 \oplus 0 = \mathbf{0}$	$1 \oplus 1 = \mathbf{0}$
	$0 \oplus 1 = \mathbf{1}$	$0 \oplus 1 = \mathbf{1}$
	$0 \oplus 0 = \mathbf{0}$	$0 \oplus 0 = \mathbf{0}$

Source: Author.

4.3.2 TBEC-RSC (16, 32): Scalability

The TBEC-RSC encoding divides the data bits into four groups. Let S be the encoded data size and n be the number of bits per group, then $S = 2^k \ \forall\ k \in \mathbb{N}, k \geq 4$, and $n = \frac{S}{4}$. This format allows TBEC-RSC to scale to base-2 data words larger than $S = 16$, e.g., 32 and 64 bits. Figure 40 shows how the TBEC-RSC scales.

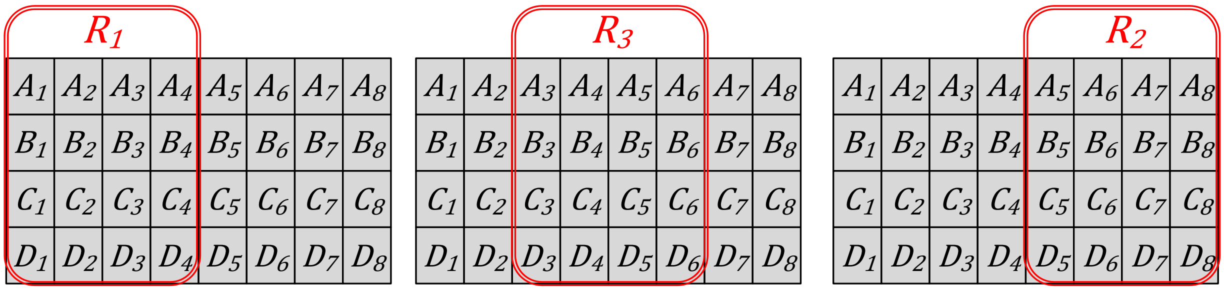
Figure 40 – TBEC-RSC scalability scheme.



Source: Author.

A remarkable feature observed when extending TBEC-RSC is that the overall error correction rate of the code increases as the codeword grows. This increase occurs because MRSC and TBEC-RSC correct errors concentrated in a region. Increasing the data bits also raises the region size, meaning that larger errors are more likely to be found in a single region. Figure 41 presents the correcting regions for the 32-bit TBEC-RSC version, which can be compared to the 16-bit version of Figure 14.

Figure 41 – TBEC-RSC regions examples for 16-bit and 32-bit formats.



Source: Author.

In resume, bigger regions increase the chance of multiple adjacent errors to be in a single region, raising the error correction rate. The following section presents the scalability effect on the error correction rate.

4.3.3 TBEC-RSC (16, 32): Final considerations

TBEC-RSC (16, 32) is a more robust version of the MRSC code, filling a gap for which other approaches need to be idealized. Although most cases of MCU in memory are adjacent, the occurrence of burst errors also poses a threat. In addition, using Matrix codes in commercial memories can be a negative point, even with these techniques having a reduced operating cost compared to other approaches. Matrix-type codes can be linearized, or their codeword written in a single memory address, but not all works show the best strategy for this to be done without affecting the correctness of the proposed technique. The organization of TBEC-RSC bits is fundamental to concentrating errors in the same region, in addition to minor logic adjustments that allowed the burst error correction of up to 3 bits.

Given everything that has been shown, the next step of this thesis consisted of validating the propositions made about the three MRSC versions by carrying out experiments involving the ability to correct the techniques, reliability analysis, cost estimation and synthesis. The next chapter describes these experiments and results.

5 EXPERIMENTAL RESULTS AND DISCUSSION

The previous chapter described the MRSC, eMRSC, and TBEC-RSC ECCs and their particularities. Error cases were also discussed, illustrating situations in which the decoding was successful and situations in which the errors were not corrected, boosting the development of improvements and, consequently, of the extended approaches.

This chapter presents the experiments carried out to validate the proposed techniques. The experiment script used followed what was applied in most of the literature regarding ECCs for memories. The experiments were split into the following categories:

- **Correction coverage evaluation** consists of evaluating scenarios from 1 to 8 errors, testing all possible patterns and the entire codeword to find the correction rate for each error scenario.
- **Reliability evaluation** is characterized by calculating the Mean Time To Failure (MTTF) of each technique according to the memory sizes.
- **Synthesis cost and trade-off performance evaluation** of the synthesis cost of each technique, using specialized tools and low-scale technology. Moreover, these results were combined with correction coverage to estimate the trade-off between correction coverage and synthesis.

Table 13 – Division of ECCs used for experimental works.

Proposed ECC	ECCs used in comparison	Reason of choosing	N° of data bits	N° of redundancy bits	Experimented error type
MRSC (SILVA et al, 2017a)	CLC (SILVA et al, 2018)	Recent notable work	16-bit	24-bit	Adjacent errors
	Matrix (ARGYRIDES et al, 2007)	Traditional work	16-bit	16-bit	
	Reed-Muller (VARGHESE et al., 2013)	Traditional work	16-bit	16-bit	
eMRSC (SILVA et al, 2020)	OLS (LIU et al., 2016)	Recent notable work	32-bit	23-bit, 36-bit	Adjacent errors
	Matrix (ARGYRIDES et al., 2007)	Traditional work	32-bit	28-bit	
	DMC (GUO et al., 2014)	Recent notable work	32-bit	36-bit	
TBEC-RSC (SILVA et al, 2023)	MRSC (SILVA et al., 2017a)	Recent notable work	16-bit	16-bit	Burst errors
	FUEC (GRACIA-Morán et al., 2018)	Recent notable work	16-bit	7-bit, 8-bit, 9-bit	
	TBEC-QUEAC (LI et al., 2018)	Recent notable work	16-bit	8-bit	

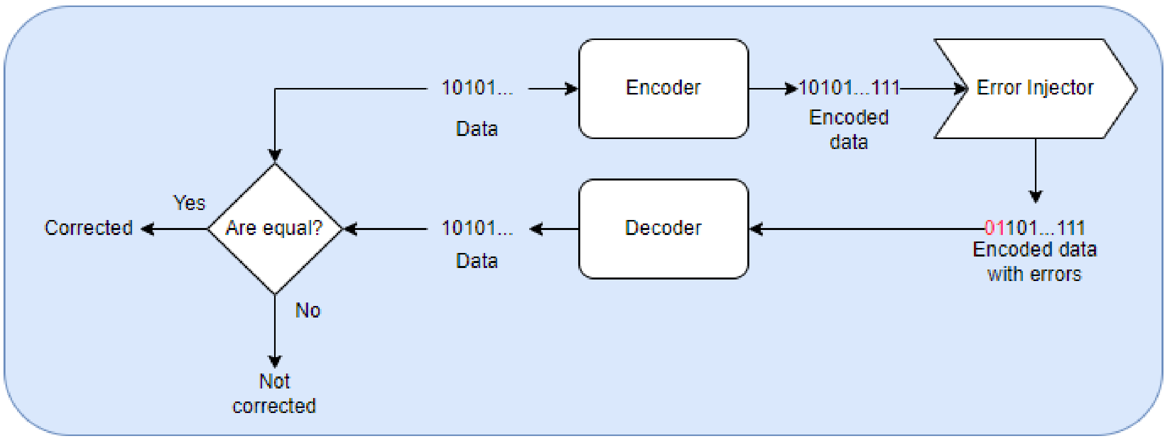
Source: Author.

For each MRSC proposal, the experiments were carried out with other works selected for comparison. Table 13 presents the codes used for comparison along with the justification for which they were chosen. As each of the chapters with the proposals resulted in different publications, each proposal had a group of ECCs to carry out the experiments. Note that among the techniques chosen in the comparison, it was sought to merge the comparison of the proposals with traditional works in the area, such as the Matrix code (ARGYRIDES et al., 2007) and the Reed-Muller (2, 5) (VARGHESE et al., 2013) with recent and well-referenced works such as the CLC (SILVA et al., 2018) and FUEC (GRACIA-MORÁN, 2018).

5.1 CORRECTION COVERAGE EVALUATION

We built a test script for correction coverage assessment that simulates the encoding, error injection, and decoding process, following the flow diagram of Figure 42.

Figure 42 – Proposed simulation flow.



Source: Author.

Figure 42 describes the simulation flow that encompasses the subsequent processes: (i) the Encoder codifies the input data following the ECC standards, resulting in a codeword; (ii) the Error injector inserts adjacent or burst errors into the codeword; (iii) The Decoder reads the codeword trying to fix the inserted errors to produce an output data free of errors; then, (iv) input and output data are compared to verify the ECC correction power.

The encoding and decoding processes for all ECCs have been extensively discussed in later chapters. The error injector was developed considering two possible situations: (I) Adjacent errors and (II) Burst errors.

Situation (I) was used to evaluate the MRSC and eMRSC proposals. Considering only adjacent errors, one million random scenarios were evaluated for each pattern from 1 to 8 bitflips. This expressive number of scenarios was necessary to ensure that most error situations were tested since the complexity of producing a simulation that quantifies the number of possibilities of adjacent errors for each analyzed error pattern is vast. All codes compared with

MRSC and eMRSC were placed in a matrix format and subjected to the same error injection conditions, performing fair simulations.

Situation (II) was used for the TBEC-RSC proposal because this ECC was developed to correct burst errors. Unlike Situation (I), the number of error possibilities for linear codes and burst errors is easily estimated. To explain this, the possibilities of multiple errors for Situation (I) are for the horizontal, vertical, and diagonal; in contrast, for Situation (II), the possibilities are limited to only one direction (horizontal, for example). Equation 8.1 computes the number of burst error patterns (NBP) per size *l*.

$$\text{NBP} = 2^{(l-2)} \tag{5.1}$$

Table 14 shows how the burst error patterns vary from 2 to 8. Therefore, it was possible to build a simulation with the exact number of test cases for each evaluated ECCs. In addition, all codes analyzed with TBEC-RSC use linear format.

Table 14 – Burst error patterns available for *l* ranging from 2 to 8.

l	**NBP**	**Burst error pattern**
2	1	*11*
3	2	*101, 111*
4	4	*1001, 1101, 1011, 1111*
5	8	*10001, 11001, ..., 10111, 11111*
6	16	*100001, 110001, ..., 101111, 111111*
7	32	*1000001, 1100001, ..., 1011111, 1111111*
8	64	*10000001, 11000001, ..., 10111111, 11111111*

Legend: 0 represents a non-error; 1 represents an error

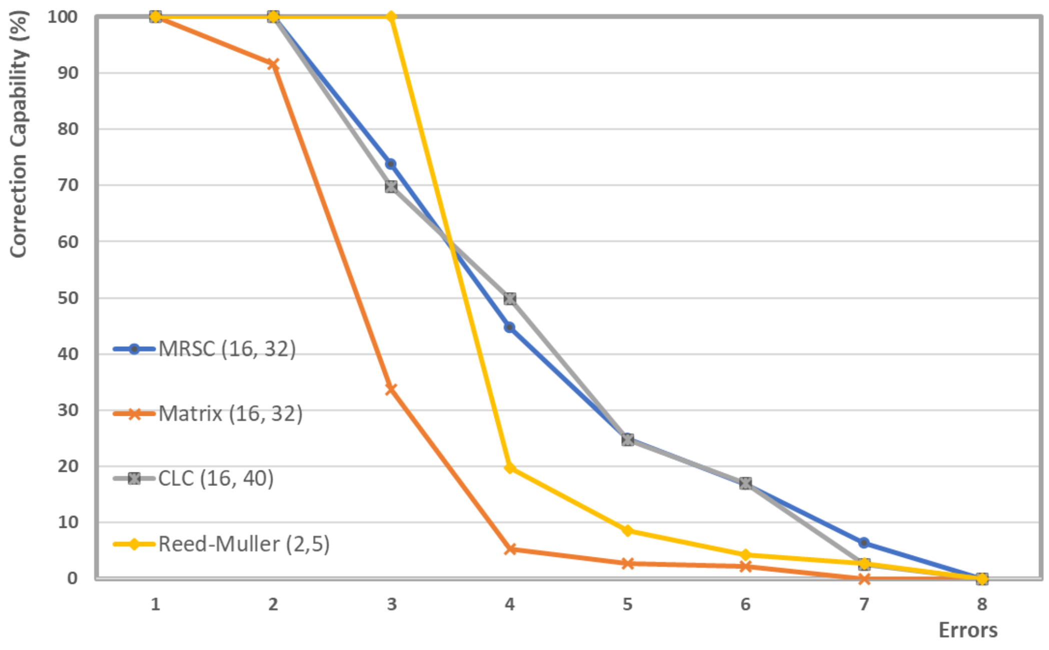
Source: Author

We used MATLAB software to build and control the simulation, which is widely used by the academic community to perform statistical measurements, to extract the correction capabilities. The following sections present the results collected for this thesis' proposal.

5.1.1 MRSC Correction Analysis

Figure 43 presents the correction results between the codes compared for the MRSC, following Table 13 .

Figure 43 – MRSC: Correction capability comparison.



Source: Author

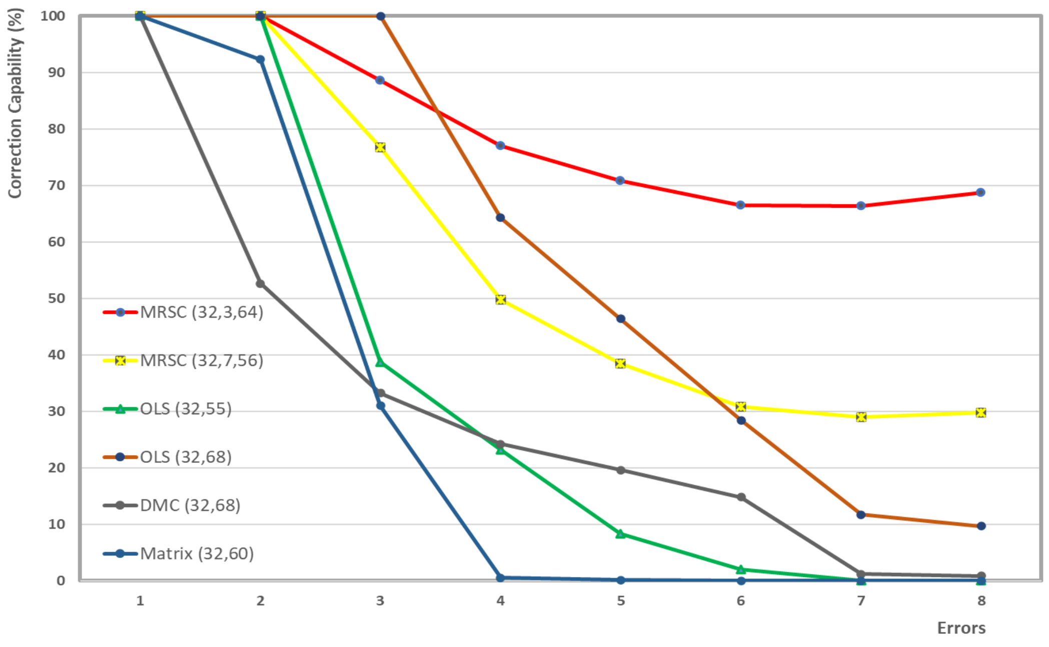
Figure 43 shows that all techniques except Matrix code corrected 2-error patterns. The Matrix code can correct double error patterns if they are concentrated in the data bits, and redundancy errors are not detectable, due to the lack of parity bits in its structure (see Figure 3). For more complex patterns, we see that both Matrix and Reed-Muller (2, 5) lose effectiveness in correcting errors. In the case of Reed-Muller (2, 5) the concept of hamming distance is used, which determines how much the code will be able to correct. In the analyzed version, the distance is eight, which will result in an ECC that corrects up to three bits 100%, other error patterns greater than three will not be detectable if they reach the data bits.

The MRSC and CLC codes both cover all bits of the coded word and are not limited to using the hamming distance concept, which made them the most robust codes in the overall analysis of correction results. However, note that the CLC Code uses more redundancy bits (eight more bits), a factor that can lead to higher computational and synthesis costs.

5.1.2 eMRSC Correction Analysis

Figure 44 presents the correction results between the codes compared to the eMRSC versions, following Table 13 .

Figure 44 – eMRSC: Correction capability comparison.



Source: Author

From Figure 44, all the codes except Matrix and DMC corrected all the two error patterns. Of these, only the OLS (32, 68) corrected triple errors. Again, we see that the proposed techniques had a less prominent performance drop than the other ECCs compared.

DMC (32, 68), Matrix (32, 60) and OLS (32, 55) were the ECCs that obtained the lowest correction capabilities within the analysis. This is due to the first being an ECC that corrects specific error patterns defined by the author, therefore it has limited correction; the second, as discussed in the previous analysis, the code was not designed to cover all bits, then errors between redundancy and data bits may not be corrected; the third is a more limited version than the OLS (32, 68), having fewer bits of redundancy and losing correction effectiveness throughout analysis.

From 4 to 8 errors, eMRSC (32, 3, 64) showed the best correction rates, surpassing the alternative version eMRSC (32, 7, 56) and OLS (32, 68). This is due to what was illustrated in the examples of section 4.2, as a larger data region encapsulates more bits, making the detection and correction process easier, and allowing the correction of more complex errors. For eMRSC (32, 3, 64) correction was over 60% throughout the analysis.

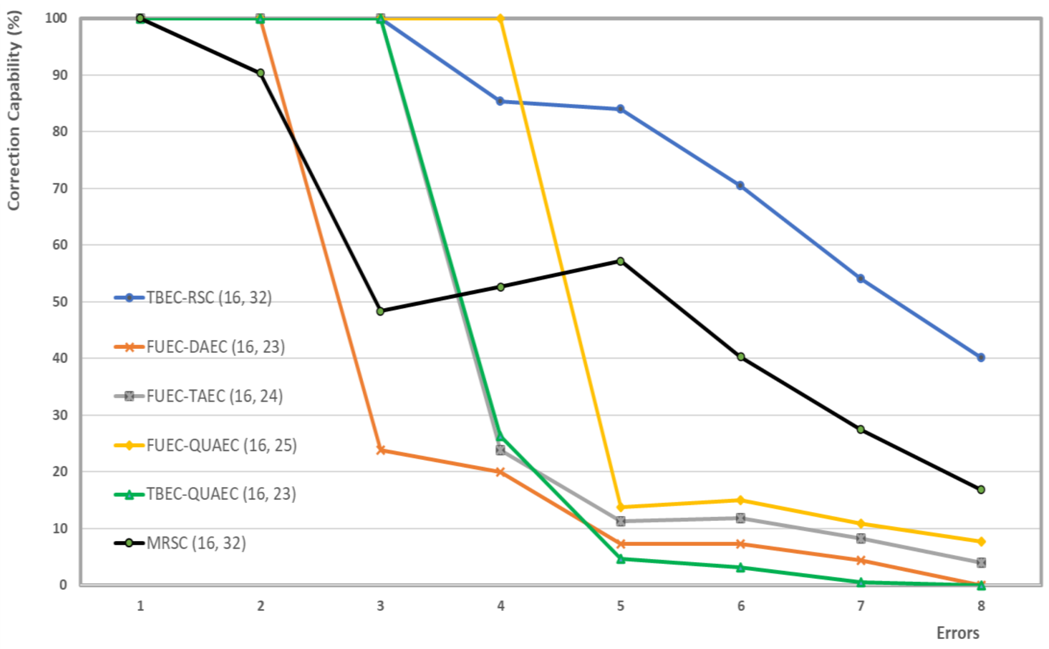
Although the OLS (32, 68) had greater correction capacity than the eMRSC (32, 7, 56) between three and five errors, it is surpassed by the same from 6 to 8 errors. The eMRSC (32, 7, 56) has as a primary feature redundancy reduction, then it was expected that the correction would also be reduced.

5.1.3 TBEC-RSC correction analysis

Figure 45 presents the correction results between the codes compared for the TBEC-RSC (16, 32), following Table 13 .

From Figure 45, we see that TBEC-RSC corrected all 3-bit burst error patterns, while the linearized version of MRSC, without the changes implemented by the new proposal, was not able to correct not all cases of double errors, noting that the changes implemented in the TBEC-RSC had major impacts on efficiency. In addition, TBEC-RSC was the code that most corrected errors in the 5 to 8 error scenarios.

Figure 45 – TBEC-RSC: Correction capability comparison.



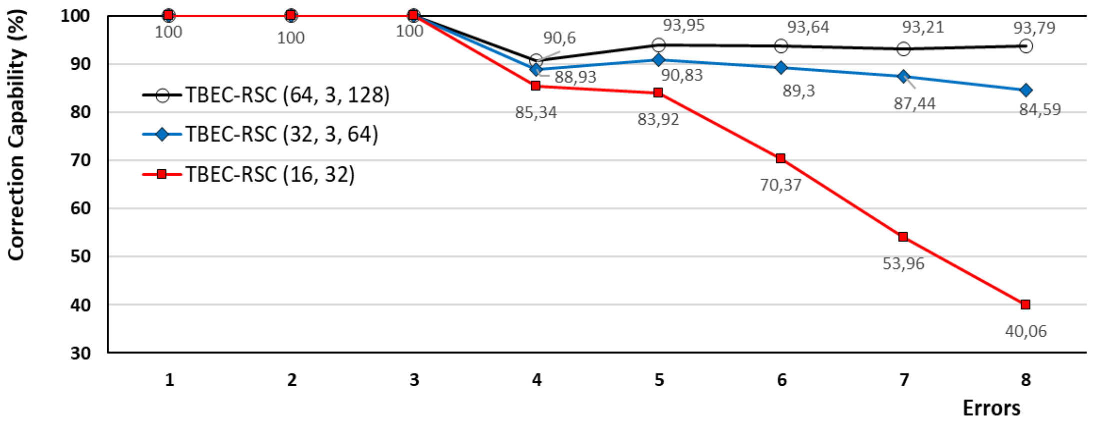
Source: Author.

Note that the code methodology applied in TBEC-RSC is not linear, which is used in FUEC codes (GRACIA-MORÁN et al, 2018) and TBEC-QUAEC (LI et al, 2018), but only the implemented format. Linear codes have a correction table predefined by the generating matrix that encodes the data, causing the correction rates to drop significantly when errors not considered in the formation of the matrix appear in the coded word, as can be seen in the correction results of the FUEC and TBEC-QUAEC codes.

As presented in Section 4.3, TBEC-RSC was designed to use the region logic originally proposed by MRSC but adopted the linear codeword format. For example, for 5-error scenarios, the TBEC-RSC still had a correction above 80%, while the proposed linear codes had results close to or less than 10%.

Figure 46 shows the error correction comparison of TBEC-RSC versions implemented with 16, 32, and 64 data bits. As expected, all versions fixed scenarios with three or fewer errors. Besides, the 32- and 64-bit versions achieved a higher error correction rate for a more significant number of errors, as stated with eMRSC(32, 3, 64), those codes have bigger regions, which increases the correction capacity.

Figure 46 – TBEC-RSC: Scaling analysis.



Source: Author.

5.1.4 Correction coverage evaluation: Conclusions

The analyzes showed that the codes proposed in this thesis were able to achieve excellent results when compared to other techniques. Region logic provides a simple and efficient way to detect and correct errors. For both adjacent errors (MRSC and eMRSC) and burst errors (TBEC-RSC), the codes performed well and were leaders in correction in most error scenarios. The extensions of the original MRSC logic (eMRSC and TBEC-RSC), brought contributions and adjustments that allowed more dynamic options and new formats for using the code.

5.2 RELIABILITY EVALUATION

This section presents reliability and Mean Time to Failure (MTTF) evaluations considering that each MCU is generated from a single event effect. This experiment was performed following the definitions employed in the works (SILVA et al. 2020) and (ARGYRIDES et al. 2007). The following concepts will serve to enlighten the theory and equations utilized to perform this estimation. Let w and c be the data and redundancy bits of each ECC such that $(w + c)$ is the codeword size, λ be the error rate, and t be the time in days associated with λ; then, Equation 5.2 calculates the $pEM(t)$, which is the probability of having a codeword with errors at time t. The value of λ depends on memory technology and the environment in which the device is inserted.

$$pEM(t) = 1 - e^{-(w+c)\lambda t} \tag{5.2}$$

Let i be the number of errors; then, Equation 5.3 estimates the probability of occurring i errors in a codeword of $(w + c)$ bits at time t, represented by $pEi(t)$.

$$pE_i(t) = \binom{w+c}{i} \times (1 - e^{-\lambda t})^i \times e^{-\lambda(w+c-i)t} \tag{5.3}$$

Equation 5.4 calculates $r(t)$, which is the reliability of a codeword at time t. $pCMi$ is the correction probability for i errors extracted from the results presented in the correction analysis. Let Me be the maximum number of errors that can arise (our experiments consider Me = 8); then, the expression $\sum_{i=1}^{Me} pE_i(t) \times pCM_i$ is the probability of error occurrence on a codeword that the evaluated ECC can correct at time t. Moreover, $(1 - pEM(t))$ is the probability of not having errors in a codeword over time t.

$$r(t) = 1 - pEM(t) + \sum_{i=1}^{Me} pE_i(t) \times pCM_i \tag{5.4}$$

Let M be the number of codewords in memory; then, Equation 5.5 calculates $R(t)$, which is the memory device reliability at time t, through the product of the reliabilities of all codewords.

$$R(t) = r(t)^M \tag{5.5}$$

Finally, Equation 5.6 estimates the MTTF of a memory device protected by an ECC through the $R(t)$ integration over time t.

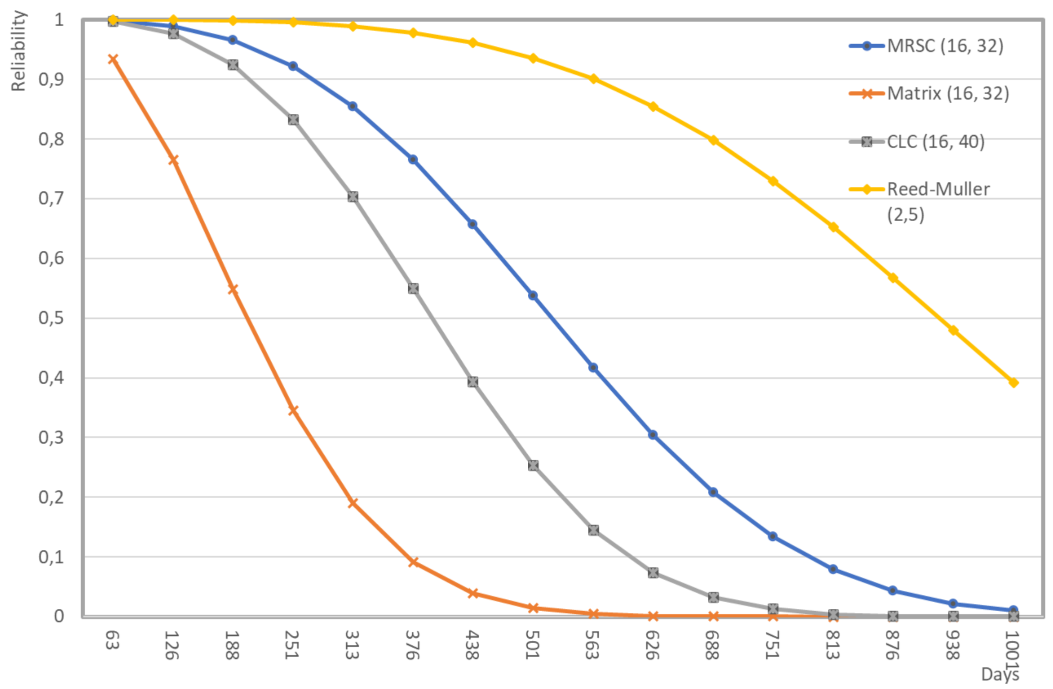
$$MTTF = \int_0^\infty R(t)\,dt \qquad (5.6)$$

To estimate the reliability results, a MATLAB script was designed to implement the equations presented in this section. For each ECC approach and the compared techniques, we estimated the $R(t)$ considering memories of configuration $M \times (w + c)$. We considered the results for M equals 1. The results of the estimation for each proposed ECC in this thesis and the ones utilized in comparison are presented as follows.

5.2.1 MRSC: Reliability analysis

Figure 47 shows the reliability curve estimated by Equations 5.2 to 5.5, and Table 15 exhibits the MTTF results calculated by Equation 5.6, varying the λ used in estimation.

Figure 47 – MRSC: Estimated $R(t)$ graph (λ = 10^{-5}, M = 4096).



Source: Author

Table 15 – MRSC: MTTF results (M = 4096).

λ	MRSC (16,32)	Matrix (16,32)	CLC (16,40)	Reed-Muller (2,5)
10^{-4}	52.67	21.56	40.09	92.16
10^{-5}	526.05	215.61	400.94	847.05

Source: Author

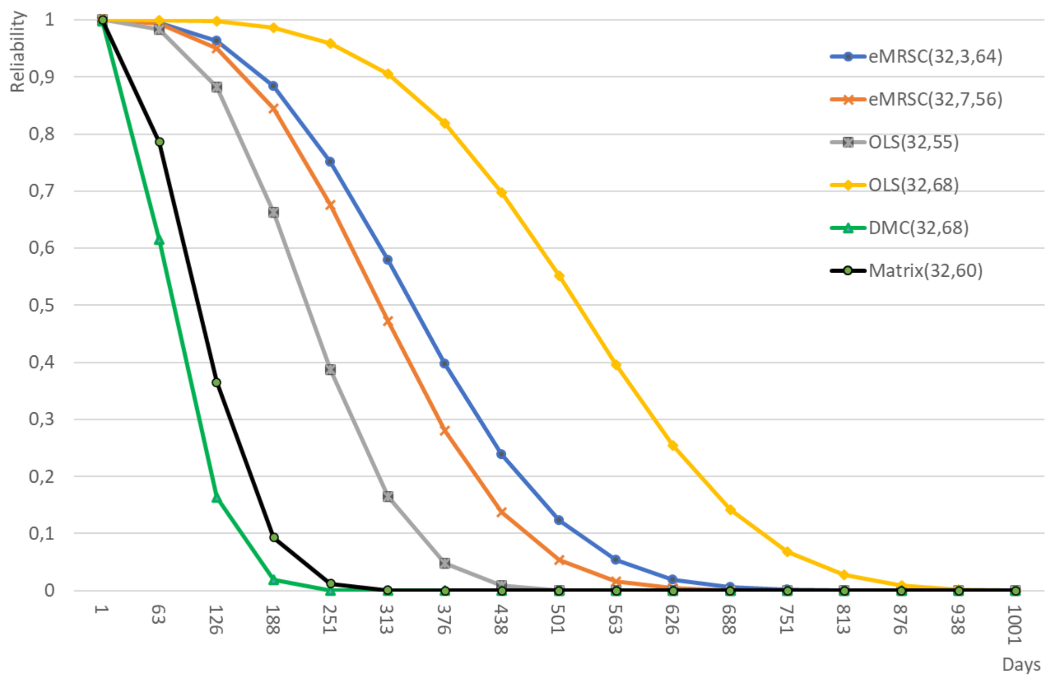
The joint analysis of Figure 47 and Table 15 demonstrates that the MRSC reliability surpassed Matrix and CLC. It happens because Matrix has less correction capacity than MRSC, decreasing the reliability value, and the additional redundancy of CLC increases the number of bits to be affected within a memory register, increasing the possibility of non-detectable cases.

Compared with Reed-Muller (2, 5), MRSC had much lower reliability results, despite MRSC having presented superior correction results for more than four errors than all other ECCs. The 100% effectiveness in correcting 3 errors is the main reason for Reed-Muller (2, 5) to have surpassed MRSC.

5.2.2 eMRSC: Reliability analysis

Figure 48 shows the reliability curve estimated by Equations 5.2 to 5.5 and Table 16 shows the MTTF results calculated by Equation 5.6, varying the λ used in estimation.

Figure 48 – eMRSC: Estimated $R(t)$ graph (λ = 10^{-5}, M = 4096).



Source: Author

Table 16 – eMRSC: MTTF results (M = 4096).

λ	eMRSC(32,3,64)	eMRSC(32,7,56)	OLS(32,55)	OLS(32,68)	DMC(32,68)	Matrix(32,60)
10^{-4}	34.55	30.90	22.83	52.06	3.52	10.97
10^{-5}	345.46	309	228.28	520.66	42.54	110.74

Source: Author

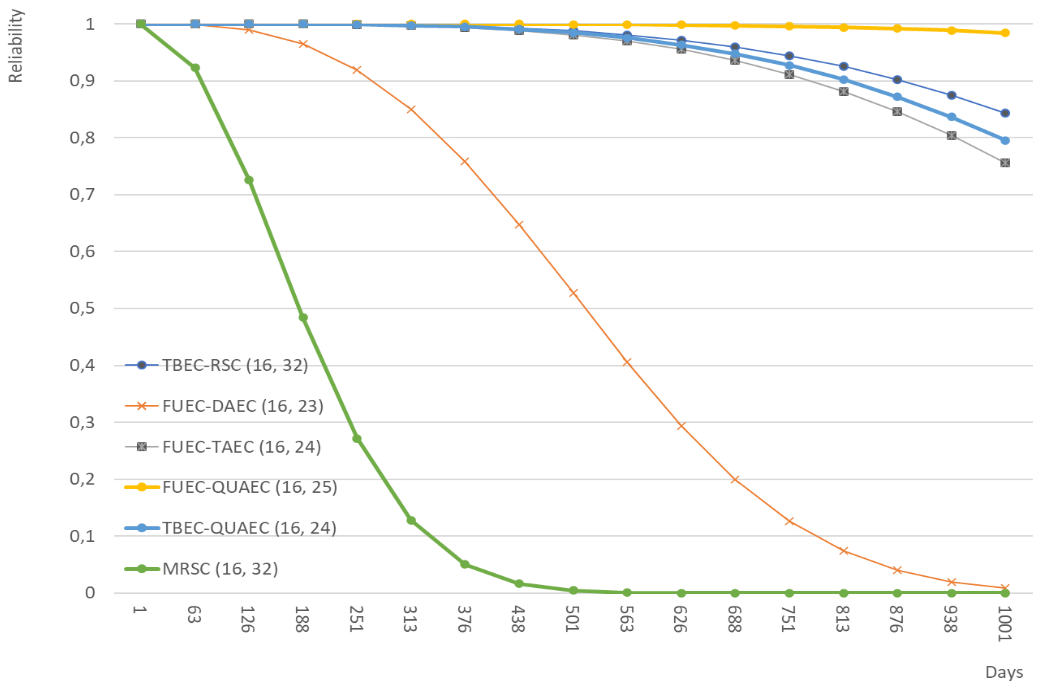
Matrix, and especially DMC, showed lower MTTF results than the others. The DMC results are justified by the shallow correction capacity provided by the code since this code only corrects specific error patterns. Matrix, which was also analyzed in the previous experiment, has a lower correction rate than the others. Looking at the more robust codes, both versions of eMRSC achieved superior results to OLS(32,55), with eMRSC(32,3,64) ahead of eMRSC(32,7,56) by just over 10%. The superior correction capability of the more robust version of eMRSC has more weight than the extra redundancy in the reliability calculation.

Compared to OLS(32,68), the proposed codes cannot surpass the reliability results. A scenario like the one found in Section 5.2.1 is observed. In most observed scenarios, eMRSC(32,3,64) has a higher correction rate than OLS(32,68) and fewer redundancy bits, but its reliability is lower. OLS(32.68) is also a triple error corrector, as is Reed-Muller(2.5), and this ended up increasing the top of the reliability curve, causing its calculated value to be higher.

5.2.3 TBEC-RSC: Reliability analysis

Figure 49 shows the reliability curve estimated by Equations 5.2 to 5.5, and Table 17 shows the MTTF results calculated by Equation 5.6, varying the λ used in the estimation.

Figure 49 – TBEC-RSC: Estimated $R(t)$ graph ($\lambda = 10^{-5}$, M = 4096).



Source: Author

The results of Figure 49 and Table 17 reveal that TBEC-RSC presents the second-highest reliability among the analyzed ECCs, achieving nearly 35% lower MTTF and a more flattened R(t) curve than FUEC-QUAEC, the most reliable ECC. Besides, the chart kept the same structure for all memory sizes.

Table 17 – TBEC-RSC: MTTF results (M = 4096).

λ	MRSC(16,32)	TBEC-RSC(16,32)	FUEC-DAEC(16,23)	FUEC-TAEC(16,24)	FUEC-QUAEC(16,25)	TBEC-QUAEC(16,24)
10^{-4}	9.6	100.3	32.5	88.00	164.6	92.8
10^{-5}	98.6	1003.6	324.5	880.04	1645.7	927.8

Source: Author

The redundancy bits significantly influence the reliability results, although the correction efficacy also weighs positively. From the results in Figure 49, FUEC-QUAEC and TBEC-RSC offer the same error correction efficacy from 1 to 3 errors. TBEC-RSC has about 15% lower efficacy for four errors, although TBEC-RSC surpasses FUEC-QUAEC by a higher margin from 5 to 8 errors (above 40%); still, the MTTF estimated ranks FUEC-QUAEC as the most reliable ECC, mainly due to the 4-bit error correction of this code being 100%. Regarding the other approaches, MRSC and FUEC-DAEC presented the lowest results on this analysis since they are the ECCs with the lowest error correction capability.

5.2.4 Reliability analysis: Conclusions

From the results collected and according to the theory presented, an ECC with the highest overall correction is only sometimes the most reliable. This may lead to a biased analysis of what ECC is more suitable for critical applications or which presents the best trade-off. Comparing FUEC-QUAEC and TBEC-RSC from the previous experiment, we saw that the first corrected 100% of the 4-bit error cases, while TBEC-RSC has a correction of around 85%. When we observe the error coverage for 5-bit error, TBEC-RSC corrects nearly the same percentage of error patterns, while FUEC-QUAEC corrects less than 15%. We stated before that with the shrinking of the technology, more aggressive error patterns are more likely to happen. In this context, TBEC-RSC is the best option for critical applications. This same thinking can be applied to the other two analyses considering the MRSC and eMRSC versions. Even with that, considering the appliance of the proposed codes on Cubesat missions, they presented results that allow them to be applicable in low-orbit satellites with a mission duration of up to 1-year (QUIAO, 2013).

Furthermore, the MTTF analysis does not include the operating costs achieved in the synthesis. The design of an ECC, especially for those focused on critical applications with low resources, must consider the overall cost that the technique brings in encoding and decoding procedures.

5.3 SYNTHESIS COST AND TRADE-OFF EVALUATION

On the one hand, the previous results showed that the techniques presented in this thesis have a better correction performance than the works used as a comparison. On the other hand, the reliability was slightly below some techniques with less corrective capacity in general, but which stand out in specific error patterns. Another relevant data for the ECC analysis is the operating cost obtained by hardware synthesis.

In the current context of integrated circuit scaling, each μm^2 is essential to allow more logic to be inserted into an SoC. In addition, material and computational resources are limited in critical applications, such as space missions, requiring low-power circuits. Finally, time is also a determining factor for a successful mission, in which fast information processing and a low retransmission rate are vital. This section discusses the synthesis and trade-offs of our work with the other works mentioned in Table 13 .

We synthesized the ECC decoders described in this work to compare the operating costs. We decided to only review the decoders for the following reasons Table 13 :

- Most works in the literature consider the synthesis of decoders to be the most relevant part to be analyzed in an ECC.
- The decoder encompasses the encoding logic and error correction algorithm; i.e., the decoder needs to redo the encoding steps to detect the syndromes (identify the error type and position), so the correction logic can be applied.

The ECCs used in experiments were written using the Verilog hardware description language, and the tool chosen to carry out the synthesis process was the Genus Synthesis Solution. All ECCs were modeled as combinational circuits, not dependent on the clock signals or processing states.

Equation 5.6 describes the *Correction per Cost* (CSC) metric, which allows us to evaluate the relationship between the error correction efficacy and the ECC operation efficiency, such that the efficacy is obtained in the designed simulation, and the efficiency is acquired in the ECC syntheses, which is computed by Equation 5.7.

$$CSC = \frac{Correction}{Synthesis\ Cost} \tag{5.6}$$

$$Synthesis\ Cost = \ Area\ x\ Power\ x\ Delay \tag{5.7}$$

This metric became known in the work of Argyrydes et al. (2009), having been used by many other works in the area and presented in alternative versions. Next, the synthesis results and the trade-off analysis of the proposed codes are presented. For a better appreciation of the trade-off results, each experiment had its results normalized to the highest CSC.

5.3.1 MRSC: Synthesis results and trade-off evaluation

Table 18 presents the MRSC synthesis results, and the codes used in its comparison for a 65nm CMOS technology, where the percentage represents the normalization of the results for the ECC with the highest cost within the analysis.

Table 18 – MRSC: Synthesis of decoders (65nm).

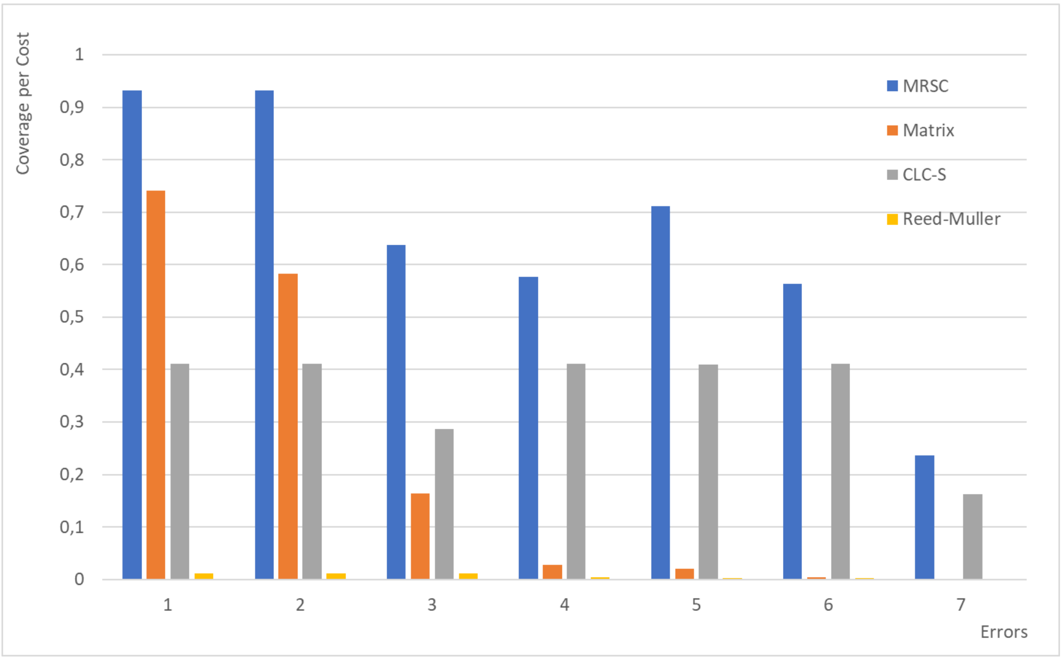
ECC	**Area**		**Power**		**Delay**	
	(μm²)	**(%)**	**(mW)**	**(%)**	**(ns)**	**(%)**
Reed-Muller (2,5)	4312	100.0	0.737	100.0	2.124	100.0
CLC	1351	31.3	0.076	10.3	1.326	62.4
Matrix	1210	28.1	0.059	8.0	1.264	59.5
MRSC	931	21.6	0.065	8.8	1.055	49.7

Source: Author

Table 18 depicts that Reed-Muller (2, 5) and MRSC have the highest and lowest overall overhead costs for the decoders. Although the MRSC power dissipation is nearly 10% bigger than Matrix, its area and delay consumptions are 23% and 16.6% smaller, respectively. CLC has a larger quantity of redundancy bits of all codes analyzed as well as a more complex logic than MRSC and Matrix, which justifies their cost results. Although Reed-Muller (2, 5) presents the same number of redundancy bits of MRSC and Matrix, its majority logic structure is very complex and expensive to implement in hardware designs.

Figure 50 presents the CSC results of four codes. It shows that MRSC performs better than all other ECCs proposed for all fault scenarios due to its lightweight implementation cost and high error coverage. Matrix obtained fine CSC results for one and two faults. Matrix code shows lower error correction and detection rates from three to seven faults compared to the other approaches. Indeed, CLC is mainly focused on aggressive error patterns, which surpassed Matrix for more than three errors in the metric. The significant cost discrepancy between MRSC and CLC represented a major advantage for the first since MRSC presented detection and correction rates near CLC, justifying the better results achieved by MRSC. Finally, Reed-Muller (2, 5) achieved the lowest $CSCs$ in all experiments due to its highest synthesis cost.

Figure 50 – CSC Results.



Source: Author.

5.3.2 eMRSC: Synthesis Results and Trade-off Evaluation

Table 19 presents the results of the eMRSCs and OLS synthesis for a 65nm CMOS technology, in which the percentage represents the normalization of the results for the ECC with the highest cost within the analysis.

Table 19 – eMRSC: Synthesis of decoders (65nm).

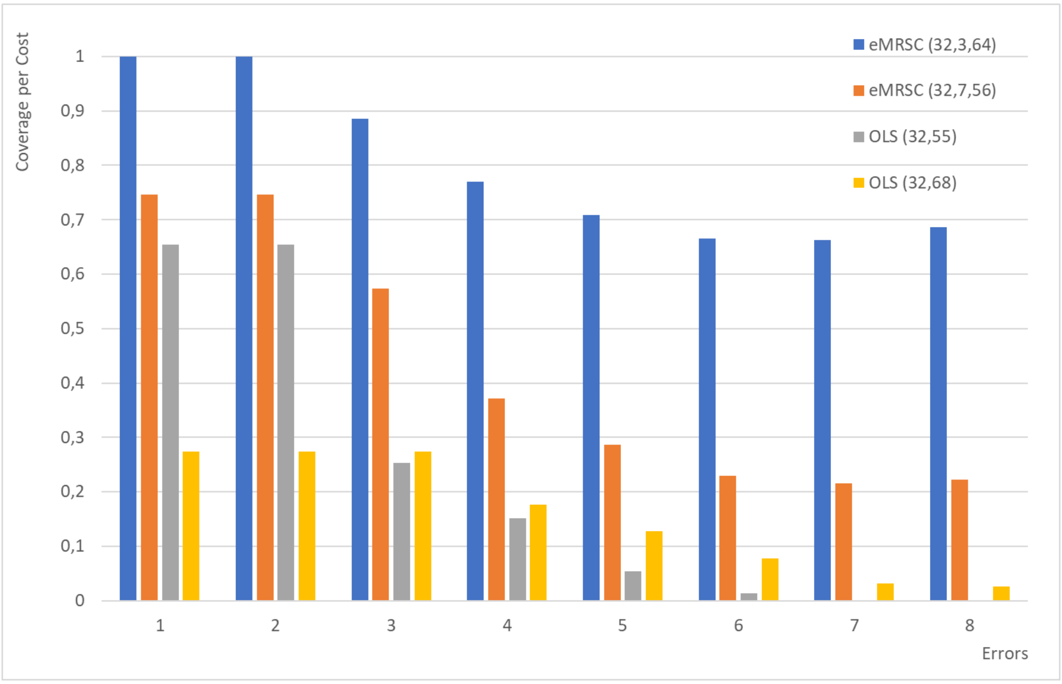
ECC	Area		Power		Dela	
	(µm²)	(%)	(mW)	(%)	(ns)	(%)
eMRSC(32,3,64)	1709	43.3	0.374	52.8	1.54	100.0
eMRSC(32,7,56)	1623	41.1	0.304	42.9	1.49	96.7
OLS(32,68)	3944	100.0	0.708	100.0	0.96	62.3
OLS(32,55)	2541	64.4	0.486	68.6	0.91	59.1

Source: Author

DMC(32,68) and Matrix(32,60) were not included in this section because the DMC reliability results are not enough for dealing with MCUs, and Matrix was previously evaluated in the MRSC comparison. Table 19 illustrates that the decoder synthesis of eMRSC(32,3,64) achieved higher values than eMRSC(32,7,56) but without a meaningful difference. On the one hand, eMRSC(32,7,56) has fewer redundancy bits, reducing the total cost; conversely, the logic to correct errors is more complex than the first eMRSC, which increases the cost. Compared to the OLS model, eMRSC has fewer costs of area consumption and power dissipation on the decoder synthesis (over 50% less area for both eMRSC against the highest OLS code). Additionally, the decoder synthesis depicts that the delays of both OLS codes are smaller than the delays of the eMRSC codes, which is explained by the fact that the detection and correction process of OLS is more straightforward than eMRSC. OLS uses an enhanced majority logic to evaluate each bit and determine the correct value. Although this approach can be slightly faster, this method has a high implementation cost.

Figure 51 displays that the eMRSC codes have the best trade-off between correction rate and synthesis cost. Besides, from 1 to 8 errors the eMRSC(32,3,64) achieved the best CSC results overall. In more detail, the eMRSC(32,3,64) correction rates surpass eMRSC(32,7,56) by a considerable margin from four to eight errors (more than 32%), which impacted the CSC analysis. Although OLS(32,68) has correction rates higher than eMRSC(32,7,56), the synthesis cost of the first is much higher than the second; consequently, the CSCs of both OLS codes never reach the $CSCs$ of the eMRSC codes.

Figure 51. eMRSC: CSC Results.



Source: Author.

5.3.3 TBEC-RSC: Synthesis Results and Trade-off Evaluation

Table 20 presents the synthesis results of the TBEC-RSC and the codes used in its comparison for a 65nm CMOS technology; the percentage represents the normalization of the results for the ECC with the highest cost within the analysis.

Table 20 – TBEC-RSC: Synthesis of decoders (65nm).

ECC	Area		Power		Delay	
	(µm²)	(%)	(mW)	(%)	(ns)	(%)
TBEC-RSC	674	27.3	0.059	20.2	1.10	63.9
MRSC	717	29.1	0.059	20.2	1.23	71.5
FUEC-DAEC	829	33.6	0.053	18.2	1.21	70.3
FUEC-TAEC	1221	49.5	0.062	21.3	1.26	73.2
FUEC-QUAEC	1704	69.1	0.077	26.4	1.38	80.2
TBEC-QUAEC	2466	100.0	0.291	100.0	1.72	100.0

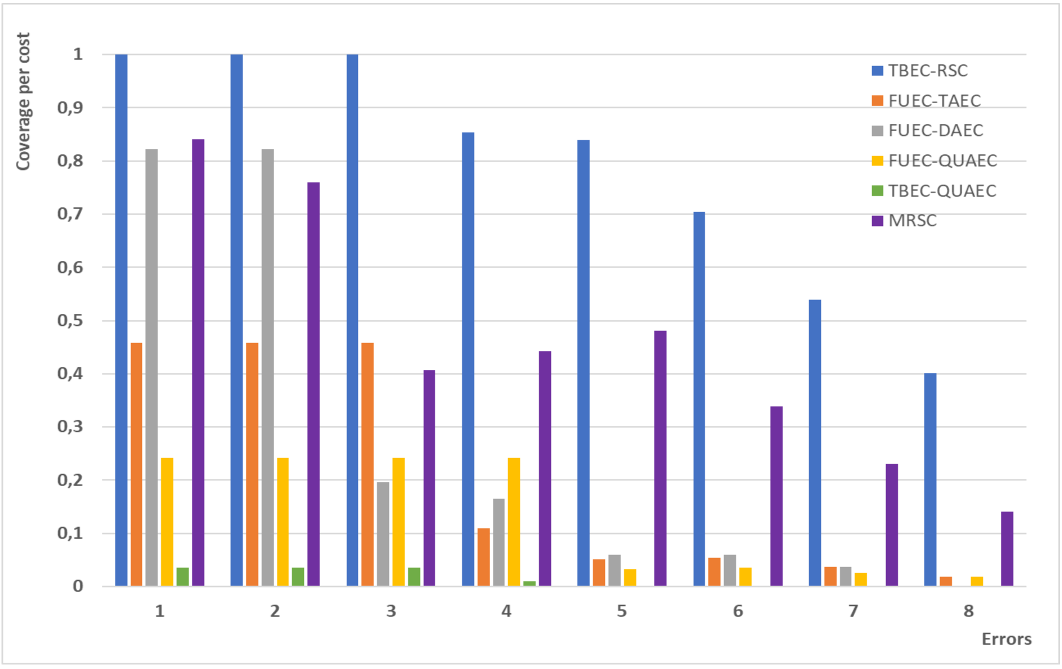
Source: Author.

The decoding methodology applied in TBEC-RSC is based on the MRSC algorithm that requires simple logic equations to detect and correct errors. Consequently, TBEC-RSC only has a higher synthesis cost compared to the least robust FUEC code (FUEC-DAEC) and the

MRSC original version, which for the last is justified by the addition of the new conditions applied to fix burst errors (less than 10% overall cost increase). At the same time, TBEC-RSC has the highest overall error correction rate among all evaluated codes. The ECCs that employ linear coding, FUEC codes, and TBEC-QUAEC use a syndrome matchup table; thus, as the code correction capacity rises, the amount of logic required increases, increasing the area consumption, power dissipation, and delay.

Figure 52 shows that except for one and two error patterns, TBEC-RSC reaches higher CSC results, especially from three to eight errors. Despite FUEC-DAEC presenting lower synthesis costs than TBEC-RSC, its error correction rate decreases significantly. FUEC-TAEC, FUEC-QUAEC, and TBEC-QUAEC have operating costs far superior to TBEC-RSC, about 647%, 947%, and 2519%, respectively, which diminished their CSC results. Since all codes, except MRSC and TBEC-RSC, require a matchup table to correct errors, the correction capability reduction in those codes considerably diminishes the $CSCs$.

Figure 52 – TBEC-RSC CSC Results.



Source: Author.

5.3.4 Synthesis results and trade-off evaluation: Conclusions

This section shows the synthesis results and the evaluation of the trade-off between correction capacity and synthesis cost. Compared to all other techniques, the codes proposed in this thesis had excellent synthesis results, presenting less operating costs. Unlike other approaches, which use linear logic and are compositions of linear codes with parity, our proposals use only parity and appropriate positioning of the bits in their coding. Analyzing the eMRSC and TBEC-RSC codes, the proposed changes did not significantly impact the cost analysis, given that both the eMRSC codes and TBEC-RSC with MRSC had similar costs.

Regarding the CSC results, the proposed codes generally obtained the best evaluation results since they also perform well in the correction analysis and have a low synthesis cost. However, the analysis must be carefully evaluated to avoid bad design choices. Figure 52 displays that the linearized version of MRSC performs much better than the other linear codes, except for TBEC-RSC. This does not mean that MRSC is more suitable for multiple errors than the FUEC-TAEC and FUEC-QUAEC codes, for example (taking correction rates into account); however, the error corrected by MRSC consumes less computational resources than the mentioned codes.

6 CONCLUSION AND FUTURE WORK

This work presented the MRSC code and its derivative eMRSC and TBEC-RSC as ECCs for critical applications. The proposed codes use RSA, an original proposal that stands out for bringing high efficacy in correcting multiple errors and low computational cost compared to other approaches.

The first version of MRSC is a 16-bit code, which uses a matrix format (2-D) and corrects bits by selecting the region with the highest number of adjacent errors detected. The eMRSC(32, 3, 64) and eMRSC(32,7,56) codes, also in matrix format, showed that larger data words enable to vary both the error correction capacity and the amount of redundancy generated, without reducing the error correction efficacy in adjacent bits. The TBEC-RSC implementation changes from matrix (2-D) to linear format (1-D), in addition to specific adjustments in the bit positioning and some equations, enabling to correct burst error patterns. The proposed codes had their encoding and decoding processes exemplified in detail to provide a better understanding of the entire cycle.

We compared the proposed codes with well-known works published in high-impact journals and conferences using the following experiments: (I) Correction coverage analysis, (II) Reliability analysis, and (III) Synthesis cost and performance trade-off analysis.

In the first evaluation, the MRSC and eMRSC codes were tested against adjacent error patterns, while TBEC-RSC was evaluated with burst errors. Both for adjacent and burst errors, the codes proposed in this Thesis had an overall better performance than the works used in the comparison, especially in multiple errors (more than four errors).

In the second evaluation, the reliabilities of the proposed codes were only below the most robust ECC used. Among the factors that explain this drop is that the MTTF methodology penalizes reliability estimation when the correction rate for an error standard does not reach 100%. Even so, the proposed codes presented results compatible with what is necessary for application in critical missions such as CubeSats.

Finally, the third evaluation brings both the ECC synthesis results and a cost-benefit relationship between the correction of codes for each error pattern and its total operating costs. Whereas the proposed codes use the logic of regions, which provides a fast and economical detection, the other approaches use linear logic, which involves using several mathematical and majority logic equations, bringing higher synthesis costs. Therefore, this analysis attested to the MRSC, eMRSC, and TBEC-RSC codes that they are excellent cost-effective ECCs for critical applications, presenting the best overall analysis results.

6.1 FUTURE WORK

The MRSC code and its expansions, eMRSC and TBEC-RSC, brought several original and impactful contributions to the ECC research field. Despite what was exposed in the previous section, this section tries to increase the scope of possible expansions and analysis of these codes as future works.

Chapter 4 described that the eMRSC codes expand the first MRSC version. eMRSC encodes 32 bits, and this expansion produced eMRSC (32,7,56) and eMRSC (32,3,64) versions. The first version reduced the redundancy bits generated by 25% (24 bits produced), which could be interesting in applications that demand high data rate transmission. The logic applied behind the expansion for a 32-bit code could be utilized for even higher data bits (e.g., 64-bit and 128-bit), which would decrease the redundancy bit rate even more. Considering that the minimum region of MRSC should be a 4×2 matrix, a 64-bit version would have 15 regions, and the code would be an eMRSC (64,15,104) which would provoke a reduction of 37.5% in redundancy from the version eMRSC (64,3,128).

Furthermore, it was stated that increasing the region designed in an MRSC would increase the adjacent error correction rate. Considering the eMRSC (64,3,128) version, the region would be a 4×8 matrix, twice the size of the eMRSC(32,3,64). It was shown that the main problems of the MRSC and eMRSC codes are situations where the errors between regions are divergent or do not result in a tie. However, the probability of these situations decreases with bigger regions being applied. Therefore, it is expected that for eMRSC(64,3,128), the correction rates would be even higher (eMRSC(32,3,64) corrected at least 60% of the errors overall). TBEC-RSC could benefit from the findings in eMRSC(32, 7, 56) and have a version with lower redundancy. For this, it would be necessary to understand potential changes in bit positioning and the equations. In resume, MRSC still has several investigations and improvements that can be conducted in future work.

REFERENCES

ABBE, E; SHPILKA, A; YE, M. Reed–Muller Codes: theory and algorithms. **IEEE Transactions on Information Theory**, Urbana, v. 67, n. 6, p. 3251–3277, 2021.

ARGYRIDES, C.; REVIRIEGO, P.; PRADHAN, D. Matrix-based codes for adjacent error correction. **IEEE Transactions on Nuclear Science**, Oak Ridge, v. 57, n. 4, p. 2106–2111, 2010.

ARGYRIDES, C.; ZARANDI, H.; PRADHAN, D. Matrix codes: multiple bit upsets tolerant method for sram memories. In: **IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems**, 22., 2007, Rome. Anais … IEEE, 2007, p. 340–348.

BAEG, S.; WEN, S.; WONG, R. SRAM interleaving distance selection with a soft error failure model. **IEEE Transactions on Nuclear Science**, Oak Ridge, v. 56, n. 4, p. 2111–2118, 2009.

BOSSER, A. L. Single-event effects of space and atmospheric radiation on memory components. 2017. Thesis, (P.h.D in Micro and nanotechnologies/Microelectronics), **University of Jyväskylä**, Finland, 2017.

BAUMANN, R. C. Soft errors in commercial integrated circuits. **International Journal of High Speed Electronics and Systems**, Troy, v. 14, n. 02, p. 299–309, 2004.

BROWNING, J. S.; KOGA, R.; KOLASINSKI, W. A. Single Event Upset Rate Estimates for A 16-K CMOS SRAM. **IEEE Transactions on Nuclear Science**, Oak Ridge, v. 32, n. 6, p. 4113-4119, 1985.

CARDARILLI, G. C.; OTTAVI, M.; PONTARELLI, S.; RE, M.; SALSANO, A. Fault-tolerant solid-state mass memory for space applications. **IEEE Transactions on Aerospace and Electronic Systems**, Cranfield, v. 41, n. 4, p. 1353–1372, 2005.

CASTRO, H.; SILVEIRA, J.; COELHO, A.; SILVA, F.; MAGALHÃES, P.; LIMA, O. A Correction Code for Multiple Cells Upsets in Memory Devices for Space Applications. In: **IEEE International New Circuits and Systems Conference (NEWCAS)**, 14., 2016, Vancouver. Anais … IEEE, 2016, p. 1-4.

CHEN, C. L.; HSIAO, M. Y. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. **IBM Journal Of Research And Development**, Yorktown Heights, v. 28, n. 2, p. 124–134, 1984.

CHOI, S. et al. A decoder for short BCH codes with high decoding efficiency and low power for emerging memories. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, Charlottesville, v. 27, n. 2, p. 387-397, 2019.

CHUGG, A. M.; MOUTRIE, M. J.; JONES, R. Broadening of the variance of the number of upsets in a read-cycle by MBU's. **IEEE Transactions on Nuclear Science**, Oak Ridge, v. 51, n. 6, p. 3701–3707, 2004.

DAS, A.; TOUBA, N. A new class of single burst error correcting codes with parallel decoding. **IEEE Transactions on Computers**, Athens, v. 69, n. 2, p. 253-260, 2020.

DAS, A.; TOUBA, N. Online correction of hard errors and soft errors via one-step decodable OLS codes for emerging last level caches. In: **Latin American Test Symposium (LATS)**, 2019, Santiago. Anais … IEEE, 2019, p. 1-6.

DUTTA, A.; TOUBA, N. A. Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code. In: **IEEE VLSI Test Symposium**, 25., 2007 Berkley. Anais … IEEE, 2007, p. 349–354.

FERREYRA, P.; MARQUES, C.; FERREYRA, R.; Gaspar, J. 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 Transactions on Nuclear Science**, Oak Ridge, v. 52, n. 1, p. 494–500, 2005.

FREITAS, D.; MOTA, D.; MARCON, C.; SILVEIRA, J.; MOTA, J. LPC: An error correction code for mitigating faults in 3d memories. **IEEE Transactions on Computers**, Athens, v. 70, n. 11, p. 2001-2013, 2021.

GAILLARD, R. **Soft errors in modern electronic systems**, New York: Springer, v. 41, p. 26-53, 2011.

GARCIA-HERRERO, F.; SÁNCHEZ-MACIÁN, A.; MAESTRO, J. Low delay non-binary error correction codes based on Orthogonal Latin Squares. **Integration**, California, v. 76, p. 55-60, 2020.

GARG, R.; KHATRI, S. P. **Analysis and design of resilient VLSI circuits**, New York: Springer, p. 8-9, 2009.

GHERMAN, V.; EVAIN, S.; AUZANNEAU, F.; BONHOMME, Y. Programmable extended SEC-DED codes for memory errors. In: **IEEE VLSI Test Symposium (VTS)**, 29., 2011, Dana Point. Anais … IEEE, 2011, p. 140–145.

GRACIA-MORÁN, J.; SAIZ-ADALID, L.; GIL-TOMÁS, D. Improving error correction codes for memory-cell upsets in space applications. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, Charlottesville, v. 26, n. 10, p. 2132-2142, 2018.

GUENZER, C. S.; CAMPBELL, A. B.; SHAPIRO, P. Single Event Upsets in NMOS Microprocessors. **IEEE Transactions on Nuclear Science, Oak Ridge,** v. 28, n. 6, p. 3955-3958, 1981.

HAMMING, R. W. Error detecting and error correcting codes. **Bell System Technical Journal**, New York, v. 29, n. 2, p. 147–160, 1950.

HAZUCHA, P.; SVENSSON, C. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. **IEEE Transactions on Nuclear Science**, Oak Ridge, v. 47, n. 6, p. 2586–2594, 2000.

HEIJMEN, T. **Soft errors in modern electronic systems**, New York: Springer, v. 41, p. 1-25, 2011.

HSIAO, M.; BOSSEN, D.; CHIEN, R. Orthogonal Latin Square Codes. **IBM Journal of Research and Development**, Yorktown Heights, v. 14, n. 4, p. 390-394, 1970.

JOHANSSON, K. et al. In-flight and ground testing of single event upset sensitivity in static RAMs. **IEEE Transactions on Nuclear Science**, Oak Ridge, v. 45, n. 3, p. 1628-1632, 1998.

KASTENSMIDT, F. L.; CARRO, L.; REIS, R. **Fault-tolerance techniques for SRAM-based FPGAs**, New York: Springer, v. 1, p. 29-73, 2006.

KUMAR, A.; VARMA, A. **Silicon on Insulator (SOI) Sign Off Semiconductors**. Disponível em: https://signoffsemiconductors.com/silicon-on-insulator-soi. Acesso em: 11/12/2022.

LABEL, K. A.; BARNES, C. E.; MARSHALL, C. J.; JOHNSTON, A. H.; REED, R. A.; BARTH, J. L.; SEIDLECK, C. M.; KAYALI, S. A.; O'BRYAN, M. V. A roadmap for NASA's radiation effects research in emerging microelectronics and photonics. In: **IEEE Aerospace Conference Proceedings**, 5., 2000, Big Sky. Anais … IEEE, 2000, p. 535–545.

LABEL, K. NASA **Past, present and future: The use of Commercial Off the Shelf (COTS) Electronics in space**. Disponível em: https://ntrs.nasa.gov/citations/20180002201. Accesso em: 03/05/2022.

LAI, Y.; CHEN, L.; CHIOU, W. A memory interleaving and interlacing architecture for deblocking filter in H.264/AVC. **IEEE Transactions on Consumer Electronics**, Hong Kong v. 56, n. 4, p. 2812-2818, 2010.

LI, J.; REVIRIEGO, P.; XIAO, L. Low delay 3-bit burst error correction codes. **Journal of Electronic Testing**, New York, v. 35, n. 3, p. 413-420, 2019b.

LI, J.; REVIRIEGO, P.; XIAO, L.; ARGYRIDES, C.; LI, J. Extending 3-bit burst error-correction codes with quadruple adjacent error correction. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, Charlottesville, v. 26, n. 2, p. 221-229, 2018a.

LI, J.; REVIRIEGO, P.; XIAO, L.; LIU, Z.; LI, L.; ULLAH, A. Low delay single error correction and double adjacent error correction (SEC-DAEC) Codes. **Microelectronics Reliability**, Maryland, v. 97, p. 31-37, 2019a.

LI, J.; XIAO, L.; GUO, J.; CAO, X. Efficient implementations of multiple bit burst error correction for memories. In: **IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)**, 2018, Qingdao. Anais … IEEE, 2018b, p. 1-3.

LIU, S.; LI, J.; REVIRIEGO, P.; et al. A double error correction code for 32-bit data words with efficient decoding. **IEEE Transactions on Device and Materials Reliability**, Piscataway, v. 18, n. 1, p. 125-127, 2018a.

LIU, S.; REVIRIEGO, P.; MAESTRO, J. A.; XIAO, L. Fault tolerant encoders for single error correction and double adjacent error correction codes. **Microelectronics Reliability**, Maryland, v. 81, p. 167-173, 2018b.

LIU, S.; REVIRIEGO, P.; XIAO, L.; MAESTRO, J. Reducing the cost of triple adjacent error correction in double error correction Orthogonal Latin Square Codes. **IEEE Transactions on Device and Materials Reliability**, Piscataway, v. 16, n. 2, p. 269-271, Jun. 2016b.

LIU, S.; XIAO, L.; LI, J.; ZHOU, Y.; et al. Low redundancy matrix-based codes for adjacent error correction with parity sharing. In: **International Symposium on Quality Electronic Design (ISQED)**, 18., 2017, Santa Clara. Anais … IEEE, 2017, p. 1-6.

LIU, S.; XIAO, Y.; MAO, G. Extend Orthogonal Latin Square codes for 32-bit data protection in memory applications. **Microelectronics Reliability**, Maryland, v. 63, p. 278-283, 2016a.

MOON, T. **Error correction coding, mathematical methods and algorithms**. New Jersey: Wiley, 2005.

MOREIRA, J.; FARRELL, P. **Essentials of error control coding**. New Jersey: Wiley, 2006.

NEUBAUER, A.; FREUDENBERGER, J.; KÜHN, V. **Coding theory: algorithms, architectures, and applications**. New Jersey: Wiley 2007.

NORMAND, E. Single event effects in avionics and on the ground. **International Journal of High-Speed Electronic Systems**, Troy, v. 14, n. 2, p. 285–298, 2004.

O'GORMAN, T. J. The Effect of Cosmic Rays on the Soft Error Rate of DRAM at Ground Level. **IEEE Transactions on Electron Devices**, Notre Dame, v. 41, n. 4, 1994.

OGDEN, C. et al. The Impact of Soft Error Event Topography on the Reliability of Computer Memories. **IEEE Transactions on Reliability**, Hsinchu, v. 66, n. 4, p. 966-979, 2017.

OGUT, M. et al. Single event effect analysis for command and data handling electronics of a millimeter-wave radiometer 6u-class satellite instrument. In: **European Conference on Radiation and Its Effects on Components and Systems (RADECS)**, 17., 2017. Geneva. Anais … IEEE, 2017, p. 1-6.

OLAZABAL, A.; PLEITE, J. Multiple Cell Upsets inside aircrafts. New Fault-Tolerant Architecture. **IEEE Transactions on Aerospace and Electronic Systems**, Cranfield, v. 55, n. 1, p. 332-342, Feb. 2019.

QUIAO, L.; RIZOS, C.; DEMPSTER, A. G. Analysis and comparison of Cubesat lifetime. In: **Australian Space Conference**, 12., 2013, Melbourne. Anais … IEEE, 2013, p. 1-5.

RADAELLI, D.; PUCHNER, H.; WONG, S.; DANIEL, S. Investigation of multi-bit upsets in a 150 nm technology SRAM device. **IEEE Transactions on Nuclear Science**, Oak Ridge**,** v. 52, n. 6, p. 2433–2437, 2005.

REIVIREGO, P.; Liu, S.; Sánchez-Macián, A. et al. A scheme to reduce the number of parity check bits in Orthogonal Latin Square codes. **IEEE Transactions on Reliability**, Hsinchu, v. 66, n. 2, p. 518-528, 2017.

SAKATA, T.; HIROTSU, T.; YAMADA, H.; KATAOKA, T. A cost-effective dependable microcontroller architecture with instruction-level rollback for soft error recovery. In: **Proc. Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw**., 37., 2007, Edinburgh. Anais … IEEE, 2007, p. 1-2.

SATOH, S.; TOSAKA, Y.; WENDER, A. Geometric effect of multiple-bit soft errors induced by cosmic ray neutrons on DRAM's. **IEEE Electronic Device Letters**, v. 21, p. 310–312, Jun. 2000.

SILVA, F.; FREITAS, W.; SILVEIRA, J.; LIMA, O.; VARGAS, F.; MARCON, C. An efficient, low-cost ECC approach for critical-application memories. In: **Symposium on Integrated Circuits and Systems Design (SBCCI)**, 2017a. Fortaleza. Anais … IEEE, 2017, p. 198-203.

SILVA, F. et al. Evaluation of Multiple Bit Upset Tolerant Codes for NoC Buffering. In: **IEEE Latin American Symposium on Circuits & Systems (LASCAS)**, 8., 2017b. Bariloche. Anais … IEEE, 2017, p. 1-4.

SILVA, F.; FREITAS, W.; SILVEIRA, J.; MARCON, C.; VARGAS, F. Extended Matrix Region Selection Code: An ECC for adjacent multiple cell upset in memory arrays. **Microelectronics Reliability**, Maryland, v. 106, p. 1-9 Jan 2020.

SILVA, F.; PINHEIRO, A.; SILVEIRA, J.; MARCON, C. A Triple burst error correction based on region selection code. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, Charlottesville, v. 31, n. 8, p. 1214-1222, 2023.

SILVA, F.; SILVEIRA, J.; SILVEIRA, J.; MARCON, C.; VARGAS, F.; LIMA, O. An extensible code for correcting multiple cell upset in memory arrays. **Journal of Electronic Testing**, New York, v. 34, n. 4, p. 417-433, 2018.

SMITH, E. C. Effects of realistic satellite shielding on SEE rates. **IEEE Transactions on Nuclear Science**, Oak Ridge, v. 41, n. 6, p. 2396-2399, 1994.

TAMBATKAR, S.; NARAYANA, S.; SUDARSHAN, V.; et al. Error detection and correction in semiconductor memories using 3D parity check code with Hamming code. In: **International Conference on Communication and Signal Processing (ICCSP)**, 2017, Chennai. Anais … IEEE, 2017, p. 1-6.

VARGHESE, B.; SREELAL, S.; VINOD, P.; KRISHNAN, A. R. Multiple bit error correction for high data rate aerospace applications. In: **IEEE Conference on Information Communication Technologies (ICT)**, 2013, Thuckalay. Anais … IEEE, 2013, p. 1086–1090.

YUANYUAN, C.; XUNYING, Z. Research and implementation of interleaving grouping hamming code algorithm. In: **IEEE International Conference of Signal Processing, Communication and Computing**, 2013, Kunming. Anais … IEEE, 2013 p. 1–4.