IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 05, May 2020

Error Correction Codes Using Burst and Random Errors for Multiple-Cell Upsets in Space Application

Nimisha.N[1], P.Rajkumar[2], S.Rajkumar[3]

M.Tech Student, Department of ECE, NCERC, Pampady, Thrissur, Kerala, India[1]

Assistant Professor, Department of ECE, NCERC, Pampady, Thrissur, Kerala, India[2]

HOD, Department of ECE, NCERC, Pampady, Thrissur, Kerala, India[3]

Abstract: In space application, signals are transit from earth to space. Due to transition, radiation may occur. Thus, the probability of occurrence of Single-Cell Upsets (SCUs) or Multiple-Cell Upsets (MCUs) augments. One of the main causes of MCUs in space applications is high cosmic radiation. A common solution is the use of Error Correction Codes (ECCs). In this paper, compare the burst errors with random bit errors in ECCs and also a series of new low-redundant ECCs has been presented .These new ECCs improve the behaviour of the well-known able to correct MCUs with reduced area, power, and delay. Also, these new codes maintain, or even improve, memory error coverage with respect to Matrix and CLC codes. Currently, faults suffered by SRAM memory systems have increased due to the aggressive CMOS integration density. Nevertheless, when using ECCs in space applications, they must achieve a good balance between error coverage and redundancy, and their encoding/decoding circuits must be efficient in terms of area, power, and delay. Different codes have been proposed to tolerate MCUs. A common property of these codes is the high redundancy introduced. FUECs codes have been designed to MCUs in space applications which allow correction up to 4 -bit adjacent errors. FUEC-DAEC maintains coverage best, obtained results show that the proposed scheme is 11.438ns, 391nm², 345µw of delay, area, and power respectively. Beyond 4-bit burst errors, the performance of our codes decreases notably due to their low redundancy. Here introducing various random error bits to compare the burst error's redundancy. And also designed DMC codes (up to 4-bit) .D-DMC maintain the low area, power, and delay of 10.402ns, 538nm², 410µW respectively over than other DMC. Comparing these two methodologies, D-DMC has low delay than FUEC-DAEC but FUEC-DAEC has low area, and power than D-DMC with an improved capability of error correction and detection.

Keywords: Error correction code (ECCs), multiple-cell upsets (MCUs), FUECs, DMCs, redundancy.

I.INTRODUCTION

Presently, AS CMOS technology scales down to nanoscale and memories are combined with an increasing number of electronic systems, the soft error rate in memory cells is rapidly increasing with a great storage capacity. Nevertheless, this size decreasing has also caused an augment in the memory fault rate [1], [2], [3]. With the present aggressive scaling, the memory cell critical charge and the energy needed to provoke a single-event upset (SEU) in storage have been reduced [4]. As shown by different experiments, in addition to traditional single-cell upsets (SCUs), this energy reduction can provoke multiple-cell upsets (MCUs), that is, simultaneous errors in more than one memory cell induced by a single particle hit [5]-[9]. In the case of space applications, the MCU problem must be taken into account for the design of the corresponding fault tolerance methods, as space is an aggressive environment subjected to the impact of high-energy cosmic particles [5].

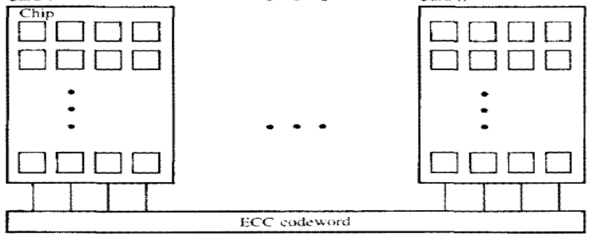


Fig. 1 A 4-bit-per-card memory array

Traditionally, Fig.1 shows error correction codes (ECCs) have been used to protect memory systems. Common ECCs employed to protect standard memories are single-error correction (SEC) or single-error-correction–double-error-detection (SEC–DED) codes. SEC codes are able to correct an error in one single memory cell. SEC–DED codes can correct an error in one single memory cell, as well as they can detect two errors in two independent cells. In critical applications, such as space applications, more complex and sophisticated codes are used. For instance, Matrix code is a well-known code that combines Hamming codes with parity check in a matrix, allowing the correction of two bits in error. Recently presented, column–line–code (CLC) [20] follows a similar approach, that is, it uses extended Hamming codes and parity bits to correct up to two adjacent bits in error. The main problem when memory systems employ an ECC is the redundancy required. The extra bits added are used to detect and/or correct the possible errors occurred. Also, redundant bits must be added for each data word stored in memory. In this way, the amount of storage occupied for redundant bit scales with the memory capacity. A series of ECCs that greatly reduces the redundancy introduced, while maintaining, or even improving, memory error coverage. In addition, area, power, and delay overheads are also reduced. These new codes have been designed using the flexible unequal error control (FUEC) methodology, developed by Saiz-Adalid *et al.* , where an algorithm (and a tool) to design FUEC codes is introduced. FUEC codes are an improvement of the well-known unequal error control (UEC) codes. Nevertheless, the FUEC methodology can also find other kinds of codes. It is employed to find low redundancy codes. These novel codes are different than those presented in. They only share the design methodology, but with different parameters. In this way, by using the tool, generate the parity check matrix of an ECC in an automatic and efficient way, just defining its error detection and/or correction capabilities. Novel random bit errors or decimal matrix code (DMC) based on divide with respect to H and V is proposed to provide enhanced memory reliability. The proposed DMC utilizes XOR and hamming bit codes to detect errors. Hamming code has been conceived to be applied at software level. It uses addition of integer values to detect and correct soft errors. The results obtained have shown that this approach have a lower delay overhead over other codes.

II. BASIC IDEA

A. Background on ECCs for Space Applications

Different ECCs has been traditionally applied to space missions. For instance, Berger code or the well known parity code has been used for detection purposes. When error correction is needed, more complex codes can be used, such as Hamming [13], Hadamard , Repetition , Golay , (BCH) , Reed–Solomon , Reed–Muller , multidimensional , or Matrix [18] codes. Hamming codes can be easily built for any word length. Also, the encoding and decoding circuits are easy to implement. Their main drawback is that only one bit in error can be corrected. Nevertheless, for common data word lengths (8, 16, 32, and 64), Hamming codes can detect some double error patterns, in addition to the SEC. Exploiting this feature, it is possible to systematize the detection of 2-bit adjacent errors with the same redundancy, as presented .
In these works, different ECCs based on hamming codes is introduced. These ECCs allow the correction of single bit errors or the detection of 2-bit adjacent errors with the same redundancy.
The main problem of Hadamard and Repetition codes is that they introduce a great redundancy for common data word lengths. This great redundancy provokes the necessity of a great memory storage capacity, which is an inconvenient for space applications. Golay code is able to correct up to 3-bit errors. Nevertheless, Golay code presents a redundancy of almost 100%. Also, this code presents a high time and power consuming ratio, as it has to execute sequentially two complementary sequences. Although BCH and Reed–Solomon codes can correct multiple errors, their main drawbacks are the great complexity and difficulty to implement them, as well as their great latency and speed. These weaknesses can be very problematic in space applications. Concerning Reed–Muller codes, although vastly used in critical applications, they present a great complexity. In this way, the overheads introduced are higher than the overheads introduced by Matrix or CLC codes. Multidimensional codes are a class of matrix codes that uses parity bits to detect and correct errors. With a low redundancy, these codes present several drawbacks. When more than two errors must be corrected, the code design is very complicated. Also, it is very difficult to adapt these codes to standard data word sizes.
A better alternative are the Matrix codes based on hamming codes [18], [20]. These codes still present a great redundancy, but they are more cost effective than previous multiple-error correcting codes. Design several new ECCs using the FUEC methodology and DMC methodology. The main characteristic of these new codes is their low redundancy. In order to check the behaviour of codes compared them with three types of codes. On the one hand, random error or MCU codes are created and compare three codes of them, as their codes present a good relationship between redundancy and area, power, and delay overheads. On the other hand, with Hamming-based codes due to the very low redundancy of these codes are also calculated. Also compare FUEC and MCU errors.

B. Basics on Coding Theory

An (n, k) binary ECC encodes a k-bit input word in an n-bit output word. The input word u = $(u0, u1, . . . , uk-1)$ is a k-bit vector which represents the original data. The code word $\mathbf{b} = (b0, b1, . . . , bn-1)$ is a vector of n bits, where the

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 05, May 2020

($n-k$) redundant bits added are called parity or code bits. **b** is transmitted across an unreliable channel which delivers the received word **r** = ($r0$, $r1$, . . . , $rn-1$). The error vector **e** = ($e0$, $e1$, . . . , $en-1$) models the error induced by the channel. If no error has occurred in the i th bit, i = 0; otherwise i = 1. In this way, **r** can be interpreted as **r** = **b** ⊕ **e**.

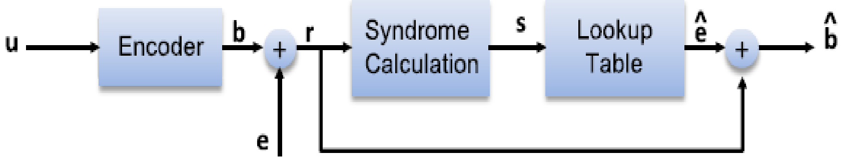


Fig. 2 Encoding, channel crossing, and decoding process

The parity-check matrix **H** $(\mathbf{n}-\mathbf{k})\times n$ of a linear block code defines the code . For the encoding process, **b** must accomplish the requirement $\mathbf{H} \cdot \mathbf{b}^T = \mathbf{0}$. For syndrome decoding, the syndrome is defined as $\mathbf{s}^T = \mathbf{H} \cdot \mathbf{r}^T$, and it exclusively depends one $\mathbf{s}^T = \mathbf{H} \cdot \mathbf{r}^T = \mathbf{H} \cdot (\mathbf{b} \oplus \mathbf{e})^T = \mathbf{H} \cdot \mathbf{b}^T \oplus \mathbf{H} \cdot \mathbf{e}^T = \mathbf{H} \cdot \mathbf{e}^T$.
There must be a different **s** for each correctable **e**. If **s** = **0**, we can assume that **e** = **0**. Therefore, **r** is correct. Otherwise, an error has occurred. Syndrome decoding is performed by addressing a lookup table that relates each **s** with the decoded error vector ˆ**e**. The decoded code word ˆ**b** is calculated as ˆ**b** = **r** ⊕ ˆ**e**. From ˆ**b**, it is easy to obtain ˆ**u** just discarding the parity bits. If the fault hypothesis employed to design the ECC is consistent with the channel behaviour, ˆ**u** and **u** must be equal with a very high probability.

C. Error Models

In coding theory [12], the term random error commonly refers to one or more bits in error, distributed randomly in the encoded word (data bits plus code bits generated by the ECC). Random errors can be single (only one bit affected) or multiple. Single errors are the simplest ones, as they only affect a single memory cell. They are commonly produced by SEUs (in random access memories) or single-event transients (in combinational logic).
Multiple errors mainly manifest as bursts. We can define a burst error as a multiple error that spans l bits in a word, i.e., a group of contiguous bits where, at least, the first and the last bits are in error. The separation l is known as burst length. Notice that adjacent errors are a particular type of burst errors where all the erroneous bits are contiguous. The main physical causes of a burst error in the context of RAM memories are diverse: high energy cosmic particles that hit some neighbour cells, crosstalk between adjacent cells, etc [2].

D. Hamming Codes

Hamming codes are able to correct single-bit errors with the lowest redundancy. For example, the parity-check matrix for the Hamming (7, 4), i.e., n = 7 and k = 4, is shown in the following equation:

$$H = \begin{bmatrix} 1010101 \\ 0110011 \\ 0001111 \end{bmatrix}$$

It is easy to deduce the encoding and decoding operations.
The encoding formulas are shown in Table I. In the same way, it is also possible to obtain the syndrome decoding formulas from parity check matrix.

TABLE I ENCODING FORMULAS FOR THE HAMMING (7, 4) CODE

b_0	b_1	b_2	b_3	b_4	b_5	b_6	**Encoding formulas**
		u_0		u_1	u_2	u_3	
1	0	1	0	1	0	1	$b_0=u_0+u_1+u_3$
0	1	1	0	0	1	1	$b_1=u_0+u_2+u_3$
0	0	0	1	1	1	1	$b_2=u_1+u_1+u_3$

Table II shows the expressions obtained to calculate the syndrome bits for the Hamming (7, 4) code.

TABLE II SYNDROME BITS FOR THE HAMMING (7, 4) CODE

r_0	**r_1**	**r_2**	**r_3**	**r_4**	**r_5**	**r_6**	**Syndrome bits**
		u_0		u_1	u_2	u_3	
1	0	1	0	1	0	1	$s_0=r_0+r_1+r_3$
0	1	1	0	0	1	1	$s_1=r_0+r_2+r_3$
0	0	0	1	1	1	1	$s_2=r_1+r_1+r_3$

If an error occurs, the syndrome bits will locate the erroneous bit. A lookup table (implemented, for example, using a binary decoder) selects the erroneous bit. Applying the "exclusive-or" operation, the output of the lookup table correct the erroneous bit. For common word lengths, such as 8, 16, 32, and 64 bits, there exist (12, 8), (21, 16), (38, 32), and (71, 64) Hamming codes, respectively. As it can be seen, redundancy decreases with longer data words. For instance, the (12, 8) Hamming code presents a 50% of redundancy, whereas the (71, 64) Hamming code introduces about 11% of redundancy.

Hamming codes can be extended to correct single errors and detect double random errors. These codes are known as SEC–DED extended Hamming codes [13]. These codes need an additional parity bit to achieve the double-error detection. It is calculated as the even parity for the whole encoded word. In this way, and just adding an extra bit *b*7, the Hamming SEC code (7, 4) shown previously converts into an extended Hamming SEC–DED code (8, 4). *b*7 can be obtained as follows:

$$b7 = b0 \oplus b1 \oplus b2 \oplus b3 \oplus b4 \oplus b5 \oplus b6.$$

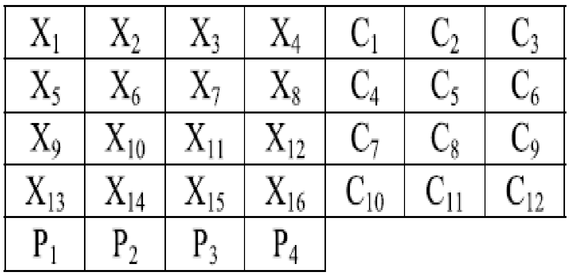
The decoding process checks two conditions:
1) The parity of the whole received word.
2) The syndrome bits, which are calculated as in the Hamming code.

There exist also (13, 8), (22, 16), (39, 32), and (72, 64) SEC–DED extended Hamming codes. As in the SEC codes, redundancy decreases with higher data word lengths.

I. METHODOLOGY AND PROCEDURE

A. Existing System

As previously, Matrix and CLC codes have been designed to tolerate MCUs [18], [20] a critical concern in space applications. Combining Hamming codes and parity checks, Matrix codes form a 2-D scheme for correcting and detecting some patterns of MCUs. In Fig. 3, where *Xi* are the data bits, *Ci* are the horizontal check bits (calculated as a Hamming code), and *Pi* are the column parity bits (even parity).



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			

Fig. 3 Layout of a 16-bit data word for the (32, 16) Matrix code

The basic behaviour of this Matrix code is as follows. The primary data input (*Xi)* is divided into groups of several bits. This division is in groups of 4 bits. Each group is codified by a (7, 4) Hamming code (*Ci)*. Last, a set of vertical parity bits (*Pi)* completes the matrix.

X_1	X_2	X_3	X_4	C_1	C_2	C_3	Pa_1
X_5	X_6	X_7	X_8	C_4	C_5	C_6	Pa_2
X_9	X_{10}	X_{11}	X_{12}	C_7	C_8	C_9	Pa_3
X_{13}	X_{14}	X_{15}	X_{16}	C_{10}	C_{11}	C_{12}	Pa_4
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8

Fig. 4 Layout of a 16-bit data word for the (40, 16) CLC

The Matrix code implemented better correction and detection performance than an extended Hamming code, as it is able to correct all single errors and to correct or to detect all 2-bit burst errors. Nevertheless, this Matrix code presents a higher redundancy than an extended Hamming code. In this way, memory required for code bits is increased, and also, area, energy, and delay overheads.
Recently presented, CLC code is another matrix code proposed to be used in space applications. The layout is employed for the implementation of this code is shown in Figure.4, where *Xi* is the primary data input, divided into groups of 4 bits. Each group is codified by an SEC–DED (8, 4) extended Hamming code (*Ci* and *Pai)*. Finally, a set of vertical parity bits (*Pi)* form the matrix. As just commented, and unlike the Matrix code, CLC uses an Extended Hamming code, allowing the correction of all single and 2-bit burst errors. Nevertheless, CLC introduces a higher number of redundant bits, provoking a greater area, power, and delay overheads, and reducing the available memory for the payload.On the other hand, Saiz-Adalid et al. Introduce an SEC–double-adjacent error detection (SEC–DAED) code with a very low redundancy. This code is able to correct all single errors or to detect all double adjacent errors in a 16-bit data word with only five redundant bits. Fig. 5, shows the data word layout of this code, where *Ci* are the code bits, and *Xi* are the data bits.

C_1	C_2	C_3	C_4	C_5	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}

Fig. 5 Layout of a 16-bit data word for the (21, 16) SEC–DAED

On the contrary, Sanchez-Macian et al. present a different approach to generate Hamming-based ECCs. In this case, select an SEC–DAED code with the same coverage characteristics, as well as the same redundancy of the SEC–DAED code. The main difference is the code layout, as shown in Fig. 6.

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	C_1	X_9	C_2	C_3	C_4	X_{10}	C_5	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}

Fig. 6 Layout of a 16-bit data word for the (21, 16) SEC–DAED

To obtain the value of the different *Ci* bits, the parity-check matrix of these two SEC–DAED codes respectively.

B. Proposed System
The proposed method focuses on identifying of burst errors and random errors. The main indications are adjacent errors of 2-to 4-bit length and random errors of 2- to 4-bit length of burst and random errors respectively.

Flexible Unequal Error Control: This methodology was developed to obtain FUEC codes. However, it can be generalized to find any kind of codes. This methodology is based on formulating the problem as a Boolean Satisfiability problem. An algorithm developed by the authors is employed to solve it and to obtain a parity-check matrix, which defines the code to be designed. After defining the values of *n* and *k* for the code, the first step is the selection of error patterns to be corrected and detected. For instance, single errors are represented with error vectors (. . . 1 . . .), and error vectors for double random errors show the pattern (. . . 1 . . . 1 . . .), where 1's represent the bits in error, and the dots represent the correct bits. The next step is to find the parity-check matrix **H**, where ***E+*** represents the set of error vectors to be corrected, and ***E*Δ** is the set of error vectors to be detected. That is, each correctable error must generate a different syndrome **H** ·

$$\mathbf{e}^{T}i \neq \mathbf{H} \cdot \mathbf{e}^{T}j \ ; \forall\forall \mathbf{e}i \ , \mathbf{e}\, j \ \mathcal{E}\ E{+}|\mathbf{e}i \ \_{=}\ \mathbf{e}j \ .$$

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 05, May 2020

In addition, each detectable error must generate a syndrome which is different to all the syndromes generated by the correctable errors

$$\mathbf{H} \cdot \mathbf{e}^{T}i \neq \mathbf{H} \cdot \mathbf{e}^{\mathbf{T}}j \text{ ; } \forall \mathbf{e}i \; \mathcal{E} \; E\Delta, \; \mathbf{e}\, j \; \mathcal{E} \; E+.$$

However, several detectable errors may have the same syndrome. To find the matrix, a recursive backtracking algorithm is used. It checks partial matrices and adds a new column only if the previous matrix satisfies the requirements. In this way, the algorithm starts with an empty partial_H matrix. New columns, with n–k rows, are added, and the new partial matrices are checked recursively. The added columns must be nonzero, so there are $2n-k$–1 combinations for each column. The complete execution of the algorithm is commonly unfeasible. Nevertheless, the first solutions are usually found quickly, if the code exists. Once selected the **H** matrix, it is easy to determine the logic equations to calculate each parity and syndrome bit, as well as the syndrome lookup table. They are required for the encoder and decoder implementation. In addition, we can apply two different optimization criteria. If we want to decrease the delay of the encoders and decoders, we have to reduce the number of 1's in those rows with the highest number of 1's of the parity-check matrix. In the case of area reduction, the total number of 1's in the parity-check matrix must be reduced.

FUEC Methodology: By using the FUEC methodology [1], have been able to design several codes that improve the coverage and/or the redundancy of the different codes presented previously (Matrix, CLC, and both SEC–DAED codes). The layout of our codes is presented in Figure.7, where *Ci* are the code bits and *Xi* are the data bits. Using our algorithm, have found a code [call it FUEC–double adjacent error correction (DAEC)] that can correct an error in a single bit, or an error in two adjacent bits, or it can detect one 3-bit burst error or one 4-bit burst error.

C_0	.. ..	C_{n-1}	X_0	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}

Fig. 7 Layout of a 16-bit data word for the FUEC codes

FUEC–DAEC needs only seven code bits for a 16-bit data word. Figure.8, shows the parity-check matrix **H** for this code, where *Ci* are the code bits and *Xi* are the primary data bits.

$$\mathbf{H} = \begin{array}{c} C_0 \cdots\cdots\cdots\cdots C_6\, X_0 X_1 \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots X_{15} \\ \left| \begin{array}{ccccccccccccccccccccccc} 1&0&0&0&0&0&0&1&0&0&0&1&0&0&1&1&0&0&1&1&1&0&0 \\ 0&1&0&0&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&0&0&1&0 \\ 0&0&1&0&0&0&0&1&0&1&0&0&0&1&1&0&1&0&0&1&0&1&1 \\ 0&0&0&1&0&0&0&0&1&0&0&1&0&0&0&1&1&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0&1&0&0&1&1&0&0&1&0&0&0&0&1&1&0&1 \\ 0&0&0&0&0&1&0&0&1&1&0&0&1&0&0&0&1&1&0&1&0&1&0 \\ 0&0&0&0&0&0&1&0&0&1&1&0&0&1&0&1&0&1&1&0&1&0&1 \end{array} \right| \end{array}$$

Fig. 8 Parity-check matrix **H** for the (23, 16) FUEC–DAEC code

Once **H** has been obtained, it is very easy to design the encoder/decoder circuitry. For example, the formulas to calculate the code bits for the FUEC–DAEC code are

$C_0 = X_0 \oplus X_4 \oplus X_7 \oplus X_8 \oplus X_{11} \oplus X_{12} \oplus X_{13}$
$C_1 = X_1 \oplus X_3 \oplus X_5 \oplus X_7 \oplus X_9 \oplus X_{10} \oplus X_{11} \oplus X_{14}$
$C_2 = X_0 \oplus X_2 \oplus X_6 \oplus X_7 \oplus X_9 \oplus X_{12} \oplus X_{14} \oplus X_{15}$
$C_3 = X_1 \oplus X_4 \oplus X_8 \oplus X_9 \oplus X_{12}$
$C_4 = X_0 \oplus X_3 \oplus X_4 \oplus X_7 \oplus X_{12} \oplus X_{13} \oplus X_{15}$
$C_5 = X_1 \oplus X_2 \oplus X_5 \oplus X_9 \oplus X_{10} \oplus X_{12} \oplus X_{14}$
$C_6 = X_2 \oplus X_3 \oplus X_6 \oplus X_8 \oplus X_{10} \oplus X_{11} \oplus X_{13} \oplus X_{15}$

On the other hand, the syndrome formulas that can be obtained from **H** to check if an error has occurred are
$S_0 = C_0 \oplus X_0 \oplus X_4 \oplus X_7 \oplus X_8 \oplus X_{11} \oplus X_{12} \oplus X_{13}$
$S_1 = C_1 \oplus X_1 \oplus X_3 \oplus X_5 \oplus X_7 \oplus X_9 \oplus X_{10} \oplus X_{11} \oplus X_{14}$
$S_2 = C_2 \oplus X_0 \oplus X_2 \oplus X_6 \oplus X_7 \oplus X_9 \oplus X_{12} \oplus X_{14} \oplus X_{15}$
$S_3 = C_3 \oplus X_1 \oplus X_4 \oplus X_8 \oplus X_9 \oplus X_{12}$
$S_4 = C_4 \oplus X_0 \oplus X_3 \oplus X_4 \oplus X_7 \oplus X_{12} \oplus X_{13} \oplus X_{15}$
$S_5 = C_5 \oplus X_1 \oplus X_2 \oplus X_5 \oplus X_9 \oplus X_{10} \oplus X_{12} \oplus X_{14}$
$S_6 = C_6 \oplus X_2 \oplus X_3 \oplus X_6 \oplus X_8 \oplus X_{10} \oplus X_{11} \oplus X_{13} \oplus X_{15}$

Our second proposal, called FUEC–triple adjacent error correction (TAEC), is able to correct an error in a single bit, or an error in two adjacent bits (2-bit burst errors) or a 3-bit burst error, or it can detect a 4-bit burst error. This is possible by adding one more code bit. In this case, for a 16-bit data word, the FUEC–TAEC code needs eight code bits. The parity-check matrix **H** for this code is presented in Fig. 9.As in the case of the FUEC–DAEC, *Ci* are the code bits and *Xi* are the primary data bits. Similarly, from **H** it is very easy to design the encoder/decoder circuitry.

$$\mathbf{H} = \begin{array}{c} C_0 \cdots\cdots C_7 X_0 X_1 \cdots\cdots X_{15} \\ \left|\begin{array}{cccccccccccccccccccccccc}
1&0&0&0&0&0&0&0&1&0&0&1&1&0&0&1&1&0&0&1&0&0&0&1\\
0&1&0&0&0&0&0&0&0&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&1&0&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&1\\
0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&1&0&0&0&1&1&1\\
0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&1&1&1&1&0\\
0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&1&1&1&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&1&0&0&0&1&0&1&0
\end{array}\right| \end{array}$$

Fig. 9 Parity-check matrix **H** for the (24, 16) FUEC–TAEC code

Finally, FUEC–quadruple adjacent error correction (QUAEC) is the last code design. This code is able to correct an error in a single bit, or an error in two adjacent bits (2-bit burst errors) or a 3-bit burst error or a 4-bit burst error. This can be done by using only nine code bits. The parity-check matrix **H** for this code is shown in Fig. 10. As in the previous codes, *Ci* are the code bits and *Xi* are the primary data bits.

$$\mathbf{H} = \begin{array}{c} C_0 \cdots\cdots C_8 X_0 X_1 \cdots\cdots X_{15} \\ \left|\begin{array}{ccccccccccccccccccccccccc}
1&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&0&0&1&0&0&0&1\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&1\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&1&0&0&0&1\\
0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&1&0&0
\end{array}\right| \end{array}$$

Fig. 10 Parity-check matrix **H** for the (25, 16) FUEC–QUAEC code

A parity-check matrix optimized to achieve the lowest delay for our three codes, that is, with a reduced number of 1's in the rows with the highest number of 1's.
It is also remarkable that the names FUEC-_AEC only indicate that they have been designed using the FUEC methodology. The codes presented here must not been confused with the FUEC codes from, an improvement of UEC codes.

Decimal Matrix Code: The proposed schematic of memory is depicted in Fig. 11. First, during the encoding (write) process, information bits *D* are fed to the DMC encoder, and then the horizontal redundant bits *H* and vertical redundant bits *V* are obtained from the DMC encoder. When the encoding process is completed, the obtained DMC codeword is stored in the memory. If MCUs occur in the memory, these errors can be corrected in the decoding (read) process.

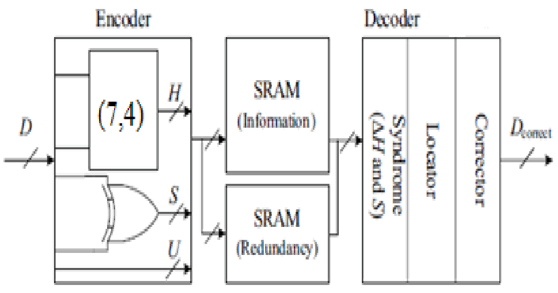


Fig .11 Proposed schematic for DMC

In the divide-symbol and arrange-matrix ideas are performed, i.e., the *N*-bit word is divided into *k* symbols of *m* bits ($N = k \times m$), and these symbols are arranged in a $k_1 \times k_2$ 2-D matrix ($k = k_1 \times k_2$, where the values of k_1 and k_2 represent the

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

Vol. 8, Issue 05, May 2020

numbers of rows and columns in the logical matrix respectively). Second, the horizontal redundant bits *H* are produced by performing hamming codes and XORs gates. Third, the vertical redundant bits *V* are obtained by binary operation among the bits per column. It should be noted that both divide-symbol and arrange-matrix are implemented in logical instead of in physical. Therefore, the proposed DMC does not require changing the physical structure of the memory. To explain the proposed DMC scheme, take a 16-bit word as an example, as shown in Fig. 12. The cells from D_0 to D_{16} are information bits. H_0–H_{11} are horizontal check bits; V_0 through V_7 are vertical check bits.

D_7	**D_6**	**D_5**	**D_4**	**D_3**	**D_2**	**D_1**	**D_0**	**H_5**	**H_4**	**H_3**	**H_2**	**H_1**	**H_0**
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	H_{11}	H_{10}	H_9	H_8	H_7	H_6
V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7						

Fig. 12 16-bits DMC logical organization

The horizontal redundant bits '***H***' can be obtained by applying hamming (7, 4) code in TABLE-I as follows:

$$H_0 = D_0 \oplus D_1 \oplus D_3$$
$$H_1 = D_0 \oplus D_2 \oplus D_3$$
$$H_2 = D_1 \oplus D_2 \oplus D_3$$
$$H_3 = D_4 \oplus D_5 \oplus D_7$$
$$H_4 = D_4 \oplus D_6 \oplus D_7$$
$$H_5 = D_5 \oplus D_6 \oplus D_7$$
$$H_6 = D_8 \oplus D_9 \oplus D_{11}$$
$$H_7 = D_8 \oplus D_{10} \oplus D_{11}$$
$$H_8 = D_9 \oplus D_{10} \oplus D_{11}$$
$$H_9 = D_{12} \oplus D_{13} \oplus D_{15}$$
$$H_{10} = D_{12} \oplus D_{14} \oplus D_{15}$$
$$H_{11} = D_{13} \oplus D_{14} \oplus D_{15}$$

When mode signal and enable signal is high, the H and V redundant bits are calculated. Mode signal become 1 then only operation take place. Horizontal redundant bits are occupied as $H_2H_1H_0$, $H_5H_4H_3$, $H_8H_7H_6$, and $H_{11}H_{10}H_9$. Random errors detect the MCU bits and correct it.

For the vertical redundant bits '***V***', we have

$$V_0 = D_0 \oplus D_8$$
$$V_1 = D_1 \oplus D_9$$
$$V_2 = D_2 \oplus D_{10}$$
$$V_3 = D_3 \oplus D_{11}$$
$$V_4 = D_4 \oplus D_{12}$$
$$V_5 = D_5 \oplus D_{13}$$
$$V_6 = D_6 \oplus D_{14}$$
$$V_7 = D_7 \oplus D_{15}$$

The encoding can be performed by decimal and binary hamming operation codes. The encoder that computes the redundant bits using multibit of hamming codes and XOR gates is shown in Fig. 13. In this Fig. 13, $H_{11} - H_0$ are horizontal redundant bits, $V_7 - V_0$ are vertical redundant bits, and the remaining bits $U_{15} - U_0$ are the information bits which are directly copied from D_{15} to D_0.

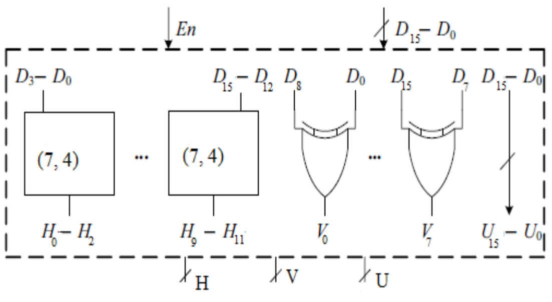
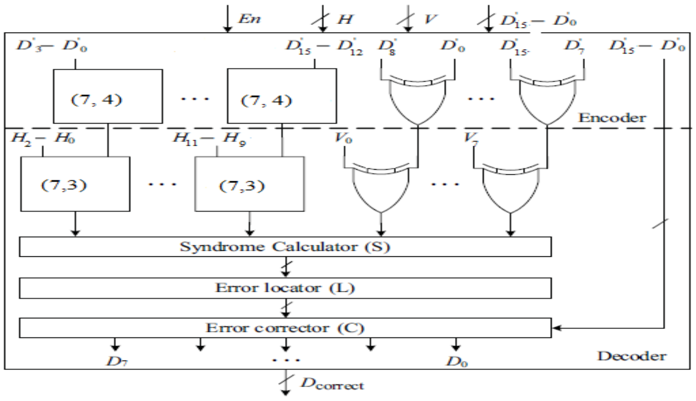


Fig. 13 16-bit DMC encoder structure using hamming codes and XOR gates

Vol. 8, Issue 05, May 2020

The proposed DMC decoder is depicted in Fig.14, which is made up of the following submodules, and each executes a specific task in the decoding process: syndrome calculator, error locator, and error corrector.
It can be observed from this Fig. 14, that the redundant bits must be recomputed from the received information bits *D′*. From Fig.14, it can be observed that the DMC encoder is also reused for obtaining the syndrome bits in DMC decoder.



Extra circuit	*En* signal		Function
	Read signal	Write signal	
Encoder	0	1	Encoding
	1	0	Compute syndrome bits

Fig .14 16-bit DMC decoder structure

Besides, this figure also shows the proposed decoder with an enable signal *En* for deciding whether the encoder needs to be a part of the decoder. In other words, the *En* signal is used for distinguishing the encoder from the decoder, and it is under the control of the write and read signals in memory. Therefore, in the encoding (write) process, the DMC encoder is only an encoder to execute the encoding operations. However, in the decoding (read) process, this encoder is employed for computing the syndrome bits in the decoder. These clearly show how the area overhead of extra circuits can be substantially reduced.

III. RESULTS AND DISCUSSION

ECCs in space applications, burst errors must achieve a good balance between error coverage and redundancy, and their circuits must be efficient in terms of area, power, and delay. And also proposed DMC has been implemented in VHDL, simulated with ModelSim and tested for functionality by given various inputs. The area, power, and critical path delay of circuits have been obtained. ECCs in space applications, random errors or DMC must achieve a good balance between redundant bits, and their circuits must be efficient in terms of area, power, and delay.

A. Error Coverage Evaluation

In order to study the error coverage of the different ECCs developed a simulator that allows injecting different types of error. The basic scheme is shown in Fig. 15.

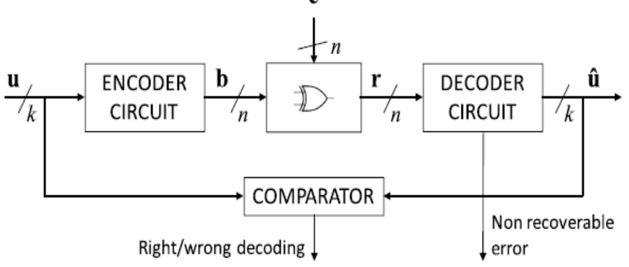


Fig. 15 Block diagram of the fault injector simulator

This tool allows the injection of different error types. By comparing the input and output words, the simulator can check if the error injected leads to a right or wrong decoding. Also, the decoder circuit can activate the NRE signal when an error is detected but it cannot be corrected. Block diagram of the fault injector simulator. process for all errors

Vol. 8, Issue 05, May 2020

of a given size and model (i.e., random or burst), it is possible to count the number of corrected and/or detected errors with respect to the total number of possible errors, that is, it is possible to calculate the coverage of each ECC.
This coverage has been calculated as

$$Ccorrec = \frac{\text{Errors\_Corrected}}{Errors-Injected} \times 100$$

Where Errors_Corrected are the number of errors corrected by the ECC, and Errors_Injected are the total amount of errors injected for a given burst length.

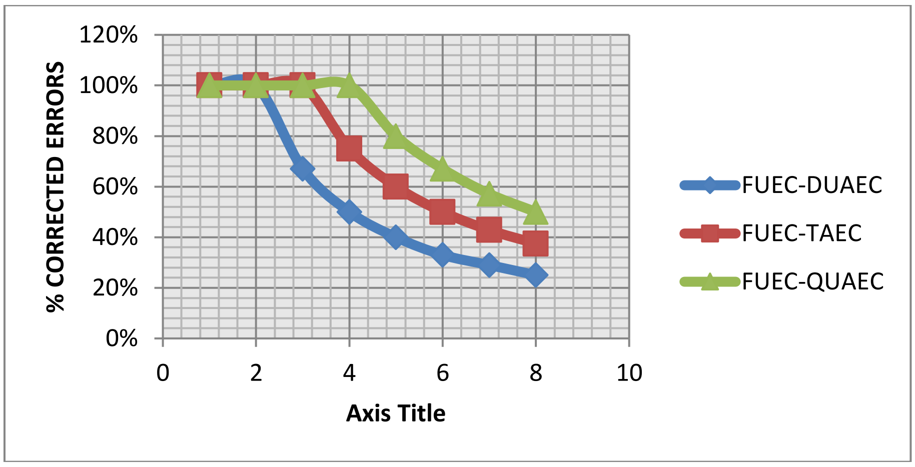


Fig. 16 Correction coverage for burst errors

As expected, FUEC–TAEC and FUEC–QUAEC codes present better correction coverage than Matrix, CLC, and both SEC–DAED codes, as they are able to correct up to 3- and 4-bit burst errors respectively. On the other hand, our FUEC–DAEC code can correct up to 2-bit burst errors.Nevertheless, for longer burst errors (5 bits or more), the correction capabilities of codes degrade more deeply than Matrix and CLC ones. This is provoked by the lower redundancy.
This detection coverage is calculated as

$$Cdetec = \frac{\text{Errors\_Corrected + Errors\_Detected}}{Errors-Injected} \times 100$$

Where Errors_Detected corresponds to the number of errors not corrected but detected by the ECCs. All codes present a 100% detection of up to 4-bit burst errors, improving the behaviour of Matrix, CLC, and both SEC–DAED codes. As in the correction coverage, the percentage of detected errors in our FUEC–TAEC and FUEC–QUAEC codes degrades sharply for longer bursts. By contrast, FUEC–DAEC maintains coverage over 60%, near the Matrix detection capability.

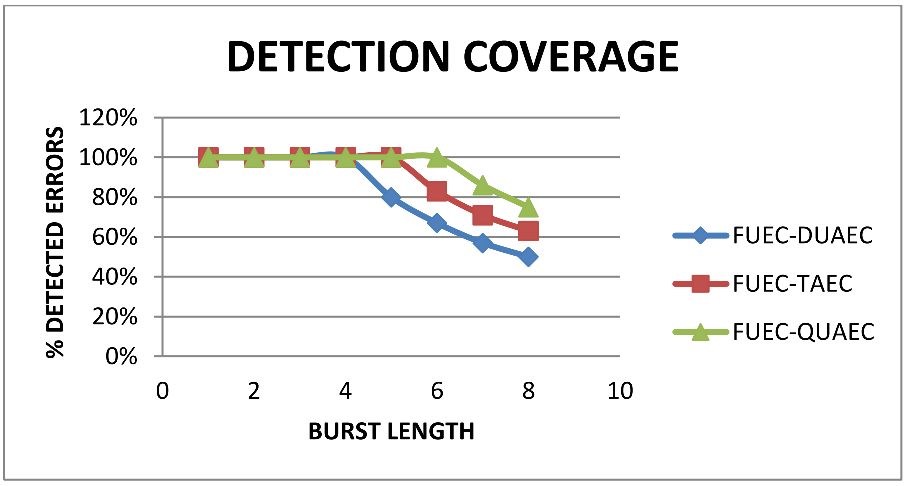


Fig. 17 Detection coverage for burst errors

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 05, May 2020

In conclusion, codes are very efficient to tolerate burst errors of from 2- to 4-bit length. Beyond 4-bit burst errors, the performance of our codes decreases notably due to their low redundancy. If these errors are expected to occur, more powerful codes must be employed.

B. Redundancy

With respect to the code bits needed, our three codes present a very low redundancy with respect to the Matrix and CLC codes. Nevertheless, the lowest redundancy corresponds to both SEC–DAED codes, but this low redundancy only permits the correction of single errors, as can be seen from Table IV.

TABLE III NUMBER OF CODE BITS FOR A 16-BIT DATA WORD

CODE	NO.Code Bits	%Redundancy	Burst error detection & correction capabilities
Matrix	16	100%	CORRECTION: Single bit errors DETEDTION: 2-bit burst errors
CLC	24	150%	CORRECTION: Single&2- bit errors DETEDTION: 2-bit burst errors
SEC-DAED[27]	5	31.25%	CORRECTION: Single bit errors DETEDTION: 2-bit burst errors
SEC-DAED[28]	5	31.25%	CORRECTION: Single bit errors DETEDTION: 2-bit burst errors
FUEC-DAEC	7	43.75%	CORRECTION: Single ,2-bit errors DETEDTION: 3-&4--bit burst errors
FUEC-TAEC	8	50%	CORRECTION: Single,2-&3- bit errors DETEDTION: 4-bit burst errors
FUEC-QUAEC	9	56.25%	CORRECTION: Single ,2-,3-&4-bit errors DETEDTION: 4-bit burst errors

Table IV shows the number of code bits introduced by each ECC, as well as the redundancy introduced with respect to a 16-bit data word. FUEC methodology and algorithm can be applied to longer code word sizes. Calculus of the redundancy has been done with

$$\text{Redundancy} = \frac{\text{No.code bits}}{No.data\ bits} \times 100.$$

The importance of a low redundancy comes from the fact that these extra bits must be also stored in memory in order to check if an error has occurred. In this way, a higher redundancy means a lower storage available for data bits.

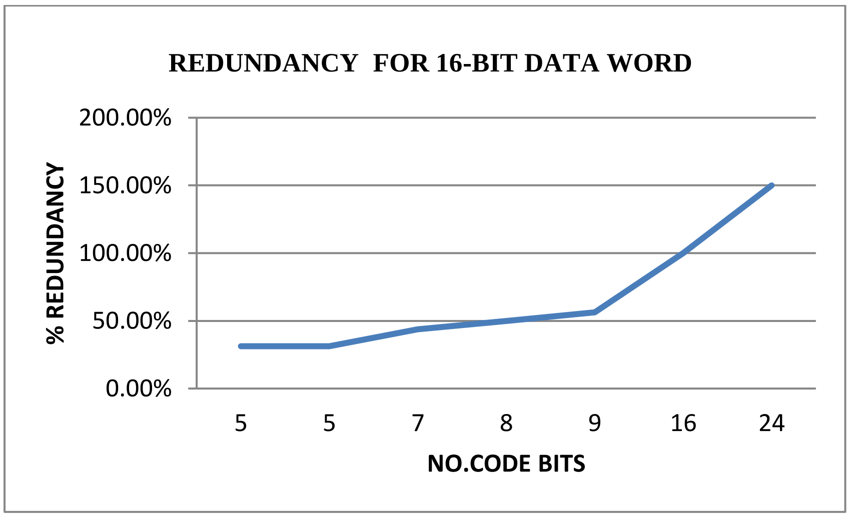


Fig .18 Graphical representation of redundancy

IJIREEICE

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

Vol. 8, Issue 05, May 2020

As an example, if we use a 1-GB memory chip, only 512 MB are available to store data bits in the case of the Matrix code, requiring the remaining 512 MB to store the code bits. In the case of the CLC code, only about 410 MB are available to store data bits. On the other hand, in the case of the SEC–DAED codes, the available memory for data is about 780 MB. Following with this example, our FUEC–DAEC code allows storing about 712 MB of primary data. In contrast, our FUEC–TAEC and FUEC–QUAEC codes allow storing 682 and 655 MB, respectively. As it can be seen, the increment in the storage available is very significant, taking into account the improvement in the coverage properties of our three codes. Fig. 20 shows the graphical representation of redundancy. Last column of Table IV shows the burst error coverage of the different ECCs, a concern to have into account in space applications. MCUs mainly provoke 2-bit adjacent errors in earth observation satellites; although a non negligible percentage of longer burst errors are also presented. In deep space exploration, a higher impact of longer MCUs is expected.

C. Redundant Bits

Table V, shows the reduction in number of redundant bits for the proposed 16-bit DMC and the original 16-bit DMC. Coding efficiency β is used to evaluate the area overhead of memory cell

$$\beta = \frac{\text{redundant bits}}{\text{redundant bits} + \text{information bits}} \times 100$$

If the value of β is small, the code needs lower memory cell.

TABLE IV COMPARISON OF REDUNDANT BITS

ECC	Redundant bits		Total no. of redundant bits	Information bits	% β
	H	V			
DMC(ORIGINAL)	10	8	18	16	52.94
PROPOSED DMC	12	8	20	16	55.56

From this table IV, proposed DMC code using hamming code and XOR gates have less coding efficiency.DMC using hamming code has the least β value but its correction capability is aConstant (up to 4-bits).

D. Synthesis Results

It has three codes, FUEC-DAEC, FUEC-TAEC, and FUEC-QUAEC. Implement this code, in order to obtain area, power, and delay. Area occupied by the different circuits in nm². The power consumption of the different FUECs circuit is observed in µW. Delay introduced by the different codes in FUECs or burst errors in ns.

1. Below figure 19, shows the RTL diagram of FUEC-DAEC from Xilinx software for burst error.

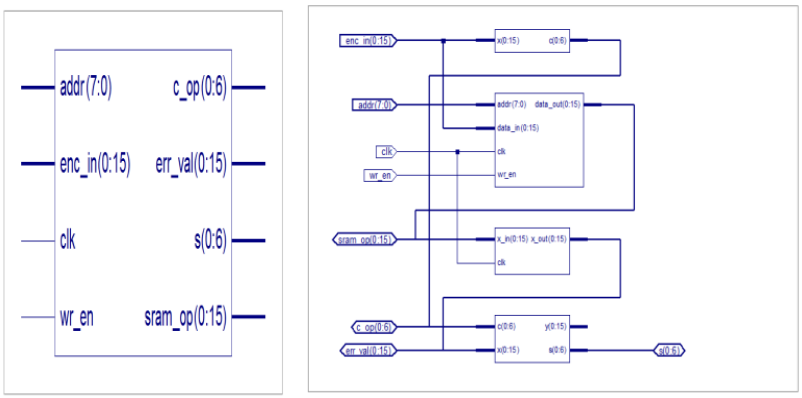
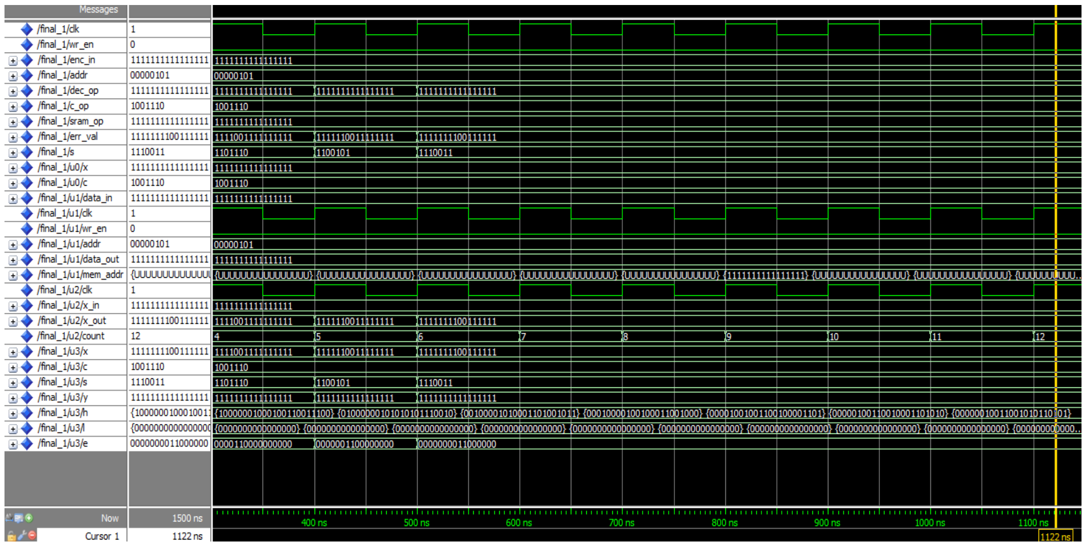


Fig. 19 RTL diagram of FUEC-DAEC.

ISSN (Online) 2321-2004
ISSN (Print) 2321-5526

Vol. 8, Issue 05, May 2020

In digital circuit design, register-transfer level is a design abstraction .Fig. 19 show the bit size of the input data , code data and clk .
2. The design of FUEC-DAEC is implemented in VHDL and simulated with ModelSim.

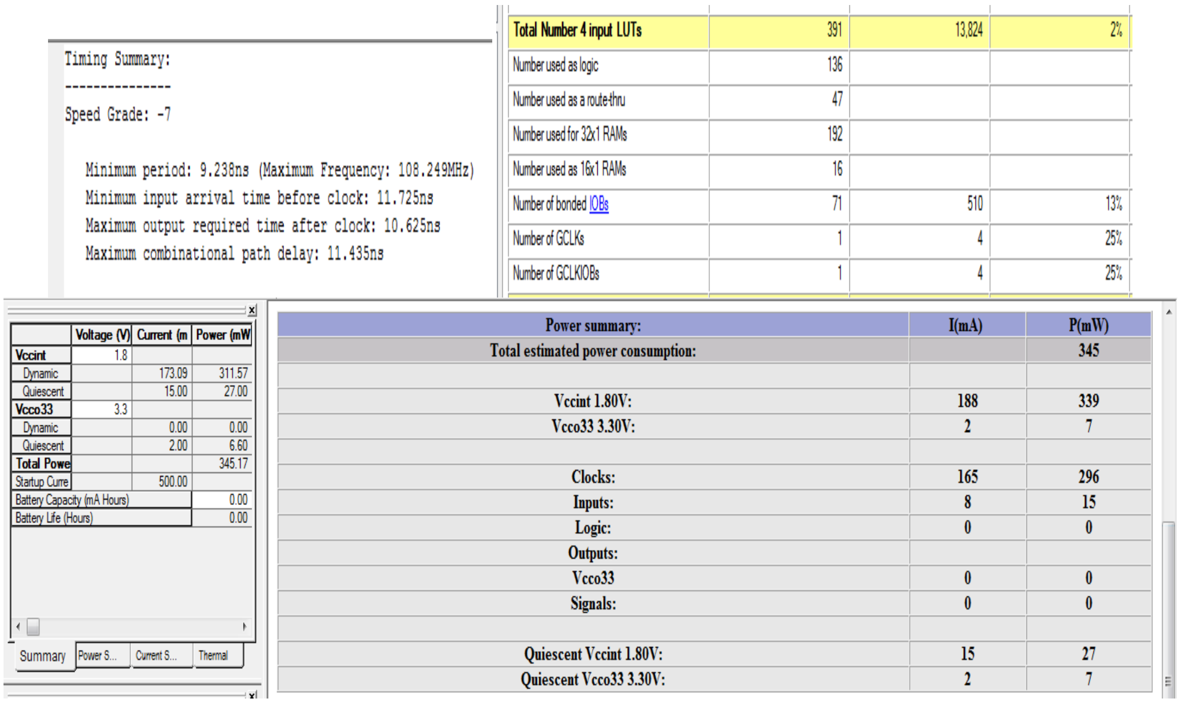


Messages	
/final_1/clk	1
/final_1/wr_en	0
/final_1/enc_in	1111111111111111
/final_1/addr	00000101
/final_1/dec_op	1111111111111111
/final_1/c_op	1001110
/final_1/sram_op	1111111111111111
/final_1/err_val	1111111100111111
/final_1/s	1110011
/final_1/u0/x	1111111111111111
/final_1/u0/c	1001110
/final_1/u1/data_in	1111111111111111
/final_1/u1/clk	1
/final_1/u1/wr_en	0
/final_1/u1/addr	00000101
/final_1/u1/data_out	1111111111111111
/final_1/u1/mem_addr	{UUUUUUUUUUUUUU
/final_1/u2/clk	1
/final_1/u2/x_in	1111111111111111
/final_1/u2/x_out	1111111100111111
/final_1/u2/count	12
/final_1/u3/x	1111111100111111
/final_1/u3/c	1001110
/final_1/u3/s	1110011
/final_1/u3/y	1111111111111111
/final_1/u3/h	{100000010001001
/final_1/u3/l	{000000000000000
/final_1/u3/e	0000000011000000
Now	1500 ns
Cursor 1	1122 ns

Fig. 20 Output of FUEC-DAEC simulation.

Clk: 1
Input data: 1111111111111111
Error value: 1111111100111111
(23,16) encoder outputs :1001110
Adress to store data in memory: 00000101
Output data:1111111111111111

3. The below Fig. 21, show the delay, area, and power values generated from Xilinx software for FUEC-DAEC.



```
Timing Summary:
---------------
Speed Grade: -7

   Minimum period: 9.238ns (Maximum Frequency: 108.249MHz)
   Minimum input arrival time before clock: 11.725ns
   Maximum output required time after clock: 10.625ns
   Maximum combinational path delay: 11.435ns
```

Total Number 4 input LUTs	391	13,824	2%
Number used as logic	136		
Number used as a route-thru	47		
Number used for 32x1 RAMs	192		
Number used as 16x1 RAMs	16		
Number of bonded IOBs	71	510	13%
Number of GCLKs	1	4	25%
Number of GCLKIOBs	1	4	25%

	Voltage (V)	Current (m	Power (mW
Vccint	1.8		
Dynamic		173.09	311.57
Quiescent		15.00	27.00
Vcco33	3.3		
Dynamic		0.00	0.00
Quiescent		2.00	6.60
Total Powe			345.17
Startup Curre		500.00	
Battery Capacity (mA Hours)			0.00
Battery Life (Hours)			0.00

Power summary:	I(mA)	P(mW)
Total estimated power consumption:		345
Vccint 1.80V:	188	339
Vcco33 3.30V:	2	7
Clocks:	165	296
Inputs:	8	15
Logic:	0	0
Outputs:		
Vcco33	0	0
Signals:	0	0
Quiescent Vccint 1.80V:	15	27
Quiescent Vcco33 3.30V:	2	7

Fig. 21 Delay, area, and power values of FUEC-DAEC.

DOI 10.17148/IJIREEICE.2020.8507

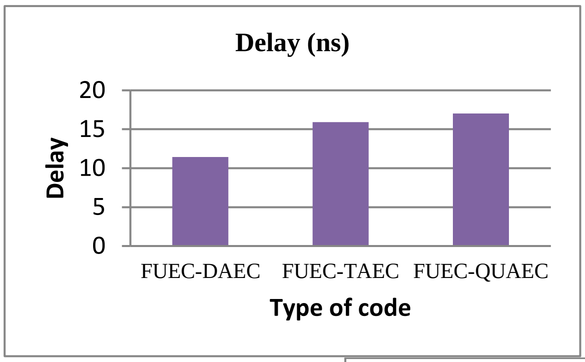
Similarly, we can find area power and delay in Xilinx and simulation waveform through ModelSim for FUEC-TAEC, FUEC-QUAEC, D-DMC, T-DMC, Q-DMC.

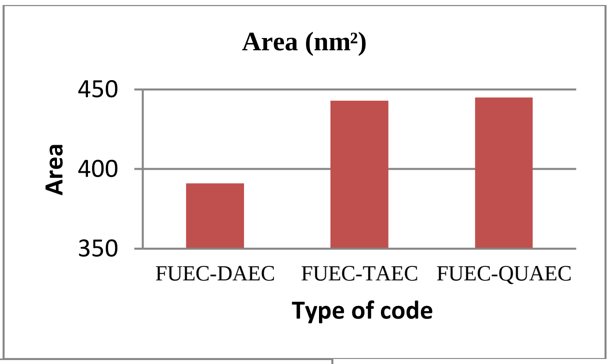
E. Performance Analysis
The delay, area, and power of the new FUECs codes are obtained and the results are compared and shown in Table V.

TABLE V COMPARISON TABLE OF FUECS

Code	Delay	Area	Power
FUEC-DAEC	11.438ns	391	345
FUEC-TAEC	15.920ns	443	369
FUEC-QUAEC	17.027ns	445	374

It can be visualized from the Table V, that the FUEC-DAEC has low area, power and delay. FUEC-DAEC is more efficient when comparing with other structure in area, power and delay.





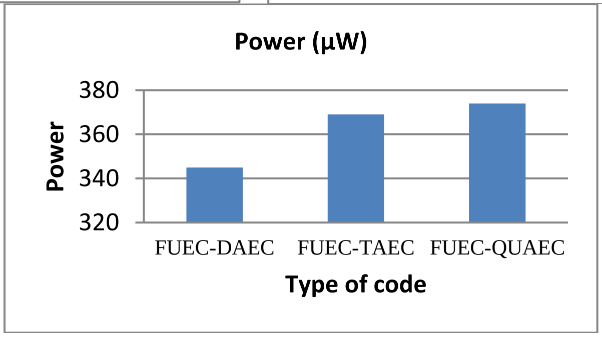


Fig. 22 Graphical representation of FUECs

Nevertheless, the probability of occurrence of burst errors decreases significantly when increasing its length. Fig. 22 show graphical representations of FUEC-DAEC, FUEC-TAEC, and FUEC-QUAEC. From this, FUEC-DAEC has low delay, area, and power due to an error in two adjacent bits and or it can detect one 3-bit burst error or one 4-bit burst error.

TABLE VI COMPARISON TABLE OF DMCS.

Code	Delay	Area	Power
D-DMC	10.402ns	538	410
T-DMC	10.559ns	630	421
Q-DMC	10.622ns	942	448

It can be visualized from the Table VI. that the D-DMC has low area, power and delay. Fig. 23 shows graphical representations of D-DMC, T-DMC, and Q-DMC. From this, D-DMC has low delay, area, and power due to an error in two random bits and or it can detect one 3-bit random error or one 4-bit random error.

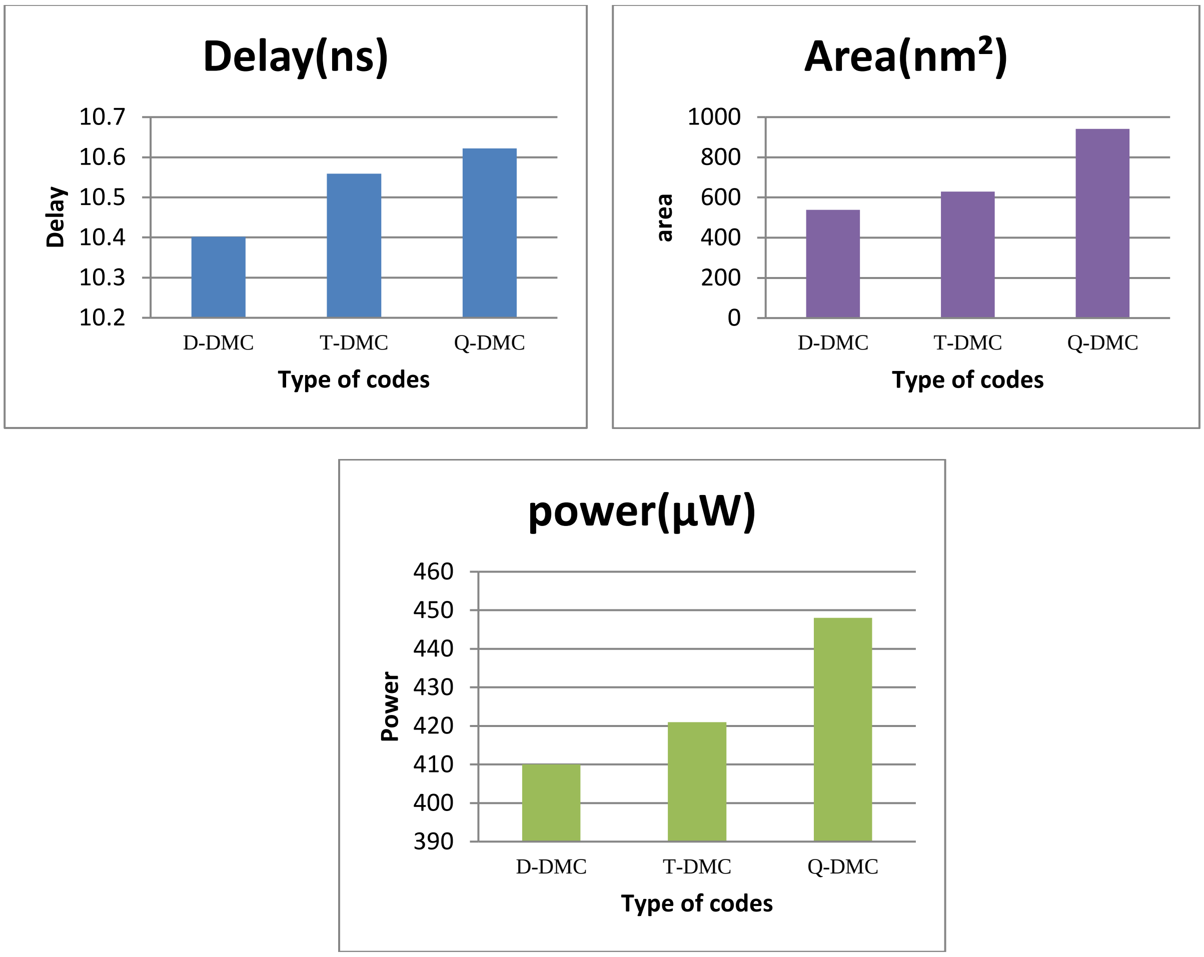


Fig. 23 Graphical representation of DMCs

From this, observe dual random errors, triple random errors and quarter random errors. And also, calculate their area, power and delay. Beyond 4-bit random errors the performance of code decreases due to the redundancy. Delay can be reduced but power and area increase due to the random error increases. So, that all errors can be corrected and detected at the expense of least time consumption.
Comparing these methodologies, D-DMC has low delay than FUEC-DAEC but FUEC-DAEC has low area and power than D-DMC. It is mainly due to the error in random bit need more area to occupy. And also need more power than adjacent errors or burst errors. Codes are very efficient to tolerate burst errors of from 2- to 4-bit length. Beyond 4-bit burst errors, the performance of our codes decreases notably due to their low redundancy. If these errors are expected to occur, more powerful codes (with a higher redundancy) must be employed.

IV.CONCLUSION

In this paper, a series of new ECCs has been presented. ECC code is a very powerful technique to correct MCUs in memory. These new ECCs improve the behaviour of the well-known Matrix code, the recently introduced CLC code and different SEC–DAED codes. A characteristic of our codes is the low redundancy they introduce. This fact provokes a reduction in the storage needed for the code bits. In addition, the detection and correction capabilities are maintained, or even increased. The insertion of an ECC in memory also provokes the introduction of area, power, and delay. FUEC–DAEC code introduces a much lower area, delay and power overhead than the CLC and Matrix codes, while FUEC–TAEC code presents a similar area overhead than Matrix and CLC codes. As expected, FUEC–QUAEC code exhibits the highest overhead, related to its high correction and detection capabilities. Reduce memory area due to the low redundancy of codes. Concerning the power overhead, the trend is similar to the area. FUEC–DAEC code introduces a much lower power overhead than the CLC and Matrix codes. Even, the power overhead of the FUEC–DAEC code is similar to the SEC–DAED codes ones. FUEC–TAEC and FUEC–QUAEC codes present a power consumption similar or a little bigger than CLC code, but with much better error correction and detection capabilities. And also, we have to take into account the reduction of memory power consumption due to the low redundancy of our codes. With respect to the delay, FUEC–DAEC presents the fastest correction, even better than both SEC–DAED codes. On the contrary, FUEC–TAEC and FUEC–QUAEC codes introduce the highest correction delay, because they are designed to correct longer burst errors. Similarly, for random errors three codes are created D-DMC, T-DMC, and Q-DMC.D-DMC has lower delay, area, and power overhead than other codes in random error correction. Comparing these methodologies, D-DMC has low delay than D-FUEC but D-FUEC has low area and power than D-DMC. It is mainly due to the error in random bit need more area to occupy. And also need more power than adjacent errors or burst errors. Beyond 4-bit burst errors, the performance of our codes decreases notably due to their low redundancy. If these errors are expected to occur, more powerful ECCs must be employed.

II. FUTURE SCOPE

In future work, continue developing ECCs, decreasing area, power, and delay overheads while maintaining, or even increasing, the code coverage. And also focus on long burst errors, which are expected to have more and more impact on space systems. Comparing the random error bits and burst errors with increasing the bits size (32, 64….) which one have low redundancy, area, delay, and power. Also, ECCs can develop using MATLAB software.

REFERENCES

[1]. Improving Error Correction Codes for Multiple-Cell Upsets in Space Applications Joaquín Gracia-Morán , Luis J. Saiz-Adalid, Daniel Gil-Tomás, and Pedro J. Gil-Vicente, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS 1

[2]. (2013). *The International Technology Roadmap for Semiconductors*. [Online].. Available: http://www.itrs2.net/2013-itrs.html

[3].S. K. Kurinec and K. Iniewski, *Nanoscale Semiconductor Memories: Technology and Application*. Boca Raton, FL, USA: CRC Press, 2014.

[4]. J. Barak, M. Murat, and A. Akkerman, "SEU due to electrons in silicon devices with nanometric sensitive volumes and small critical charge," *Nucl. Instrum. Methods Phys. Res. B, Beam Interact. Mater. At.*, vol. 287, pp. 113–119, Sep. 2012.

[5]. E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," *IEEE Trans. Electron Devices*, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[6]. G. Tsiligiannis *et al.*, "Multiple cell upset classification in commercial SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 4, pp. 1747–1754, Aug. 2014.

[7]. G. I. Zebrev, K. S. Zemtsov, R. G. Useinov, M. S. Gorbunov, V. V. Emeliyanov, and A. I. Ozerov, "Multiple cell upset cross-section uncertainty in nanoscale memories: Microdosimetric approach," in *Proc. 15th Eur. Conf. Radiat. Effects Compon. Syst. (RADECS)*, Sep. 2015, pp. 1–5.

[8]. N. Chechenin and M. Sajid, "Multiple cell upsets rate estimation for 65 nm SRAM bit-cell in space radiation environment," in *Proc. 3rd Int. Conf. Exhib. Satell. Space Missions*, May 2017, p. 77.

[9]. N. N. Mahatme, B. L. Bhuva, Y.-P. Fang, and A. S. Oates, "Impact of strained-Si PMOS transistors on SRAM soft error rates," *IEEE Trans. Nucl. Sci.*, vol. 59, no. 4, pp. 845–850, Aug. 2012.

[10]. Y. Bentoutou, "Program memories error detection and correction onboard earth observation satellites," *Int. J. Elect. Comput. Eng.*, vol. 4, no. 6, pp. 933–936, 2010.

[11]. M. J. Gadlage, A. H. Roach, A. R. Duncan, A. M. Williams, D. P. Bossev, and M. J. Kay, " Multiple-cell upsets induced by single high-energy electrons," *IEEE Trans. Nucl. Sci.*, vol. 65, no. 1, pp. 211–216, Jan. 2018, doi: 10.1109/TNS.2017.2756441.

[12]. E. Fujiwara, *Code Design for Dependable Systems: Theory and Practical Applications*. Hoboken, NJ, USA: Wiley, 2006.

[13]. R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.

[14]. C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," *IBM J. Res. Develop.*, vol. 58, no. 2, pp. 124–134, Mar. 1984.

[15]. G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault tolerant solid state mass memory for space applications," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 41, no. 4, pp. 1353–1372, Oct. 2005.

[16]. S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Error correction codes for SEU and SEFI tolerant memory systems," in *Proc. 24th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (DFT)*, Oct. 2009, pp. 425–430.

[17]. A. Sánchez-Macián, P. Reviriego, J. Tabero, A. Regadío, and J. A. Maestro, "SEFI protection for nanosat 16-bit chip onboard computer memories," *IEEE Trans. Device Mater. Rel.*, vol. 17, no. 4, pp. 698–707, Dec. 2017, doi: 10.1109/TDMR.2017.2750718.

[18]. C. Argyrides, D. K. Pradhan, and T. Kocak, "Matrix codes for reliable and cost efficient memory chips," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 3, pp. 420–428, Mar. 2011.

[19]. C. Argyrides, H. R. Zarandi, and D. K. Pradhan, "Matrix codes: Multiple bit upsets tolerant method for SRAM memories," in *Proc. 22nd IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Sep. 2007, pp. 340–348.

[20]. H. S. de Castro, J. A. N. da Silveira, A. A. P. Coelho, F. G. A. e Silva, P. D. S. Magalhães, and O. A. de Lima, "A correction code for multiple cells upsets in memory devices for space applications," in *Proc. 14th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2016, pp. 1–4.