LPC: An Error Correction Code for Mitigating Faults in 3D Memories

David C. C. Freitas, David F. M. Mota, César Marcon, Jarbas A. N. Silveira, João C. M. Mota

Abstract—The radiation sensitivity of memory cells increases dramatically as CMOS manufacture technology scales down; therefore, the reliability of memories has become a challenge. 3D technology has gained attention for having several advantages compared to the 2D counterpart, such as high integration density, high performance, low power, and high communication speed. Although several studies are targeting 3D memories, the effects on reliability using this technology have received little attention. This work introduces Line Product Code (LPC), a modified product code-based Error Correction Code (ECC) that uses both Hamming and parity in both rows and columns to implement reliable 3D memories. We implemented two lightweight LPC-based decoding algorithms in interleaved (LPCa-I) and non-interleaved (LPCa) versions, which allowed us to analyze LPC through a set of simulation cases that considers four severity levels of error incidence. The experimental results showed the effectiveness of the LPC-based algorithms, reaching correction rates of up 2.3 times higher compared to other Hamming-based algorithms.

Index Terms—Error Correction Codes, Fault Tolerance, Radiation Effect, 3D Memories.

1 Introduction

THE market demand for complex applications boosts researches on CMOS manufacturing technologies that carried a significant reduction in the transistor size [1]. In turn, the transistor scaling down contributes to the increase of temporary faults in electronic elements, such as memories, whose content modification may cause a wrong execution of programs that may not be tolerated in some cases [2]. These faults have been studied for over 40 years [3]-[6] and are classified as Single-Event Upset (SEU), Multiple-Cell Upsets (MCU), and Multiple-Bit Upsets (MBU). SEU occurs in a single cell while an MCU arises in more than one cell; finally, an MBU happens when an MCU occurs in the same logical word [7].

There are several techniques for mitigating these faults in electronic devices, such as improving the process technology, using hardened memory cell, Triple Modular Redundancy (TMR) or Error Correction Code (ECC). To minimize faults, Silicon on Insulator (SoI) technology uses a thin layer of silicon on top of the insulator during the chip manufacturing process. In the hardened memory cell approach, some circuits are replaced by their hardened versions, which are less susceptible to faults but consuming more area and implying more latency. The TMR technique uses three identical implementations of the same logic function, and the outputs are connected to a voter that decides mostly the correct result [8]. Lastly, the ECC basic concept is to have an encoding and decoding algorithm for restoring the correct value of the information placed in a memory cell or transmission channel [9].

The evolution of manufacturing technology reaches significant reductions in Two-Dimensional (2D) memories, increasing the challenges to reach reliable circuits [1]. Recently, Three-Dimensional (3D) integration technology, which enables multi-layer stacking, has attracted attention – Section 3 gives some recent works targeting reliability on 3D devices. The advantages of 3D technology include high integration density, high performance, low power dissipation, and high on-chip communication speed [10]-[17]. Section 2 describes another advantage of the 3D integration technologies - stacking several dies on top of each other suggests that incident particles must penetrate multiple layers of material before reaching transistors on the inner layers. Thus, stacked dies can block some particles before reaching deeper layers of the 3D chip, changing the Soft Error Rate (SER) at different 3D chip dies [2]. This stacking effect on SER is one of the essential points that this paper regards to evaluate the ECCs capacity.

This work also addresses the problem of chip warming; especially for 3D memories placed on the top of the active logic, the bottom layer of the memory is the most exposed to heat dissipation, making the bottom layer in 3D memory hotter than the top ones. Heat is another source of transient errors, and the heat profile in 3D memory provides varying degrees of reliability for each layer. From a heat perspective, the upper dies are less susceptible to errors, forming a different SER distribution in 3D memory [13]. Besides, the performance benefits and thermal impact of the stacked 3D microarchitecture have been studied recently, but the reliability implications and the MBU patterns when using 3D technology have received little attention.

The novelty of this work is to propose the Line Product Code (LPC), a new product-type ECC, to increase the correction rate and reliability of 3D memories. LPC is a lighter ECC that does not employ the redundancy overhead of the straightforward product codes, providing a decoding algorithm elaborated to achieve a high error correction rate. Section 4 details the LPC organization and two lightweight

- *David Freitas, David Mota, Jarbas Silveira and João Mota are with the Federal University of Ceará, Fortaleza, CE, Brazil. E-mail: {davidciarlinifreitas, davidfmmota, jarbassilveira}@gmail.com, mota@gtel.ufc.br.*
- *César Marcon is with the Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, RS, Brazil. E-mail: cesar.marcon@pucrs.br.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

decoding algorithms based on LPC - LPCa and LPCa-I, which are non-interleaved and interleaved versions. Besides, we described two other ECC configurations used in the analysis of this work. Section 5 explains how the organization of LPC is mapped on a physical memory. Section 6 presents the experimental results with the test sequence and an MBU generation algorithm, which is suitable to represent faults on 3D memories. Section 7 demonstrates that the LPCa-based algorithms achieve high error correction rates and are suitable for use in applications where reliability is a critical requirement, such as in space applications. Besides, Section 7 includes the results and discussions based on code correctability, reliability, and computational cost.

2 Soft Error Rate Analysis on 3D Memories

Due to the physical structure of 3D technologies, the upper layers protect the lower ones from high energy particles. Zhang and Li [2] analyze SERs for 3D-ICs based on the effect of alpha particles emitted from the decay of radioactive impurities in the interconnect metallization and package material. The authors state that the flux of alpha particles generated from the Integrated Circuit (IC) plastic packaging material is almost ten times greater than that of the metallization layers, and the metallization layers block more than 30% of these particles before reaching an active layer; besides, only 0.4% of the particles can reach the active layer of a second die from the top. Thus, the lower layers of a 3D memory have lower error rates than the higher ones, which is an advantageous feature of 3D technologies [12][13][17][18].

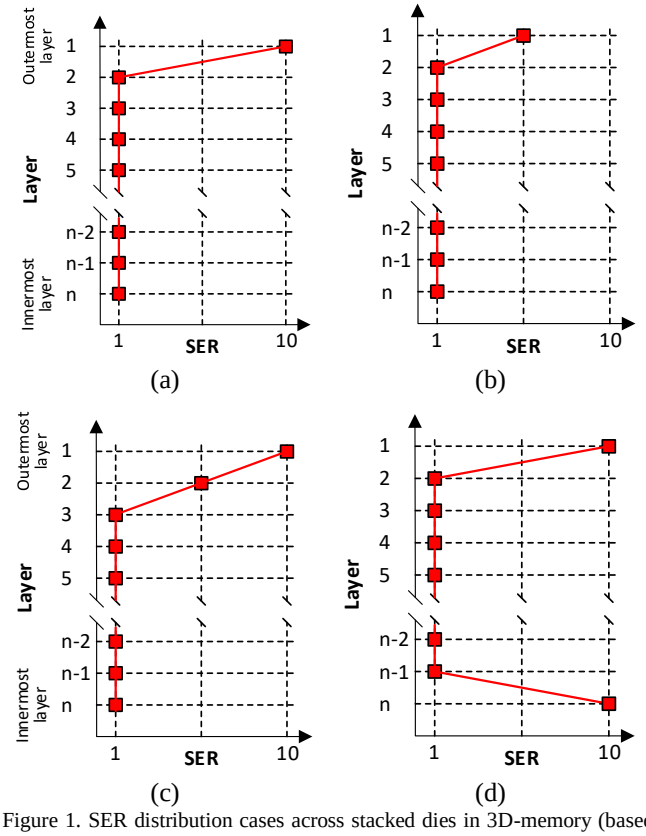


Figure 1. SER distribution cases across stacked dies in 3D-memory (based on [13]).

Additionally, high temperature is another source of transient errors, and the stacked architecture causes a heating problem since the lower layers of memory are less exposed to the heat dissipation, making them warmer more than the upper layers; thus, the heat profile in 3D memory provides different degrees of reliability for each layer. From a heat perspective, this makes the lower dies more susceptible to errors, forming an unequal distribution of SER through the 3D layers [13][18].

Han, Chung, and Yang [13] used both the effect of radiation and heat to produce a model of equations to estimate SER among the layer levels of a 3D-IC. Figure 1 depicts four test cases created by the authors using these equations: (a) SER of the uppermost layer is 10× higher than the other layers. This case is based on the analysis introduced in [2], which only considers the effect of alpha particles strike on the top layer, (b) SER is 5× higher than the others since the flux of alpha particles is reduced, (c) SER of the first and second layers are respectively 10× and 5× higher than the other layers, and (d) SER of the uppermost and lowermost layers are 10× higher than the others. This case regards the strike of alpha particles and the heat dissipation from an active layer bellow the stacked memories.

3 State-of-the-Art

There are some works that investigate reliability issues on 3D devices. For instance, Bagatin et al. [14] investigated the sensitivity of 3D NAND flash memories to wide-energy spectrum neutrons. The effects of neutron exposure were studied in terms of threshold voltage shifts and raw bit error rates; they extrapolated the neutron failure rates obtained in the accelerated tests to field conditions at sea level and aircraft altitudes. Kim and Yang [18] proposed a reliability structure for reducing faults on the bits, which considers asymmetric SERs per layer in a 3D die-stacked memory using a deep neural network. Their experimental results demonstrate that the proposed method improves fault tolerance regardless of the model type.

The works [1] and [15] also focus on 3D NAND flash memories. Bagatin et al. [1] investigated the effects of heavy-ion irradiation on 3D memory cells; threshold voltage distributions are studies before and after exposure, as a function of the irradiation angle. The same authors investigated in [15] the effects of total ionizing dose on 3D memories irradiated with gamma rays.

Finally, we describe three works that analyze ECCs targeting 3D memories. Han, Chung, and Yang [13] proposed a novel ECC organization scheme for 3D memories to secure reliable operations under SER profiles. The proposed scheme does not require additional redundant arrays. Instead, it employs unused spare columns of relatively reliable layer memories to store additional check-bits of less reliable layer memories. Chang, Huang, and Li [17] proposed an area and reliability-efficient ECC scheme for 3D RAMs, taking advantage of the shielding effect. Han and Yang [12] introduced a 3D memory scheme to ensure reliable operations by enhancing the ECC capacity of upper layer memories. Experimental results show that the proposed method can tolerate more than three times the bit

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

error rate than the conventional method. These three papers introduce new ECC configurations in 3D components, but there is no standard in how to present the results. Our work focuses on the presentation through three metrics: ECC correctability, hardware cost analysis, and system reliability. Besides, this work introduces a methodology to generate synthetic upsets considering radiation and heat effects on 3D devices.

4 Hamming and Modified Line Product Code (LPC) Foundations

R. Hamming [19] proposed a linear block code for error correction, whose generic structure is shown in Figure 2.

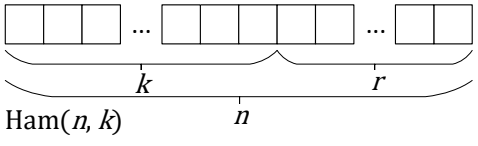


Figure 2. Representation of a generic Hamming code Ham(n, k); n is the total number of bits, k is the number of data bits, and r is the redundancy.

The Hamming code is denoted as Ham(n, k), where n, k and r are the numbers of bits of the codeword, data and redundancy, respectively. Equations 1, 2 and 3 describe the relations among n, r and k [19].

$$n = r + k \tag{1}$$

$$r = \log_2(n + 1) \tag{2}$$

$$k = 2^r - r - 1 \tag{3}$$

Extended Hamming is a Hamming code having one more parity bit that increases the code capacity to detect double errors and correct a single error, i.e., an SEC-DED code [20]. The parity bit can be either 0 or 1, depending on the parity type (i.e., even or odd). This work uses even parity so that the total of 1s in the codeword, including the added parity bit, is even [21]. Also, we employ Ham(8, 4) (n=8, k=4, and r=4), as the basic Hamming format to produce the other codes used in this work.

Figure 3 displays that given the two linear codes $C_1(n_1, k_1)$ and $C_2(n_2, k_2)$, then the modified product code is the combination of both codes (n_1n_2, k_1k_2) without checks on check bits, which is denoted by C_1C_2. The data is written in a matrix k_1k_2. Each one of the k_2 rows is encoded using code C_1, forming n_1 columns. Each one of the k_1 columns is encoded using code C_2, performing a matrix with $n_1n_2 - (n_1 - k_1) \times (n_2 - k_2)$ bits.

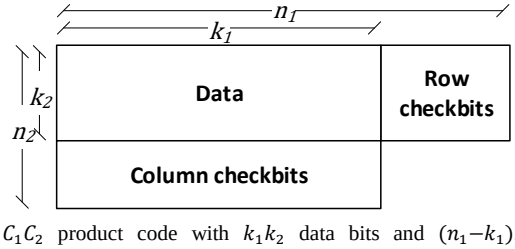


Figure 3. C_1C_2 product code with k_1k_2 data bits and $(n_1-k_1)k_2 \times (n_2-k_2)k_1$ check bits in rows and columns.

The linearity of the modified product code is the same as that of the product code, which allows coding to begin with C_1 followed by C_2, or vice versa [22]-[24]. The Minimum Distance (MD) between two codes of the same length informs the number of positions in which the codes differ. Since C_1 and C_2 have MDs defined as d_1 and d_2, respectively, Equation 4 shows how to compute the MD of the modified product code C_1C_2. Using MD as a metric, we can define mathematically the minimum of errors that can be corrected ($ECap$) or detected ($EDap$) in any position of the code with a 100% confidence [22]. Equations 5 and 6 compute the maximum number of errors *in any position* that this code can, at least, correct $ECap$ or detect $EDap$ [22], respectively. Note that the modified product code can correct more than $ECap$, depending on the placement of the errors.

$$MD = d_1 + d_2 - 1 \tag{4}$$

$$ECap = \frac{MD - 1}{2} \tag{5}$$

$$EDap = MD - 1 \tag{6}$$

LPC is a modified product code-based ECC, which uses Extended Hamming code. Figure 4 exemplifies LPC in a (48, 16) code format, wherein a 16-bit word (represented by bits D_0-D_{15}) is encoded into 48 bits distributed as follows: (i) 16 data bits - D, (ii) 12 row-check bits - CR, (ii) 4 row-parity bits - PR, (iii) 12 column-check bits - CC, and (iv) 4 column-parity bits - PC.

D_0	D_1	D_2	D_3	CR_0	CR_1	CR_2	PR_0
D_4	D_5	D_6	D_7	CR_3	CR_4	CR_5	PR_1
D_8	D_9	D_{10}	D_{11}	CR_6	CR_7	CR_8	PR_2
D_{12}	D_{13}	D_{14}	D_{15}	CR_9	CR_{10}	CR_{11}	PR_3
CC_0	CC_1	CC_2	CC_3				
CC_4	CC_5	CC_6	CC_7				
CC_8	CC_9	CC_{10}	CC_{11}				
PC_0	PC_1	PC_2	PC_3				

Figure 4. LPC structure encompassing five regions of bits: data (D), row-check (CR), column-check (CC), row-parity (PR), and column-parity (PC).

The bits organization of LPC makes $d_1 = d_2 = 4$; consequently, applying Equations 4, 5 and 6, LPC has $MD = 7$ and at least $ECap = 3$ or $EDap = 6$. Nevertheless, depending on the position of the errors, applying elaborated decoding algorithms, LPC can correct until seven bitflips into the data field and until 20 bitflips regarding data and control bits (see examples and comments on Figure 5), only using Hamming and some logical rules.

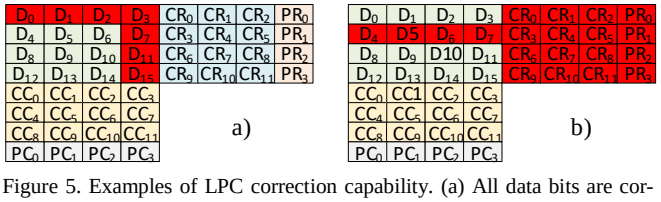


Figure 5. Examples of LPC correction capability. (a) All data bits are corrected by computing the syndromes and applying Hamming on the last three rows; then, recalculate the syndromes of all columns and apply Hamming in all columns to correct the remaining four errors. (b) Discard all row check and parity bits and apply Hamming on the columns; next, recompute the check and parity bits of the rows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

Let q be a bit position index, then Equations 7 to 9 and 10 to 12 compute, using XOR ($\oplus$) operations, the recalculated check bits of rows (rCR) and columns (rCC), respectively. Additionally, Equations 13 and 14 compute the recalculated parity bits of rows (rPR_q) and columns (rPC), respectively.

$$rCR_{3q} = D_{4q} \oplus D_{4q+1} \oplus D_{4q+3} \quad \forall\, 0 \le q \le 3 \quad (7)$$

$$rCR_{3q+1} = D_{4q} \oplus D_{4q+2} \oplus D_{4q+3} \quad \forall\, 0 \le q \le 3 \quad (8)$$

$$rCR_{3q+2} = D_{4q+1} \oplus D_{4q+2} \oplus D_{4q+3} \quad \forall\, 0 \le q \le 3 \quad (9)$$

$$rCC_q = D_q \oplus D_{q+4} \oplus D_{q+12} \quad \forall\, 0 \le q \le 3 \quad (10)$$

$$rCC_{q+4} = D_q \oplus D_q \oplus D_{q+4} \quad \forall\, 0 \le q \le 3 \quad (11)$$

$$rCC_{q+8} = D_{q+4} \oplus D_{q+8} \oplus D_{q+12} \quad \forall\, 0 \le q \le 3 \quad (12)$$

$$rPR_q = D_{4q} \oplus D_{4q+1} \oplus D_{4q+2} \oplus D_{4q+3} \oplus CR_{3q} \oplus CR_{3q+1} \oplus CR_{3q+2} \quad \forall\, 0 \le q \le 3 \quad (13)$$

$$rPC_q = D_q \oplus D_{q+4} \oplus D_{q+8} \oplus D_{q+12} \oplus CC_q \oplus CC_{q+4} \oplus CC_{q+8} \quad \forall\, 0 \le q \le 3 \quad (14)$$

LPC allows verifying and correcting data errors using the syndromes of each row and column, which are computed by Equations 15 to 18.

$$sCR_q = (CR_{3q} \oplus rCR_{3q}) \text{ OR } (CR_{3q+1} \oplus rCR_{3q+1}) \text{ OR } (CR_{3q+2} \oplus rCR_{3q+2}) \quad \forall\, 0 \le q \le 3 \quad (15)$$

$$sPR_q = PR_q \oplus rPR_q \quad \forall\, 0 \le q \le 3 \quad (16)$$

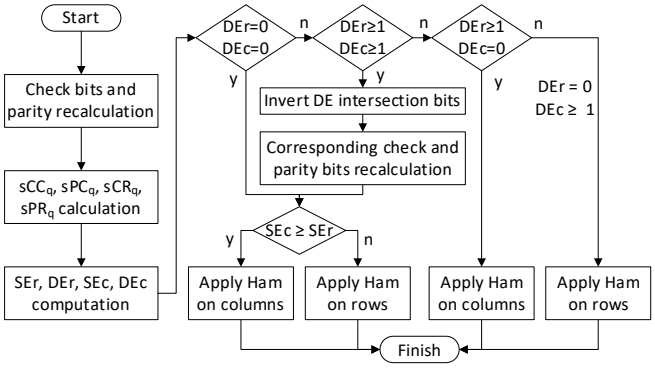
$$sCC_q = (CC_q \oplus rCC_q) \text{ OR } (CC_{q+7} \oplus rCC_{q+7}) \text{ OR } (CC_{q+14} \oplus rCC_{q+14}) \quad \forall\, 0 \le q \le 3 \quad (17)$$

$$sPC_q = PC_q \oplus rPC_q \quad \forall\, 0 \le q \le 3 \quad (18)$$

4.1 LPC-based Decoding Algorithm

The decoding algorithm can explore full code potentialities or implement a more lightweight version to reduce synthesis costs. This work introduces LPCa and LPCa-I, two LPC-based decoding algorithms in non-interleaved and interleaved versions, respectively. Both LPC-based algorithms have the same correction method that explores double and single error knowledge to perform a heuristic technique that reaches high error correction rates, without increasing a lot the implementation cost of the decoding algorithm. LPCa and LPCa-I differ only on the codeword organization into the target memory.

Figure 6 displays that the LPC-based algorithms start recalculating the check and parity bits to compute the syndromes of all columns and rows. Next, the algorithms calculate SEr, DEr, SEc, and DEc, which are the single and double errors, both in rows and columns, respectively.



Remark: "Apply Ham" means correct only single errors using Hamming in a word

Figure 6. Main flow of the LPC-based algorithms.

Table 1 shows the types of errors associated with each pair of syndromes (check and parity bits) for any row or column. Double errors are detected when the check bit syndrome is true, but the parity syndrome does not point out an error.

TABLE 1
Error Type Regarding Syndromes of Rows and Columns

$[sCR_q, sPR_q]$ or $[sCC_q, sPC_q]$	*Error type*
[0, 0]	No error
[0, 1]	Error in the parity bit
[1, 0]	Double error
[1, 1]	Single error

Based on Table 1, Equations 19 and 20 compute SEr_q and SEc_q, and Equations 21 and 22 compute DEr_q and DEc_q, both for each q-row and q-column, respectively.

$$SEr_q = \left([sCR_q, sPR_q] = [1,1]\right) ? 1 : 0 \quad \forall\, 0 \le q \le 3 \quad (19)$$

$$SEc_q = \left([sCC_q, sPC_q] = [1,1]\right) ? 1 : 0 \quad \forall\, 0 \le q \le 3 \quad (20)$$

$$DEr_q = \left([sCR_q, sPR_q] = [1,0]\right) ? 1 : 0 \quad \forall\, 0 \le q \le 3 \quad (21)$$

$$DEc_q = \left([sCC_q, sPC_q] = [1,0]\right) ? 1 : 0 \quad \forall\, 0 \le q \le 3 \quad (22)$$

SEr and SEc, and DEr and DEc contain the sum of single and double errors, both on rows and columns, being computed by Equations 23, 24, 25 and 26, respectively.

$$SEr = \sum_{q=0}^{3} SEr_q \quad \forall\, 0 \le q \le 3 \quad (23)$$

$$SEc = \sum_{q=0}^{3} SEc_q \quad \forall\, 0 \le q \le 3 \quad (24)$$

$$DEr = \sum_{q=0}^{3} DEr_q \quad \forall\, 0 \le q \le 3 \quad (25)$$

$$DEc = \sum_{q=0}^{3} DEc_q \quad \forall\, 0 \le q \le 3 \quad (26)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

Figure 7 displays the composition of SEr, SEc, DEr, and DEc from the syndromes and the recalculated check bits and parity of the rows and columns graphically.

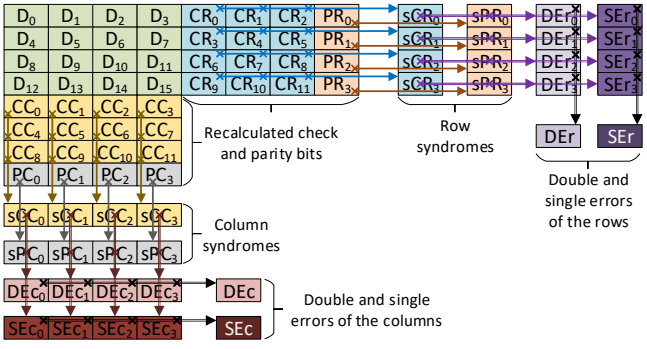


Figure 7. Graphical representation of SEr, SEc, DEr, and DEc compositions.

Next, LPCa selects the correction procedure according to DEr and DEc. If $DEr = 0$ and $DEc = 0$, LPCa decodes the codeword applying Hamming to rows or columns depending on where most errors occurred; if most errors occurred on the rows ($SEr > SEc$), the algorithm corrects single errors using Hamming on the rows and use Hamming on the columns, in the opposite situation. This decision is performed because it increases the probability of correcting more errors. Figure 8 exemplifies a situation where the variables point out two single errors on columns ($SEc = 2$) and four single errors on rows ($SEr = 4$), but none double errors. Therefore, the decoding algorithm applies Hamming on the rows. This procedure enables us to correct all errors. Note that if the decoding algorithm decided to apply Hamming on the columns, it would correct a false error in the second column.

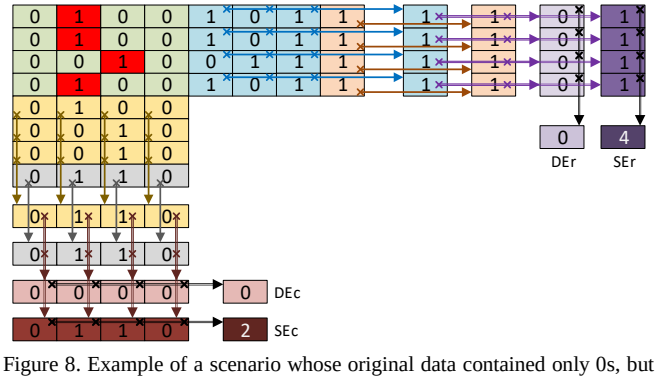


Figure 8. Example of a scenario whose original data contained only 0s, but four bits were flipped. Although there is a triple error in the second column, the variables of the decoding algorithm detect it as a single error.

If $DEr \geq 1$ and $DEc \geq 1$, LPCa starts by checking if there is an intersection of rows and columns where a double error occurs and inverts this bit. Next, LPCa corrects rows or columns, depending on the comparison $SEc \geq SEr$; a similar procedure that occurs in the case of $DEr = 0$ and $DEc = 0$. One the one hand, the inversion of a DE intersection bit is a technique that allows correcting error scenarios where a single Hamming approach cannot act, thus increasing the code correction capacity. On the other hand, this technique requires recomputing the check and parity bits of the corresponding column or row after the bit inversion; consequently, increasing the implementation cost of the decoding algorithm.

Figure 9 exemplifies a scenario with two double errors and two single errors in both rows and columns. The first row and first column have a double error intersection, implying the inversion of bit D_0 (see Figure 7). Additionally, there is a double error intersection in bit D_{10}; however, LPCa only corrects the first occurrence of double errors the algorithm finds from left to right and from top to bottom of the code. Consequently, D_{10} is not inverted, precluding some correction of errors through Hamming. Next, LPCa correct bits D_1, D_4 and D_{11} through Hamming on columns because of $SEc = SEr$. After applying LPCa, the data field remains with two errors on bits D_4 and D_{11} since Hamming cannot correct the double error in the third column.

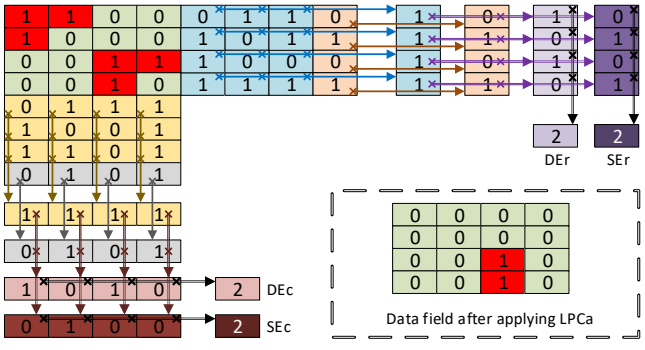


Figure 9. Example of a scenario whose original data contained only 0s, but six bits were flipped, performing two double errors and two single errors in both rows and columns. Within the dashed rectangle appears the data area after the decoding algorithm is applied.

The cases ($DEr \geq 1$ and $DEc = 0$) and ($DEr = 0$ and $DEc \geq 1$) are symmetric; the only difference is that the first case applies Hamming on columns, and the second one applies Hamming on rows. For both cases, LPCa does not check SEr and SEc; it assumes that rows or columns without double error can correct more possibilities. Due to the symmetry of these cases, we only show in Figure 10 an example of the case $DEr \geq 1$ and $DEc = 0$. This example of error scenario makes LPCa decide for applying Hamming on columns, which allows correcting all single errors.

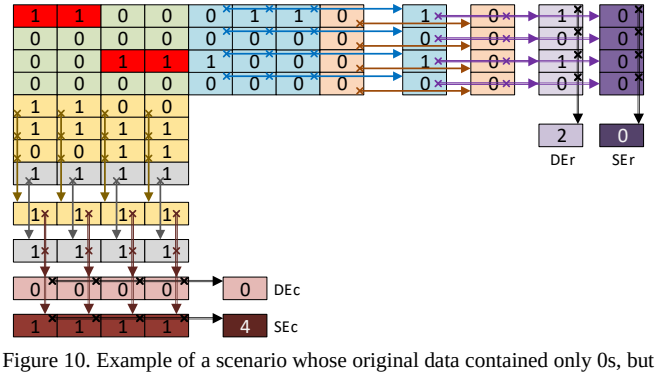


Figure 10. Example of a scenario whose original data contained only 0s, but four bits were flipped, performing two double errors on rows and four single errors in columns.

Note that if LPCa exploited all double error occurrences instead of only the first one, the scenario of Figure 9 would

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

be decoded without errors, but the limitation of applying the inversion only on the first double error incidence was made because we decided to implement a lightweight LPC-based decoding algorithm; consequently, with lower costs of implementation. Even so, Section 6 shows that LPCa reached an outstanding performance.

4.2 LPC Format Considerations

Several works aim to achieve high reliability by applying ECCs to memories with different technologies and organizations. A set of these works employs ECCs with low redundancy overhead (i.e., the ratio of the number of data bits versus the number of redundancy bits) to correct sporadic errors [25]-[29]. This type of ECC provides a low error correction rate, being suitable for environments that are not susceptible to faults and non-critical applications. However, critical applications running in environments that are more susceptible to faults demand more robust ECCs implemented with a more considerable redundancy overhead to achieve high error correction rates [30]-[33]. LPC belongs to this last class of ECCs, designed to increase significantly the ability to correct and detect errors.

LPC is an ECC based on the format of a product code composed of extended Hamming codes, without using the check bits of the check bits; therefore, multiple data bit sizes/redundancy can respect this general format. This work implements LPC with 16-bit data for two reasons explained next.

One of the reasons is choosing the best tradeoff between cost and correction/detection capacity. Ham(4, 1) is the smallest extended Hamming code, which codifies a single data bit using three redundancy bits (2 check bits and one parity bit), implying a high redundancy overhead. The second smallest extended Hamming encoding is Ham(8, 4) - the one employed here, where 4 data bits are encoded with four redundancy bits (3 check bits and one parity bit). The higher the number of data bits, the smaller the proportion of redundancy bits. However, the Hamming code only guarantees the correction of a single data bit; therefore, the higher the number of data bits, the smaller the relation between correctable bits and codeword size, which reduces the decoding efficacy. Thus, the 16-bit data is a natural consequence of the Ham(8, 4) spatially distributed as a product code.

Another reason is associated with reading and writing memory access latencies, as well as energy efficiency. Both are related to the applicability of the LPC encoding/decoding to existing processors and commercial memories. Although still exists 8-bit processors used for specific applications, they are in disuse. 16-bit processors and memories are still a reality for many embedded systems. For the case of a 16-bit processor/memory, the proposed LPC requires three memory accesses for each processor operation. If LPC had a larger data format, energy consumption and latency would be much higher, once more justifying the choice for LPC's 16-bit data format.

5 LPC Mapping on Memory

The LPC described in this work was implemented to be used in 16-bit memories, the size of memory used in the experimental results section. However, the coding model defined by LPC can be applied to memories with different manufacturing technologies, sizes, formats, and protocols, as it implements a coding layer that can be adapted to different types of reading and writing procedures in memory. For example, a logical organization of 32-bit memory words can be implemented employing LPC with two 16-bit data codes (i.e., 2×48 bits). Considering this example, the processor would only have one access for writing or reading, and the subsequent level implemented by an encoder/decoder adjusts the physical memory requirements. In this same example, assuming a physical memory with 32-bit words, three writes/readings to/from physical memory is needed to access the 96 bits required by the two LPC codewords. In this case, the encoder/decoder is responsible for converting the physical and logical words.

Figure 11 illustrates the encoding and decoding schemes considering various types of memories with specific reading and writing drivers to clarify the synthesized modules. It is important to note that while the ECC encoder and decoder modules are only dependent on the processor address/data size and ECC algorithms, the driver modules are memory configuration dependents.

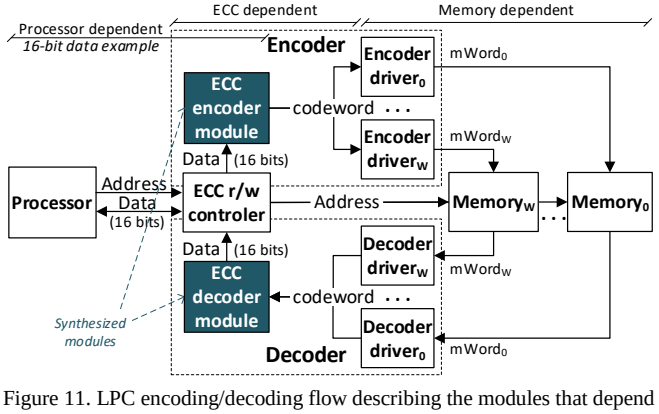


Figure 11. LPC encoding/decoding flow describing the modules that depend on the processor, ECC, and memory characteristics.

Additionally, the 3D fault model makes room for the codeword to be addressed in multiple layers; this is because different paths have different rates of error incidences. Thus, the distribution of a word in more than one memory layer increases the probability of having fewer errors within the codeword and, consequently, more decoding success. Another opportunity is to use LPC in layers more susceptible to faults, such as the upper and lower memory layer, and use less robust ECCs (therefore, with a lower associated cost) in layers less susceptible to bitflips.

Finally, we point out that LPC can even be used as the standard for memories that require a high degree of reliability and employ ECC on-die [34][35], as in this case, the ECC becomes transparent to the memory controller. Implementing an on-die ECC allows the complete knowledge of the physical organization of the memory bits; this knowledge makes it possible to apply codes that implement, for example, interleaving techniques, further increasing the ability to correct bitflips concentrated in a memory neighborhood.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

6 Experimental Setup and Methodology

In the experimental setup, we choose to evaluate the potentials of LPCa with 16-bit memory, and we compared LPCa with Extended Hamming-based ECCs; both codes were implemented with interleaved (LPCa-I and 4×Ham(8,4)-I) and non-interleaved (LPCa and 4×Ham(8,4)) versions. Note that bit interleaving is a mitigation technique that makes an MBU appears as multiple single bit upsets in different codewords [36]. This technique is 100% efficacious when the physical MBU is lesser or equal than the interleaving scheme.

Figure 12 describes the methodology employed to obtain and evaluate the experimental results, covering a flow with the four main activities.

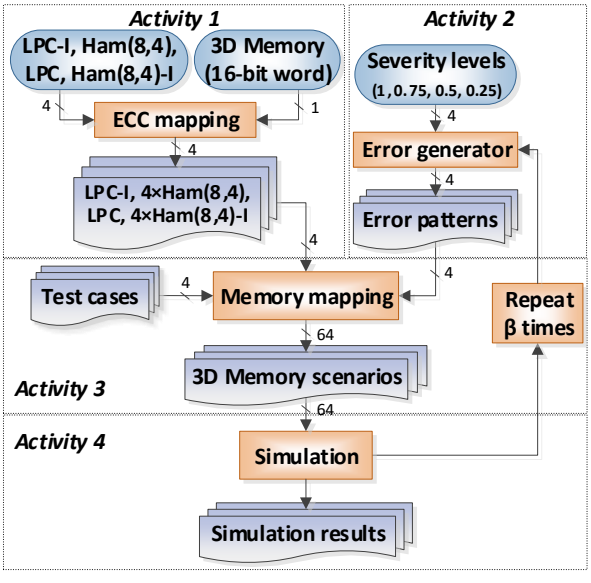


Figure 12. Methodology applied to the work containing the four main activities used in the experimental results.

Activity 1 represents the proposed ECC configurations with and without interleaving when placed in consecutive 16-bit memory words, as shown in Figure 13; all codes of Figure 13 cover exactly 16 data bits.

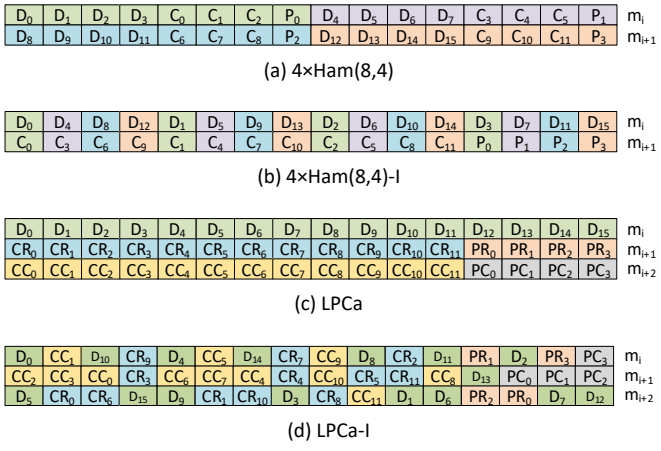


Figure 13. Memory organization of the ECCs after the encoding process (m_i is the ith memory position): (a) 4×Ham(8,4), (b) 4×Ham(8,4)-I (Hamming with interleaving), (c) LPCa, and (d) LPCa-I (LPCa with interleaving).

Figure 13a describes 4×Ham(8,4) - a configuration of four Ham(8,4) encoded into four 8-bit words. Figure 13b displays 4×Ham(8,4)-I - a configuration of four 8-bit interleaved words. Figure 13c describes the LPCa distribution in three consecutive 16-bit memory positions. Finally, Figure 13d illustrates the LPCa-I in an interleaved distribution, which was performed by an in-house tool that mapped the related bits in memory over a distance higher than a cell. The configurations 4×Ham(8,4) and 4×Ham(8,4)-I are implemented with k=4 and n=8; thus, only two memory addresses are required to encode 16-bit data. LPCa and LPCa-I, in turn, have k=16 and n=48, requiring three memory addresses to write 16-bit data.

The 3D experimental memory was implemented in four dies, each one containing six memory addresses with 16-bit words, achieving a die with 84 bits, as shown in Figure 14. We chose this reduced memory size to minimize the number of simulations performed without losing the generalization of the 3D error model.

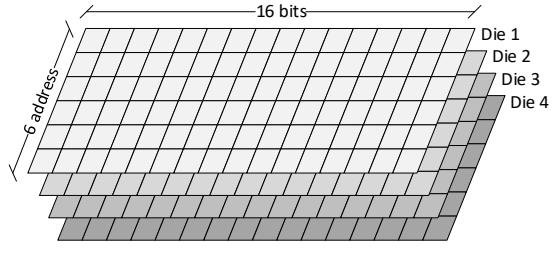


Figure 14. Experimental 3D memory, including four dies, each one with six memory addresses of 16-bit words.

Activity 2 represents the error pattern generation containing four levels of severity, which were created to represent how much energy the alpha particles have or how much the heat is affecting the memory layers. The higher the level, the greater the percentage of errors to be generated in the randomly selected layer, and consequently, the higher the number of errors propagated to neighboring layers.

Several works [37]-[39] show that the incidence of errors due to radiation or heat occurs within a neighborhood. For example, in the region of a radiation event, more than one neighboring cell may have its content changed. In this work, we proposed a fault model that considers a region composed of a cell considered the center of the event, all the neighboring cells of this event, making up an inner rectangle, and all the neighboring cells of the inner rectangle, making up the outer rectangle. Figure 15 illustrates that the error pattern is inserted in a region consisting in a reference cell (in red), which is virtually surrounded by an inner rectangle of cells (in green) that is wrapped by an outer rectangle of cells (in blue); thus, an error pattern can comprise from one to 25 error cells.

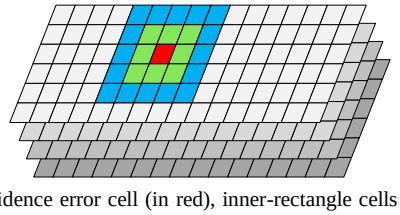


Figure 15. Incidence error cell (in red), inner-rectangle cells encompassing the incidence error cell (in green), and outer-rectangle cells around the inner one (in blue).

This work follows the assumptions for the error pattern generation based on the neighborhood models [37]-[39]:

a) *The radiation event changes the bit selected as a center of the error region called incidence error cell;*
b) *The inner rectangle surrounding the incidence error bit can have up to 8 random bitflips called ib (inner bitflips);*
c) *The outer rectangle that surrounds the inner rectangle can have up to 16 random bitflips called ob (outer bitflips);*
d) *To avoid an exhaustive analysis of 1 to 25 errors, we decided to use four levels of severity (ls) with values 1, 0.75, 0.50, and 0.25, representing the probability of error generation; ls is used as a multiplicative factor to produce the error pattern;*
e) *The total number of errors in a pattern (#e) is given by the equation #e = 1 + ib + ob, with ib = 8 × ls and ob = 16 × ls. For example, if ls = 0.75, ib = 6, ob = 12 and #e = 19.*

The levels of severity affect only the incidence die, producing an error pattern; the error mapping in the remaining dies depends on the selected Test case (Figure 1), which is performed in Activity 3.

Activity 3 is the error and ECC mapping in the 3D experimental memory, which starts fulfilling all memory with a selected ECC. The six memory addresses of each die allows fitting two LPCa, two LPCa-I, three 4×Ham(8,4) or three 4×Ham(8,4)-I, per die; therefore, the entire 3D memory allows mapping eight LPCa-based ECCs or 12 4×Hamming-based ECCs. Next, the error mapping algorithm fills errors in the dies of the 3D memory.

The error mapping algorithm starts randomly selecting one of the four test cases shown in Figure 1, which allows defining the model of error propagation among the dies and the die of the initial error incidence. Test cases (a)-(c) use only die 1 to map the initial error incidence, whereas test case (d) has two initial error incidences, one for die 1 and another one for die 4. Next, the algorithm randomly generates the initial cell of error incidence in the selected die, which can be one of the 96 memory cells; this cell is the center of the error pattern produced in Activity 2.

The next step of the algorithm defines how many errors are propagated to the neighbor dies and their random position relative to the cells of the initial error pattern. As previously described, the number of errors propagated depends on the radiation decay or heat effects among the dies. The error cells in the neighbor dies, respect the limits defined by the mask of the initial error pattern, can be non-adjacent, and use at least the same position of the initial error incidence (the center of the error pattern – cell in red). Besides, test case (d) propagates error patterns from dies 1 and 4, allowing for the composition of error patterns in the central dies.

Figure 16 exemplifies an error pattern and its propagation among the dies in the test case (c). Figure 16a depicts that the error pattern covers 19 errors in the first die, 7 in the inner-rectangle, and 11 in the outer-rectangle, besides the initial error incidence. Figure 16b and Figure 16c show that 6 and 2 errors were propagated to the die 2 and 3, respectively, while no error reached die 4 (Figure 16d). The propagation to neighboring dies follows randomly the error positions generated in the incidence layer, and with an error rate decay that respects the models defined in the test case of Figure 1 that is being used in the experiment.

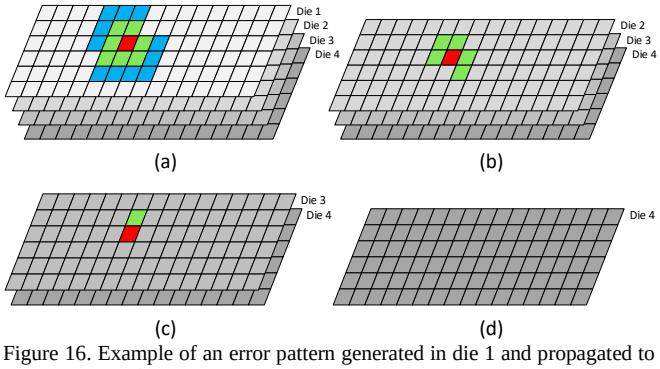


Figure 16. Example of an error pattern generated in die 1 and propagated to the other dies of the 3D memory.

According to the ECC mapped, Activity 4 uses in-house simulation software to analyze the error correction capability in each die of the 3D memory. We repeated β times the simulations of each one of the 48 experiments to have a certain degree of confidence and representativeness due to the number of random elements. In the experiments, we used β equal to 30 thousand, which allows reaching more than 99.9% confidence degree.

7 Experimental Results and Discussions

This section presents and discusses the error correction results achieved when applied the methodology described in Section 6, and analyzes the implementation costs and reliability of LPCa, LPCa-I, 4×Ham(8,4), and 4×Ham(8,4)-I.

7.1 Error Corrected Analysis

Figure 17 displays the average error correction rates of the four ECCs evaluated in each die, relative to four error severity levels and the four test cases described in Figure 1. The 3D error model makes die 1 (for all test cases) and die 4 (in the test case (d)) to receive the highest incidence of errors; consequently, the error correction capabilities of each ECC are better observed in these dies. Moreover, in the test case (c), the incidence of errors falls by only 50% in the second die; therefore, it also allows the observation of high variation among the correction capacities of each ECC. Only die 3 has little error correction need, as it is protected against alpha particles by two dies, and the heating from the lower die does not propagate errors significantly.

We focus the error correction analysis on die 1 because it is the die with the highest error incidence. In general, LPCa-I and LPCa have the highest correction capacity, followed by 4×Ham(8,4)-I, and with the lowest error correction rates, comes 4×Ham(8,4). LPCa-I obtained the highest rates of error correction in all cases; besides, LPCa has similar values than 4×Ham(8,4)-I only when the severity error level is 1.0, varying at most 2.63%. Additionally, the aggressiveness of error level 1, does not allow any code to achieve an error correction rate greater than 30% for test cases (a), (b), and (c).

The error correction capacity of LPCa-I is up to 2.3 times higher compared to 4×Ham(8,4) when regarding die 1 and all error severity levels, on average; but the error correction rate is reduced to 1.4 times higher when this same analysis is performed between LPCa-I and 4×Ham(8,4)-I.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

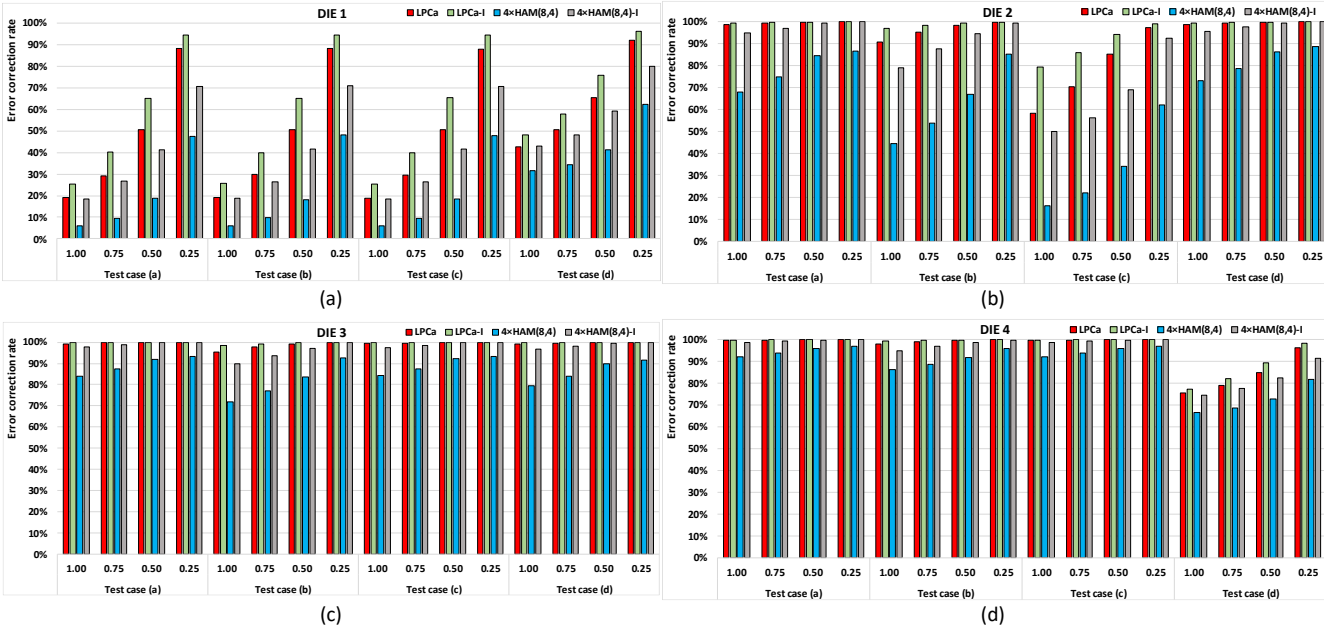


Figure 17. Correctability results for LPCa, LPCa-I, 4×Ham(8,4), and 4×Ham(8,4)-I in each die of the 3D memory, considering the four test cases shown in Figure 1, and four levels of error severity.

As die 3 is the most protected die against faults, all codes reach high error correction rates, on average, LPCa-I, LPCa, 4×Ham(8,4)-I and 4×Ham(8,4) reach 99.8%, 99.3%, 97.9% and 86.5% of error correction rates, respectively.

7.2 Analysis of the Algorithm Implementation Costs

We performed the implementation cost analysis based on the redundancy needs to implement LPCa-I, LPCa, 4×Ham(8,4)-I and 4×Ham(8,4), and on the area consumption (in μm²), power dissipation (in nW) and delay (in ps) of the encoders and decoders employed on these ECCs.

Let r be the number of redundancy bits, and n be the number of codeword bits; then, Equation 25 computes the redundancy rate rr, and Table 2 presents the rr results for the three ECC configurations.

$$rr = \frac{r}{n} \tag{27}$$

Both LPCa-based algorithms have the highest rr with about 66% of their bits in the codeword being redundant, whereas both Hamming-based codes have half of the codeword being redundant. This higher rr naturally conducts LPCa-based algorithms to higher error correction rate capacity and higher synthesis costs.

TABLE 2
Redundancy Rate Results

Configuration	rr(%)
LPCa(48, 16) and LPCa(48, 16)-I	66.7
4×Ham(8,4) and 4×Ham(8,4)-I	50.0

Figure 18 allows us to compare the synthesis costs of the four ECCs proportionally; the values shown are achieved with Cadence software synthesis RTL Compiler for a 65nm CMOS technology.

For all analyzed ECCs, the decoder synthesis values are higher than the encoder ones, which was expected since most of the calculations performed are made on the decoder side. LPCa and LPCa-I have the same synthesis cost, and both Hamming-based codes have practically equal synthesis values because the only difference between them is the interleaving technique. The area consumption and power dissipation of the LPC-based decoders are about 5× larger than the corresponding values of Hamming-based ECCs, whereas the delay of the LPC-based decoders is about 3.8× higher than the delays of the two other two ECCs.

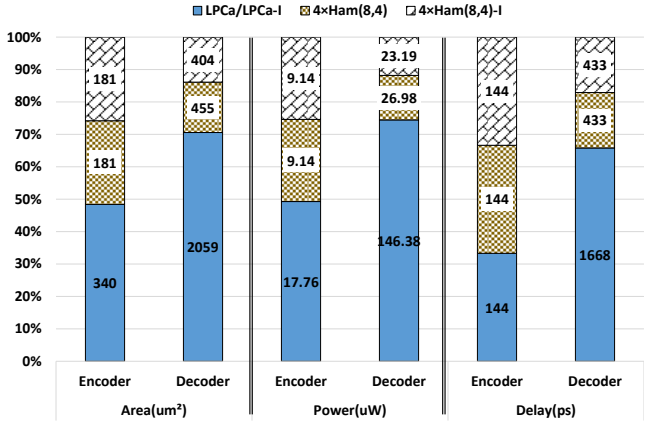


Figure 18. Synthesis cost analysis, encompassing area consumption, power dissipation, and delay of each ECC configuration.

7.3 Reliability Analysis

The reliability analysis of this work is based on the works of Silva et al. [20] and Argyrides et al. [40]. We assume the following statements that were also assumed by [40]: (i) transients faults occur with a Poisson distribution, and (ii) bit faults are statistically independent.

Let Ne be the maximum number of errors that can arise during time t, FC be the errors corrected, MF be a value that indicates if memory fails and iF be a value indicating

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2020.3034400, IEEE Transactions on Computers

i faults in the memory; then, Equation 2 computes the fault correction in a word $F_c(t)$ in a given time t.

$$F_c(t) = \sum_{i=1}^{Ne} (P\{FC|iF\} \times P\{iF|MF\}) \quad (28)$$

The probability of having exact i upsets in memory when memory is faulty can be reached by Equation 2.

$$P\{iF|MF\} = \frac{P\{iF\}}{P\{MF\}} \quad (29)$$

Let n be the number of bits in the codeword and λ be the one-bit fault per day; then, $P\{iF\}$ is given by Equation 30.

$$P\{iF\} = \binom{n}{i} \left(1 - e^{-\lambda t}\right)^i e^{-\lambda(n-i)t} \quad (30)$$

Equation 31 computes the probability of a memory failure over time.

$$P\{MF\} = 1 - e^{-\lambda n t} \quad (31)$$

Since $P\{FC|iF\}$ values are obtained in the previous section through the simulation results of the first die presented in Figure 17a and M is the number of words in memory, then the reliability of a memory $R(t)$ is the product of the reliability of all words, which is computed by Equation 32.

$$R(t) = \left(1 - P\{MF\} + \sum_{i=1}^{Ne} P\{iF\} \times P\{FC|iF\}\right)^M \quad (32)$$

For the sake of simplicity, this paper uses $M = 1$. Additional information on the equations can be found in [40].

Figure 19 shows the reliability over time $R(t)$ of LPCa, LPCa-I, 4×Ham(8,4), and 4×Ham(8,4)-I regarding the error correction rates of the die 1, and encompassing three values of λ (10^{-4}, 10^{-5}, 10^{-6}). The horizontal axis is time expressed in days, while the vertical axis is reliability $R(t)$ expressed in %.

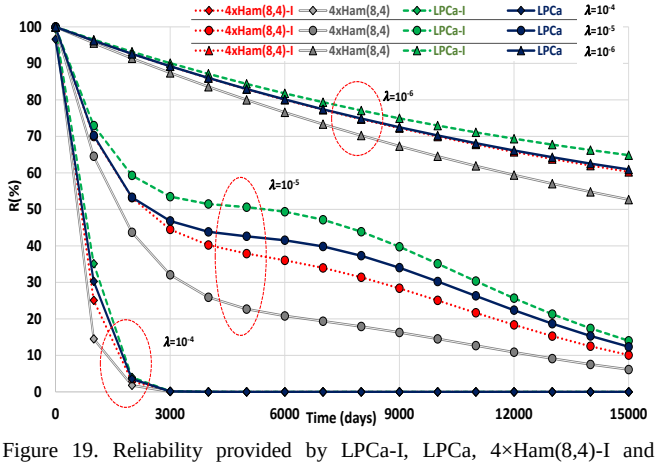


Figure 19. Reliability provided by LPCa-I, LPCa, 4×Ham(8,4)-I and 4×Ham(8,4) on die 1. The reliability regards three values of λ (bit faults per day). The horizontal axis is the time in days, and the vertical axis is the reliability in %.

The λ parameter indicates the error incidence rate in memory. For example, λ=10^{-4} indicates one bitflip every 10,000 days; consequently, one bitflip every 208 days in the 48-bit LPC. As $R(t)$ is computed cumulatively, Figure 19 shows that in 3000 days, for instance, the memory would have 14 bitflips, which would lead to reliability close to zero for all evaluated ECCs.

For all values of λ, Figure 19 displays that LPCa-I, followed by LPCa, is the most reliable ECC throughout the period. For instance, with λ=10^{-5}, LPCa-I reaches a rate of 99.96%, 70.26% and 53.33% at days 1, 1000, and 2000, respectively. For these same days, 4×Ham(8,4)-I has 99.96%, 69.99%, and 53.22%, while Ham(8,4) has 99.95%, 64.57%, and 43.74%. Until day 2000, the values of LPCa and 4×Ham(8,4)-I are very close to each other, not exceeding a difference greater than 0.4%. After this day, the difference reaches almost 20%. As of day 7000, 4×Ham(8,4) has a reliability of less than 20%. Finally, at day 15000, LPCa, 4×Ham(8,4)-I and 4×Ham(8,4) have reliabilities of 12.38%, 10.10%, and 6.13%, respectively.

7.4 Final Remarks

Experimental results show that there is a high variation in the incidence of errors between the dies of a 3D memory. This feature makes room for the research of heterogeneous ECC models, whose correction capacity is higher in the upper and lower dies, significantly reducing in the intermediate dies. This research involves exploring the relationship between the number of information bits versus redundancy bits and, consequently, working with various requirements, such as minimizing area consumption and power dissipation.

8 Conclusions

This work proposes LPC - a product-type ECC that uses Hamming and parity codes on both rows and columns. The experimental results demonstrate that this code implemented in two lightweight decoding versions, in interleaved (LPCa-I) and non-interleaved (LPCa) algorithms, has high error correction capability enabling its usage in space application memories.

The validation of the proposed ECC and the correction technique applied by the LPCa-based decoding algorithms were performed using a set of simulations varying the error severity level and test cases, producing different numbers of errors on the dies of the 3D memory. The results were analyzed and discussed comparing LPCa-I and LPCa with two other ECCs based on the Hamming codes (4×Ham(8,4)-I and 4×Ham(8,4)), equally designed for use in space application memories.

For each ECC, 16 combinations of error severity levels and test cases were simulated 30,000 to achieve a high confidence degree for error correction rate and reliability results. When considering the higher error severity level, the error correction capacities of LPCa-I is only 1.3 times higher than 4×Ham(8,4)-I; and, in the same situation, the LPCa-I error correction capacity is more than 2 times higher when compared to 4×Ham(8,4). When considering all dies and all error severity levels, the error correction

rate of LPCa-I, on average, is 7.1% and 22.1% higher than 4×Ham(8,4)-I and 4×Ham(8,4), respectively. The higher error correction results of LPCa-based algorithms are due to (i) its matrix format and (ii) the existence of two syndromes for each row and column (Hamming check and parity bit), and (iii) a novel technique that improves Hamming capacity by applying bit inversions in double error occurrences.

Finally, the experimental results show that the die stacking characteristics of 3D memories provide radiation protection, reducing the incidence of errors in lower layers; Additionally, the heat dissipated by the active logic below the lower die also generates errors, but these are little propagated to the upper dies. This 3D error incidence model makes room for researching ECCs with different error correction capabilities applied to each layer of 3D memory.

REFERENCES

[1] A. Bagatin, S. Gerardin, A. Paccagnella, S. Beltrami, E. Camerlenghi, M. Bertuccio, "Effects of Heavy-Ion Irradiation on Vertical 3-D NAND Flash Memories", IEEE Transactions on Nuclear Science, vol. 65, n. 1, pp. 318-325, Jan. 2018.

[2] W. Zhang, T. Li, "Microarchitecture soft error vulnerability characterization and mitigation under 3D integration technology", Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 435-446, 2009.

[3] G. Brucker, B. Parson, "Radiation Test, Simulation of CMOS/SOS/Si-Gate ALU, ROM Devices", IEEE Transactions on Nuclear Science, vol. 23, n. 6, pp. 1720-1727, Dec. 1976.

[4] R. Kjar, B. Peterson, J. Blandford, "Radiation Hardened 64-bit CMOS/SOS RAM", IEEE Transactions on Nuclear Science, vol. 23, n. 6, pp. 1728-1731, Dec. 1976.

[5] G. Brucker, "Characteristics of CMOS/Bulk, SOS Memories in a Transient Environment", IEEE Transactions on Nuclear Science, vol. 24, n. 6, pp. 2209-2212, Dec. 1977.

[6] T. Ellis, "Radiation Effects Characterization of the SBP9900A 16-bit Microprocessor", IEEE Transactions on Nuclear Science, vol. 26, n. 6, pp. 4735-4739, Dec. 1979.

[7] T. Kato, T. Yamazaki, N. Saito, H. Matsuyama, "Neutron-Induced Multiple Cell Upsets in 20-nm Bulk SRAM: Angular Sensitivity and Impact of Multiwall Potential Perturbation", IEEE Transactions on Nuclear Science, vol. 66, n. 7, pp. 1381-1389, Jul. 2019.

[8] R. Kjar, B. Peterson, J. Blandford, "Radiation Hardened 64-bit CMOS/SOS RAM", IEEE Transactions on Nuclear Science, vol. 23, n. 6, pp. 1728-1731, Dec. 1976.

[9] G. Brucker, "Characteristics of CMOS/Bulk, SOS Memories in a Transient Environment", IEEE Transactions on Nuclear Science, vol. 24, n. 6, pp. 2209-2212, Dec. 1977.

[10] M. Sato, R. Egawa, H. Takizawa, H. Kobayashi, "On-chip checkpoint with 3D-stacked memories", Proceedings of the International 3D Systems Integrations Conference (3DIC), pp. 1-6, 2015.

[11] K. Puttaswamy, G. Loh, "3D-Integrated SRAM Components for High-Performance Microprocessors", IEEE Transactions on Computers, vol. 58, n. 10, pp. 1369-1381, Oct. 2009.

[12] H. Han, J. Yang, "Asymmetric ECC organization in 3D-memory via spare column utilization", Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), pp. 13-16, 2015.

[13] H. Han, J. Chung and J. Yang, "READ: Reliability Enhancement in 3D-Memory Exploiting Asymmetric SER Distribution", IEEE Transactions on Computers, vol. 67, n. 8, pp. 1193-1201, Aug. 2018.

[14] M. Bagatin, S. Gerardin, A. Paccagnella, S. Beltrami, C. Cazzaniga, C. Frost, "Atmospheric Neutron Soft Errors in 3-D NAND Flash Memories", IEEE Transactions on Nuclear Science, vol. 66, n. 7, pp. 1361-1367, Jul. 2019.

[15] M. Bagatin, S. Gerardin, A. Paccagnella, S. Beltrami, A. Costantino, M. Muschitiello, A. Zadeh, V. Cavrois, "Total Ionizing Dose Effects in 3D NAND Flash Memories", IEEE Transactions on Nuclear Science, vol. 66, n. 1, pp. 48-53, Jan. 2019.

[16] H. Yoon, J. Park, J. Kim, "An Efficient Error Detection Technique for 3D Bit-Partitioned SRAM Devices", Journal of Semiconductor Technology and Science, vol. 15, n. 5, pp. 445-454, Oct. 2015.

[17] L. Chang, Y. Huang, J. Li, "Area and reliability efficient ECC scheme for 3D RAMs", Proceedings of VLSI Design, Automation and Test (VLSI-DAT), pp. 1-4, 2012.

[18] J. Kim, J. Yang, "DRIS-3: Deep Neural Network Reliability Improvement Scheme in 3D Die-Stacked Memory based on Fault Analysis", Proceedings of the ACM/IEEE Design Automation Conference (DAC), pp. 1-6, 2019.

[19] R. Hamming, "Error detecting and error correcting codes", Bell System Technical Journal, vol. 29, n. 2, pp. 147-160, Apr. 1950.

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

[21] R. Tocci, N. Widmer, G. Moss, "Digital Systems – Principles, Applications", 10th ed., Ed. Pearson, pp. 41-46, 2007.

[22] F. Macwilliams, N. Sloane, "The Theory of Error-Correcting Codes", 3rd ed., vol. 16, Ed. North-Holland, pp. 568-570, 1977.

[23] T. Moon, "Error Correcting Code – Mathematical Methods, Algorithms", 1st ed., vol. 1, Ed. Wiley, pp. 430-432, 2005.

[24] R. Zaragoza, "The Art of Error Correcting Coding", 2nd ed., Ed. Wiley, West Sussex, England: Wiley, pp. 170-201, 2006.

[25] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, J. Hoe, "Multi-bit Error Tolerant Caches using Two-Dimensional Error Coding", Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 197-209, 2007.

[26] M. Manoochehri, M. Annavaram, M. Dubois, "CPPC: Correctable Parity Protected Cache", Proceedings of the Annual International Symposium on Computer Architecture (ISCA), pp. 223-234, 2011.

[27] P. Nair, D. Roberts, M. Qureshi, "Citadel: Efficiently Protecting Stacked Memory from Large Granularity Failures", Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 51-62, 2014.

[28] X. Jian, V. Sridharan, R. Kumar, "Parity Helix: Efficient protection for single-dimensional faults in multi-dimensional memory systems", Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 555-567, 2016.

[29] P. Nair, B. Asgari, M. Qureshi. "SuDoku: Tolerating High-Rate of Transient Failures for Enabling Scalable STTRAM", Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 388-400, 2019.

[30] M. Kishani, H. Zarandi, H. Pedram, A. Tajary, M. Raji, B. Ghavami, "HVD: horizontal-vertical-diagonal error detecting and correcting code to protect against with soft errors", Design Automation for Embedded Systems, vol. 15, pp. 289-310, May, 2011.

[31] J. Gracia-Morán, L. Saiz-Adalid, D. Gil-Tomás, P. Gil-Vicente. "Improving Error Correction Codes for Multiple-Cell Upsets in Space Applications", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, n. 10, pp. 2132-2142, Oct. 2018.

[32] F. Alzahrani, T. Chen, "On-chip triple-error correction and quadruple-error detection ECC structure for ultra-large, single-chip memories", Computers and Electrical Engineering, v. 26, n. 5, pp. 317-335, Jan. 2000.

[33] F. Silva, W. Freitas, J. Silveira, C. Marcon, F. Vargas, "Extended Matrix Region Selection Code: An ECC for adjacent Multiple Cell Upset in memory arrays", Microelectronics Reliability, vol. 106, pp 1-9, Jan. 2020.

[34] P. Nair, V. Sridharan, M. Qureshi, "XED: Exposing On-Die Error Detection Information for Strong Memory Reliability", Proceedings of the ACM/IEEE Annual International Symposium on Computer Architecture (ISCA), pp. 341-353, 2016.

[35] M. Patel, J. Kim, H. Hassan, O. Mutlu, "Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices", Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 13-25, 2019.

[36] D. Radaelli, H. Puchner, S. Wong, S. Daniel, "Investigation of multi-bit upsets in a 150nm technology SRAM device", IEEE Transactions on Nuclear Science, vol. 52, n. 1, pp. 2433-2437, Dec. 2005.

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

[38] A. Neale, M. Jonkman, M. Sachdev, "Adjacent-MBU-Tolerant SEC-DED-TAEC-yAED Codes for Embedded SRAMs", IEEE Transactions on Circuits and Systems-II: Express Briefs, vol. 62, n. 4, pp. 387-391, Apr. 2015.

[39] Jiaqiang Li, P. Reviriego, L. Xiao, C. Argyrides, Jie Li, "Extending 3-bit Burst Error-Correction Codes with Quadruple Adjacent Error Correction", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, n. 2, pp. 221-229, Feb. 2018.

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

David C. C. Freitas received the bachelor's degree in mechatronics engineering from Federal Institute of Ceará (IFCE), Fortaleza, Ceará, Brazil in 2012. He received the master's degree in electrical engineering from the Federal University of Paraíba (UFPB), Campina Grande, Paraíba, Brazil, in 2016. He is a PhD student at Federal University of Ceará (UFC) in Teleinformatics Engineering Department since 2018. Since 2017, he has been with the Federal Institute of Ceará, where he is currently Professor. His current research interests are in electronic circuits, Petri Nets, Embedded Systems Hardware, Lead Acid Batteries, Digital Circuit Design (ASIC and FPGA), Fault Tolerant Systems and Error Correction Codes.

David F. M. Mota is an engineer and master in Teleinformatics Engineering from the Federal University of Ceará (UFC). He was a developer in electronic engineering at Instituto Atlântico for 7 years. Was a research fellow at UFC in the Embedded Systems area. Worked at the Computer Systems Engineering Laboratory (LESC) as a Hardware Engineer. At the Orion Institute of Science and Technology he also served as a hardware engineer. He worked at Schneider Electric developing low power Nobreaks and implementing electronic circuits and printed circuit boards. He is currently a UFC server / engineer based at the Institute of Physical Education and Sports (IEFES) and a PhD student in the Teleinformatics Engineering postgraduate program. Has experience in Embedded Systems projects with emphasis on electronic circuit implementation and development of high complexity multi-layer printed circuit boards with focus on High Speed Design.

César Marcon received the Ph.D. degree in computer science from the Federal University of Rio Grande do Sul, Brazil, in 2005. He has been a Professor with the School of Computer Science, Pontifical Catholic University of Rio Grande do Sul (PUCRS), Brazil, since 1995. He is a senior member of Institute of Electrical and Electronics Engineers (IEEE) and member of the Brazilian Computer Society (SBC). He has more than 150 papers published in prestigious journals and conference proceedings. His research interests include embedded systems in the telecom domain, MPSoC architectures, partitioning and mapping application tasks, and fault-tolerance and real-time operating systems.

Jarbas A. N. Silveira received the Ph.D. degree in teleinformatics engineering from Federal University of Ceará (UFC) in 2015. He has been an Adjunct Professor with the Teleinformatics Department, UFC, Brazil, since 2009, where he is also with the Engineering Laboratory Computer Systems. His research interests are in the areas of embedded systems on digital circuits, computer architecture, on-chip communication architectures, fault tolerance, and real-time systems.

João C. M. Mota received the B.S. degree in physics from the Federal University of Ceará, Fortaleza, Brazil, in 1978, the M.Sc. degree in electrical engineering from the Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, in 1984, and the Ph.D. degree in electrical engineering from the State University of Campinas, Campinas, Brazil, in 1992. He is currently a Professor at the Federal University of Ceará and the Assistant Director for Interinstitutional Relationships of its Technology Center, founding member of the Brazilian Society of Telecommunications, member of the Brazilian Society of Health Informatics, adviser to the Student Branch of the Institute of Electrical and Electronics Engineers (IEEE) in FUC, member of the Signal Processing Society and IEEE Communications Society.