# Codes for Limited Magnitude Error Correction in Multilevel Cell Memories

Shanshan Liu, *Member, IEEE*, Pedro Reviriego, *Senior Member, IEEE*, and Fabrizio Lombardi, *Fellow, IEEE*

***Abstract*— Multilevel cell (MLC) memories have been advocated for increasing density at low cost in next generation memories. However, the feature of several bits in a cell reduces the distance between levels; this reduced margin makes such memories more vulnerable to defective phenomena and parameter variations, leading to an error in stored data. These errors typically are of limited magnitude, because the induced change causes the stored value to exceed only a few of the level boundaries. To protect these memories from such errors and ensure that the stored data is not corrupted, Error Correction Codes (ECCs) are commonly used. However, most existing codes have been designed to protect memories in which each cell stores a bit and thus, they are not efficient to protect MLC memories. In this paper, an efficient scheme that can correct up to magnitude-3 errors is presented and evaluated. The scheme is based by combining ECCs that are commonly used to protect traditional memories. In particular, Interleaved Parity (IP) bits and Single Error Correction and Double Adjacent Error Correction (SEC-DAEC) codes are utilized; both these codes are combined in the proposed IP-DAEC scheme to efficiently provide a strong coding function for correction, thus exceeding the capabilities of most existing coding schemes for limited magnitude errors. The SEC-DAEC code is used to detect the cell in error and correct some bits, while the IP bits identify the remaining erroneous bits in the memory cell. The use of these simple codes results in an efficient implementation of the decoder compared to existing techniques as shown by the evaluation results presented in this paper. The proposed scheme is also competitive in terms of number of parity check bits and memory redundancy. Therefore, the proposed IP-DAEC scheme is a very efficient alternative to protect and correct MLC memories from limited magnitude errors.**

***Index Terms*— Multilevel cell memories, limited magnitude errors, error correction codes, SEC-DAEC codes.**

## I. Introduction

In THE last decade, computing has been pervasively utilized in many systems and applications. This trend is accelerating with the Internet of Things (IoT) and the use of advanced mobile devices. The number of devices has dramatically grown together with requirements of large volume in storage data [1]. Demands on memory have become a very stringent requirement to support large storage for data and computational growth. Many technologies have been proposed for the next generation of memories, such as phase change (PC) memories, magneto-electric (ME) memories and memristor-based memories [2]–[5]. To achieve larger densities and departing from the design of cells for a single bit (as in traditional SRAMs or DRAMs), most emerging technologies store several bits per memory cell (e.g., octal is already available for PCMs [6]–[9]). This is accomplished by dividing the range of the physical storage parameter (such as memristance and resistance) for storing the value into several levels and assigning a bit pattern to each of them. However, a potential issue when using multiple levels is that the separation between them (i.e. the margin) is smaller and therefore in addition to process variations, deviations on the physical storage parameter can cause levels to shift (in either one or both directions), or overlap, so creating so-called limited magnitude errors between nearby levels [10], [11]. Also, a trend in multilevel cell (MLC) design is to reduce the physical size of the non-volatile (NV) memory element (for area/power dissipation considerations), often resulting in a smaller range of the resistance (as storage parameter). This however compounds the negative effects of phenomena such as drift, due to the large number of levels (as required for a high MLC memory density [7], [9]); in these cases, the magnitude of the possible errors also increases.

A common technique to protect memories is to use Error Correction Codes (ECCs) that add redundancy to the data in the form of parity bits that are then used to detect and/or correct errors [12], [13]. Traditionally, codes for protecting memories have focused on single bit error detection/correction (e.g., a single parity (SP) bit or Single Error Correction (SEC) codes), because these are the most relevant error patterns [14], [15]. In the last decade, there has been a growing interest to deal also with errors that affect adjacent bits as relevant patterns for traditional memories due to technology scaling [16]. A possible scheme to deal with adjacent errors is to use interleaving, i.e. to place bits that belong to the same word physically apart [17]. This ensures that an adjacent error affects bits of different words and thus each word suffers only a single bit error. Therefore, an SEC code can be used to correct errors. However, the use of interleaving has drawbacks, because it complicates the layout and interconnections of the memory and may also limit its aspect ratio [17]. Another approach is to use ECCs that can deal with adjacent bit errors; for example, interleaved parity (IP) bits can detect multiple

Manuscript received July 27, 2019; revised November 19, 2019; accepted December 19, 2019. Date of publication January 9, 2020; date of current version May 1, 2020. The work of P. Reviriego was supported in part by the TEXEO Project funded by the Spanish Research Plan under Grant TEC2016-80339-R and in part by the Madrid Community Research Project TAPIR-CM under Grant P2018/TCS-4496. This article was recommended by Associate Editor W. Zhao. *(Corresponding author: Shanshan Liu.)*

Shanshan Liu and Fabrizio Lombardi are with the Department of ECE, Northeastern University, Boston, MA 02115 USA (e-mail: ssliu@ece.neu.edu; lombardi@ece.neu.edu).

Pedro Reviriego is with the Departamento de Ingenieria Telematica, Universidad Carlos III de Madrid, 28903 Madrid, Spain (e-mail: revirieg@it.uc3m.es).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2019.2961847

adjacent bit errors, or codes that achieve single error correction and double adjacent error correction (SEC-DAEC) can be constructed [14], [17], [18].

Different from traditional (binary) memories in which an error on a cell affects only a single bit, the relevant bit error patterns for a multilevel cell (MLC) memory depend mostly on the mapping of levels to bits and the magnitude of the errors. For example, with a binary coding of levels to bits, even magnitude-1 errors can cause several bit errors [10], [11]. Gray coding of levels to bits can be used to ensure that magnitude-1 errors affect a single bit and thus, they can be corrected by an SEC code. However, this approach has limited benefits when dealing with larger magnitude errors. For example, when considering magnitude-3 errors, Gray coding can only ensure that errors affect at most 3 bits and thus a strong ECC is needed to correct them. Therefore, as the magnitude of errors increases, the benefits of Gray coding decrease, so making its use less attractive.

Another approach is the use of interleaving by placing bits of different words on the same cell, so that an error on a cell affects a single bit per word. However, when applied to next generation MLC memories, interleaving has additional drawbacks to the ones already discussed for traditional memories. Consider a memory with cells of $b$ bits and a word of $w$ bits that map to the first bit of each cell. A first issue is that reading a word now requires reading $w$ cells instead of $w/b$ cells. This translates to an increase in power dissipation and support circuitry complexity. The scenario is worse for writes to the memory because to avoid corrupting the data of other words stored on the remaining $b$-1 bits of the cell, we need to first read the cells, modify the relevant bit and then write back the result to the cells. This introduces a significant overhead for writes that may not be acceptable if the memories are to be used as an alternative to SRAMs. The use of interleaving would also increase the number of writes made to each cell and thus may also impact the lifetime of the memory as next generation MLC memories typically can withstand a limited number of write operations prior to deterioration and catastrophic failure.

From the previous discussion, it is evident that the use of interleaving in next generation MLC memories is problematic. Therefore, we consider the use of ECCs. Given the difference in error patterns with traditional memories, existing codes (as commonly used to protect traditional memories) are in most cases not adequate for MLC memories. For example, when dealing with single cell errors of magnitude larger than one, the SEC codes used for traditional memories are not sufficient, so stronger ECCs are needed. However, these ECCs target errors on the entire memory word rather than errors in a single cell and therefore, they protect against error patterns that are not relevant. Hence, protection of MLC memories against limited magnitude errors has been considered in recent works. In particular, asymmetric errors of magnitude-1 are targeted in [19] by noticing that the lowest bit will always be affected by a magnitude-1 error when a binary mapping is utilized. Then, an SEC Hamming code on the lowest bit is sufficient to detect the cell in error; since the magnitude and sign of the error are known (asymmetric and magnitude one), then it can be corrected with a relatively simple code. More advanced schemes that can correct symmetric limited errors have also been proposed based on the use of Orthogonal Latin Square (OLS) codes [20]; combined with a check of the truth table for all possible error patterns, they can provide additional protection for errors of any magnitude, but at an increasing cost with a large number of parity check bits. Therefore, the complexity of the required circuitry increases when the number of bits stored in the memory cells is large. Efficient Spotty codes [21] have an advantage in terms of number of parity bits. Using a Gray mapping of levels to bits, magnitude-1 errors only erroneously affect one bit, while magnitude-2 errors only affect two bits; limited magnitude errors are converted into single and double bit errors in single cells. Spotty codes that can correct double bit error patterns per cell have been proposed to deal with only single and double bit errors, leading to a low redundancy implementation. However, these schemes can only correct up to magnitude-2 errors. Some non-binary codes that correct single or multiple cell errors have also been studied [22], [23]; however, they require a large hardware overhead. In terms of limited magnitude error detection, two efficient schemes for symmetric errors in MLC memories have been proposed in [24]: the One-Bit Parity (OBP) scheme that can detect any magnitude-1 error and the Two-Bit Parity (TBP) scheme that can detect any magnitude-2 error. As all magnitude-1 (magnitude-2) errors always corrupt the lowest bit (the second lowest bits) of the memory cell, only a single parity bit (two parity bits) that covers all lowest bits (and the second lowest bits) per cell is needed by the OBP (TBP) scheme. These schemes have a significant advantage in terms of memory redundancy and decoding speed; however, they have a limitation in applications, because they are only applicable to systems in which there is an exception handler and only error detection is needed. Therefore, an efficient scheme that has a strong limited magnitude error correction capability for MLC memories with low decoding latency and overhead is needed.

In this paper, an efficient scheme to correct limited magnitude errors on multilevel cell memories is presented. The proposed scheme combines the use of a low redundancy SEC-DAEC code in the two lowest bits of the cell with IP bits to correct up to magnitude-3 symmetric errors. The SEC-DAEC code is used to locate the cell in error and correct errors on some bits. The IP is used to identify the error pattern on the remaining bits. The proposed IP-DAEC scheme provides a simple implementation that achieves a low delay with a reduced number of parity bits. The proposed scheme has been compared to existing schemes that have similar capability showing an advantage of memory redundancy, as well as a lower encoder/decoder overhead in most cases.

The rest of the paper is organized as follows. In Section II, limited magnitude errors in MLC memories and the features of the error patterns are discussed; the IP and SEC-DAEC codes are also reviewed. In Section III, the IP scheme combined with SEC-DAEC codes for symmetric magnitude-3 errors correction is proposed. Two approaches to design low redundancy SEC-DAEC codes for the proposed scheme are also presented. Section IV compares the proposed schemes with existing

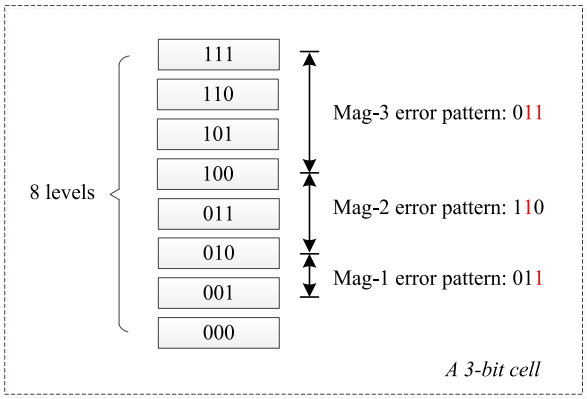



Fig. 1. A 3-bit MLC memory cell with binary mapping.

schemes in terms of memory overhead and the complexity of the encoder/decoder circuitry. Finally, Section V presents the conclusion of this paper.

## II. Preliminaries

### A. Limited Magnitude Errors in MLC Memories

Different from traditional (binary) memory cells, in MLC memories (implemented using emerging technologies), the range of the physical parameter of a cell is divided by levels based on the feature of the storage medium. For example, phase change (PC) memories rely on the reversible thermally-assisted phase transformation of the chalcogenide alloy $Ge_2Sb_2Te_5$ (GST), as occurring between the amorphous phase (with a high resistivity) and the poly-crystalline phase (with a low resistivity). The large resistance range between these two phases (usually fixed by the physical properties of the PC material) makes possible that several levels can be established by dividing the range, however, the larger the number of levels (hence the memory states) the smaller the margin between them [25]. Different states are possible and as defined by the levels, they correspond to multiple bits of a cell (i.e., $2^b$ levels correspond to $b$ bit per cell). A binary coding is commonly used for the mapping from levels to bits; for example, Figure 1 shows a 3-bit MLC memory cell with binary mapping from eight states (i.e. an octal memory cell).

However, a potential issue of MLC memories is that when a cell is programmed, or after a number of switching cycles, the margins between levels shift, or overlap as a result of the deterioration of the storage medium. In this case, the stored state changes, causing some stored bits to be corrupted too. This shift can occur in either one or both directions (e.g., unidirectional in PCM cells [25], [26] and bidirectional in memristor-based memory cells [11]); this shift increases over time or as a function of the number of switching cycles, hence causing errors of limited magnitude greater than one [11], [27], [28], but it is unlikely to cross a large number of levels. Therefore, errors in MLC memories are always of a limited magnitude. Each MLC memory cell can be affected by errors due to the effect of the resistance shift; however, prior work has shown that the error rate is low (such as in a PCM with 3-bit cells [29]), so the probability of two cell errors occurring at the same time is negligible [28]. Hence, this paper targets single cell errors, by providing correction at a reasonable cost to further reduce the error rate.

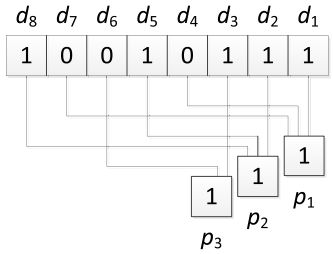



Fig. 2. $t = 3$ IP bits calculated for $k = 8$ data bits.

An important observation is that when using binary mapping from levels to bits, magnitude-1 and magnitude-2 errors always affect the lowest or the second lowest bit. This occurs because the difference in values between maginitude-1 levels is "01" while between magnitude-2 levels it is "10". For example, in Figure 1 (where "mag" denotes "magnitude"), a magnitude-1 error causes a pattern of "1" on the lowest bit, and the magnitude-2 error causes a "1" on the second lowest bit. This observation has been exploited to propose several limited magnitude error correction schemes found in the technical literature [19], [20], [22]. Similarly, a magnitude-3 error affects both the lowest and the second lowest bits because the value of the difference between these levels is "11". Again, in Figure 1, the magnitude-3 error causes a pattern of "11" on those two bits. All of these features of the errors lead us to propose in this paper a new and efficient limited magnitude error correction scheme.

### B. Interleaved Parity (IP) Scheme

Interleaved parity (IP) is widely used in conventional binary memories to detect errors that affect adjacent bits [14]. To detect $t$-bit adjacent errors on a $k$-bit data, $t$ IP bits are needed to cover all data bits; they are calculated by:

$$p_i = d_i \oplus d_{i+t} \oplus d_{i+2t} \oplus \ldots \oplus d_{i+j\cdot t} \tag{1}$$

where $\oplus$ is the addition operation in GF(2) and can be implemented by using a *xor* logic gate. $p$ represents the parity bit and $d$ the data bits. $i$ and $j$ are integers, and $1 \leq i \leq t$, $j = \lfloor k/t \rfloor$. For example, the case of $t = 3$ IP bits calculated for $k = 8$ data bits "10010111" is shown in Figure 2.

The calculation process for the IP bits (i.e., Eq. (1)) is implemented by an encoder. Then in a write operation, the calculated IP bits are stored together with the data bits into a memory word, forming an $n$-bit codeword (i.e., $n = k + t$).

In a read operation, the IP bits are recalculated first as per the read-out data bits $d'$, and then compared with the read-out IP bits $p'$ to generate the syndrome bits $S$; this is given by:

$$p''_i = d'_i \oplus d'_{i+t} \oplus d'_{i+2t} \oplus \ldots \oplus d'_{i+j\cdot t} \tag{2}$$

$$S_i = p'_i \oplus p''_i \tag{3}$$

So if the codeword is error free, the recalculated parity bits are the same as the read-out ones, such that the syndrome bits are all-zero; otherwise, any error that affects up to $t$

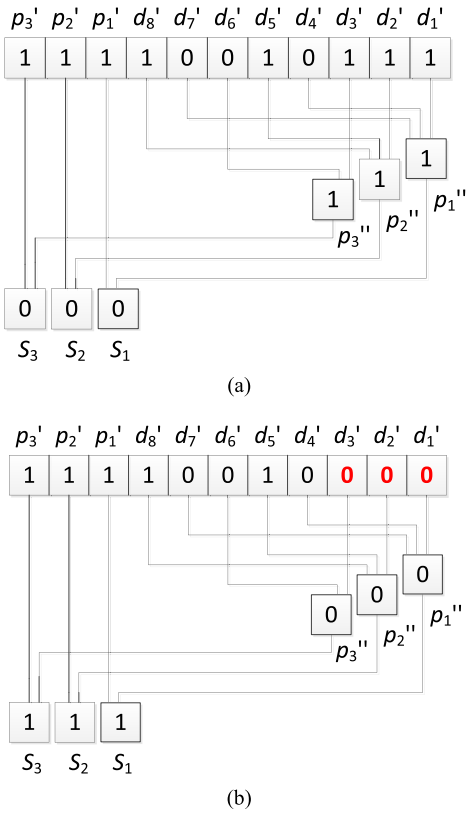



Fig. 3. Syndrome bits calculated as per the read-out codeword: (a) error free case; (b) case of erroneous bits $d_1$', $d_2$', $d_3$'.

adjacent bits can be detected by generating a non-zero $S$. Figure 3 shows the cases of error free in (a) and erroneous bits $d_1$', $d_2$', $d_3$' in (b). In the first case $S$ = "000" and in the second case $S$ = "111". Therefore, an error detection signal can be obtained if the syndrome bits are not all zero. This calculation process for the syndrome bits (i.e., Eqs. (2) and (3)) is implemented by a decoder.

As per Eqs. (1)-(3), the error pattern is reflected on the syndrome bits. Suppose the error only affects the first data bit, i.e., $d_1' = d_1 \oplus e$, where $e$ is the error pattern (for other bits, $e = 0$). In this case, as per Eqs. (1)-(3), then:

$$S_i = p_i' \oplus p_i'' = \begin{cases} d_1 \oplus d_1' = e & i = 1 \\ 0 & 1 < i \le t \end{cases} \quad (4)$$

The use of an IP-based scheme can only determine the error pattern, but it cannot locate the position of the erroneous bits and thus, it is unable to achieve error correction. For example, in the case shown in Figure 3 (b), an error affecting $d_4$', $d_5$', $d_6$' generates the same syndrome bits of "111". To correct errors, stronger ECCs are therefore required.

### C. SEC-DAEC Codes

As discussed in the introduction, SEC codes are a common technique to correct single bit errors; SEC-DAEC codes have been further developed to correct also double adjacent bit errors [17], [18]. In this case, differently from the IP scheme, a generator matrix $\mathbf{G} = [\mathbf{P}\mathbf{I}_k]$ (a parity check matrix $\mathbf{H} = [\mathbf{I}_{n-k}\mathbf{P}^{\mathbf{T}}]$), that is related to each code, is utilized to calculate the parity bits for the generation of the $n$-bit codeword in the encoder (the syndrome bits in the decoder). Therefore, the codeword $C$ is calculated by Eq. (5) (the syndrome $S$ is calculated by Eq. (6)). As operations are done in GF(2) and implemented by using *xor* logic, the number of "1" in the $\mathbf{G}$ ($\mathbf{H}$) matrix has an impact on the delay to calculate the parity bits (syndrome bits) in the encoding (decoding) process.

As per Eqs. (5) and (6), as shown as top of the next page, an error-free codeword (i.e., $C' = C$) always generates an all-zero $S$, while a codeword with any correctable error (i.e., $C' = C \oplus e$) will cause a unique $S$ (i.e., $S = e \cdot H^T$). An error locating process is implemented to find the position of the error; such process mostly determines the complexity of the decoder circuitry. Once the error is located, the erroneous bits can be corrected by simply flipping the values. The number of parity bits (i.e., $n$-$k$), which is equal to the number of columns of the $\mathbf{P}$ submatrix, can provide $2^{n-k}$-1 available non-zero syndrome patterns. It also determines the number of additional memory bits needed per word, as well as the number of *xor* logic gates in the encoder/decoder. This of course affects the overhead of the memory and the protection circuitry. The smallest value of $n$-$k$ should be designed for the best implementation in terms of overhead when the desired error correction capability of the ECCs is accomplished.

For SEC codes, each column in the $\mathbf{H}$ matrix that is equal to the syndrome for each single bit error (as per Eq. (6)), is unique; hence, the syndromes for all single bit errors are distinguishable and the error can be corrected. For SEC-DAEC codes, each column in the $\mathbf{H}$ matrix is unique to achieve single bit error correction as for SEC codes. Moreover, the *xor* of each pair of adjacent columns is also unique and different from the columns themselves; this ensures that all possible SEs and DAEs on the codeword have different syndromes and thus, they can be corrected. Therefore, a larger size of the $\mathbf{H}$ matrix (corresponding to more parity bits) is needed by the SEC-DAEC code than the SEC code in some cases when protecting data with same length. Table I shows the number of parity bits needed by the SEC and SEC-DAEC codes [12], [18].

## III. Proposed Scheme

A novel limited magnitude error correction scheme is proposed in this section. The proposed scheme combines IP with SEC-DAEC codes and can correct up to magnitude-3 errors.

### A. IP Combined With SEC-DAEC (IP-DAEC) Scheme

As discussed in section II-A, symmetric magnitude-3 errors (i.e. errors occur in both directions) will always corrupt at least one of the lowest and second lowest bits. By considering these bits in each cell as a pair of double adjacent bits, the corrupted bit(s) in the erroneous cell is transferred to a single bit or double adjacent bit error. Therefore, an SEC-DAEC code that covers the lowest and second lowest bits per cell can locate the erroneous cell, as well as correcting errors on those bits. However, magnitude-3 errors can also affect some upper bits; for example, in Figure 1 a magnitude-2 error affects the second and third lowest bits causing a

$$C = d \cdot G = (d_1, d_2, \cdots, d_k) \cdot \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1,n-k} & 1 & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & p_{2,n-k} & 0 & 1 & \cdots & 0 \\ p_{31} & p_{32} & \cdots & p_{2,n-k} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ p_{k,1} & p_{k,2} & \cdots & p_{k,n-k} & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5}$$

$$S = C' \cdot H^T = (p'_1, p'_2, \cdots, p'_{n-k}, d'_1, \cdots, d'_k) \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{11} & p_{21} & \cdots & p_{k,1} \\ 0 & 1 & \cdots & 0 & p_{12} & p_{22} & \cdots & p_{k,2} \\ 0 & 0 & \cdots & 0 & p_{13} & p_{23} & \cdots & p_{k,3} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & p_{1,n-k} & p_{2,n-k} & \cdots & p_{k,n-k} \end{bmatrix}^T \tag{6}$$

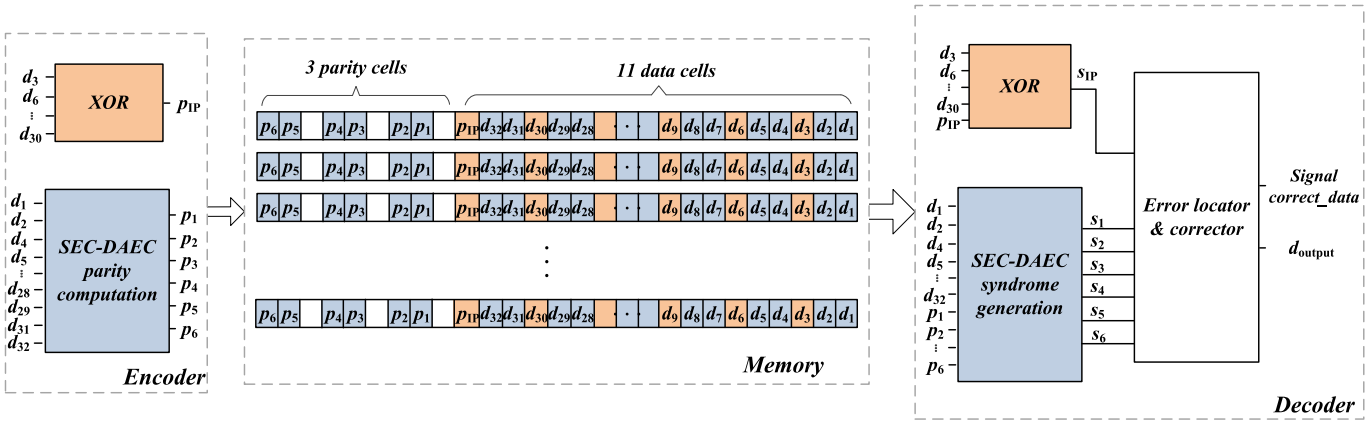



Fig. 4. Implementation of the proposed scheme to protect a 32-bit MLC memory with 3-bit cells.

pattern of "110". To handle errors on the upper bits in the erroneous cell, the IP scheme that covers those bits in each cell, is utilized to provide the error pattern for those bits in the IP syndromes. Therefore, by combining IP with SEC-DAEC codes, the proposed scheme can always guarantee correct data under symmetric magnitude-3 errors. Next, the encoder and decoder for the proposed scheme are described.

*1) Encoder Circuitry:* As the proposed scheme combines two ECC techniques (IP and SEC-DAEC), the encoder includes the following two parity computational blocks:

- The IP bits are calculated as per Eq. (1). In our case, $t$ is equal to the number of bits stored in the MLC memory cells as we target single cell errors. Then $t$-2 IP bits (i.e., $1 \leq i \leq$ t-2 in Eq. (1)) are calculated based on the upper bits of each cell.
- The SEC-DAEC parity bits are calculated as per Eq. (5). In our case, $k$ is equal to twice the number of memory cells, because in each cell two data bits are covered by the codes. Then $n$-$k$ SEC-DAEC parity bits are obtained (they can be found in Table I).

As this is 3 bit memory cell
t=3 bit
t-2= 3-2=1 bit
This 1 bit is the upper bits of each cell.

*2) Decoder Circuitry:* The proposed decoder mainly includes the following blocks: two syndrome generation blocks, an error locator and an error corrector.

- In the IP syndrome generation block, syndromes are obtained as per Eqs. (2) and (3) (again, $1 \leq i \leq t$-2).
- In the SEC-DAEC syndrome generation block, syndromes are obtained as per Eq. (6).

6 bit parity 22 bit data

$$H = \begin{bmatrix} 1\,0\,0\,0\,0\,0 & 1\,1\,1\,1\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0 \\ 0\,1\,0\,0\,0\,0 & 0\,0\,0\,1\,1\,1\,1\,1\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,0\,1\,1 \\ 0\,0\,1\,0\,0\,0 & 1\,0\,0\,1\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0 & 0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,0\,1 & 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \end{bmatrix}$$

Fig. 5. H matrix of the (28, 22) SEC-DAEC code.

- In the error locator, the SEC-DAEC syndrome bits are compared with all correctable error patterns to determine the erroneous memory cell.
- In the error corrector, an *xor* operation is implemented between the data bits read from the incorrect memory cell and the IP syndrome bits.

The implementation of the proposed limited magnitude error correction scheme for a 32-bit MLC memory with 3-bit cells is shown in Figure 4. The (28, 22) SEC-DAEC code of [18] with the **H** matrix shown in Figure 5 is utilized. In this case, one IP bit $p_{IP}$ (i.e., 3-2 = 1) and six SEC-DAEC parity bits $p_1$ to $p_6$(as per Table I) are needed. The SEC-DAEC parity bits can only be stored on the lowest and second lowest bits of each cell, such that all magnitude-3 errors can corrupt at most two adjacent bits of the SEC-DAEC codeword; else, miscorrections may occur. For example, if the first three parity bits are stored in the same cell, an error that affects all these bits, could

TABLE I

NUMBER OF PARITY BITS NEEDED BY DIFFERENT ECCS

| Data length | ECC | # parity bits | # correctable errors | # available syndromes |
|---|---|---|---|---|
| 4-bit | SEC | 4 | 8 | 15 |
| | SEC-DAEC | 4 | 15 | 15 |
| 6-bit | SEC | 4 | 10 | 15 |
| | SEC-DAEC | 5 | 19 | 31 |
| 7-bit | SEC | 4 | 11 | 15 |
| | SEC-DAEC | 5 | 23 | 31 |
| 8-bit | SEC | 4 | 12 | 15 |
| | SEC-DAEC | 5 | 25 | 31 |
| 9-bit | SEC | 4 | 13 | 15 |
| | SEC-DAEC | 5 | 27 | 31 |
| 11-bit | SEC | 5 | 16 | 31 |
| | SEC-DAEC | 5 | 31 | 31 |
| 14-bit | SEC | 5 | 19 | 31 |
| | SEC-DAEC | 6 | 39 | 63 |
| 16-bit | SEC | 5 | 21 | 31 |
| | SEC-DAEC | 6 | 43 | 63 |
| 22-bit | SEC | 5 | 27 | 31 |
| | SEC-DAEC | 6 | 55 | 63 |
| 26-bit | SEC | 6 | 32 | 63 |
| | SEC-DAEC | 7 | 65 | 127 |
| 32-bit | SEC | 6 | 38 | 63 |
| | SEC-DAEC | 7 | 77 | 127 |
| 43-bit | SEC | 6 | 49 | 63 |
| | SEC-DAEC | 7 | 99 | 127 |

S6 S5 S4 S3 S2 S1

generate a syndrome of "111000", which is the same as for the error on the 6th data bit; hence, a correct data bit would be flipped. Therefore, in Figure 4 three memory cells need to be added to each word for storing the parity bits, i.e. fourteen cells per word in total.

The flowchart of the proposed limited magnitude error correction scheme is shown in Figure 6. In a write operation, the original data is first provided as input to the encoder. The IP bits and the SEC-DAEC parity bits are calculated, and then written together with the data bits into memory, thus a codeword is stored as memory word. In a read operation, the codeword is read out from the memory and then input to the decoder. Once the decoder receives the codeword, the IP and SEC-DAEC syndrome bits are generated. If all syndrome bits are zero, the data is then directly provided as output, because it is error free; a valid signal *correct_data* that identifies a correct output, is generated (equal to "1" ("0") when the data is error free (erroneous)). Otherwise, the SEC-DAEC syndrome bits that cover the lowest and second lowest data bits per cell can be used to identify if the error is correctable. If the syndrome bits are non-zero (so a correctable error occurs on a single memory cell), then the SEC-DAEC syndrome bits are compared with that of all considered single and double adjacent bit errors in each cell to determine the position of the erroneous cell, as well as the error pattern on the lowest and second lowest bits. Once the erroneous cell is located, an *xor* operation between the IP syndrome bits and the error pattern (determined by the SEC-DAEC syndrome bits) with the data bits read from the erroneous cell is performed to correct the error. Finally, the decoder outputs the corrected data, as well as generating a valid signal *correct_data* (i.e., equal to "1") because the error has been corrected. If not all of the syndrome bits are zero but the SEC-DAEC ones are zero, an uncorrectable error is detected. In this case, the data received by the decoder is immediately provided as output; the signal *correct_data* is then given by "0", i.e. the data is incorrect. Figure 7 illustrates an example of correcting a limited magnitude 1-error in the first data cell of a 32-bit data "11...1011" stored in a memory word with 3-bit cells (as shown in Figure 4). The erroneous data cell is identified by the SEC-DAEC code and the DAE on the two lowest data bits is corrected by the SEC-DAEC code; the SE on the remaining data bit of the erroneous cell is corrected by the IP. In this case, the signal *correct_data* is given by "1".

Write Operation

Read Operation

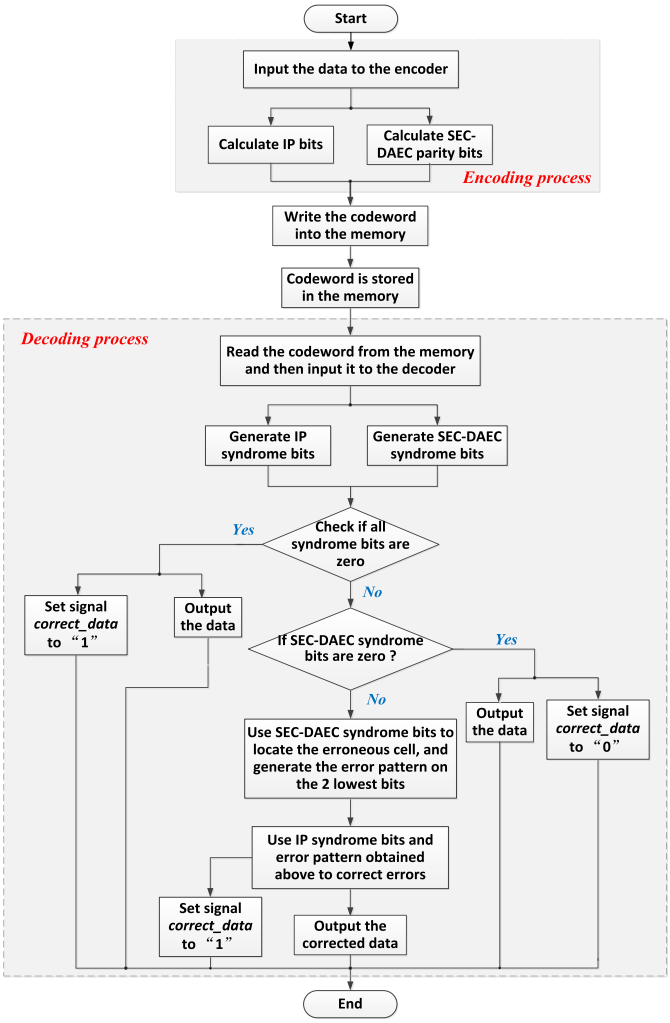



Fig. 6. Flowchart of the proposed limited magnitude error correction scheme.

As in the flowchart of Figure 6, the two schemes are combined (i.e., IP and SEC-DAEC); they are implemented in parallel for both the encoding and the decoding. Therefore, the critical path of the proposed scheme is mostly due to the SEC-DAEC codes, which have a more complex decoding process than the IP scheme. This leads to a low latency for correcting limited magnitude errors, because in this case, the SEC-DAEC

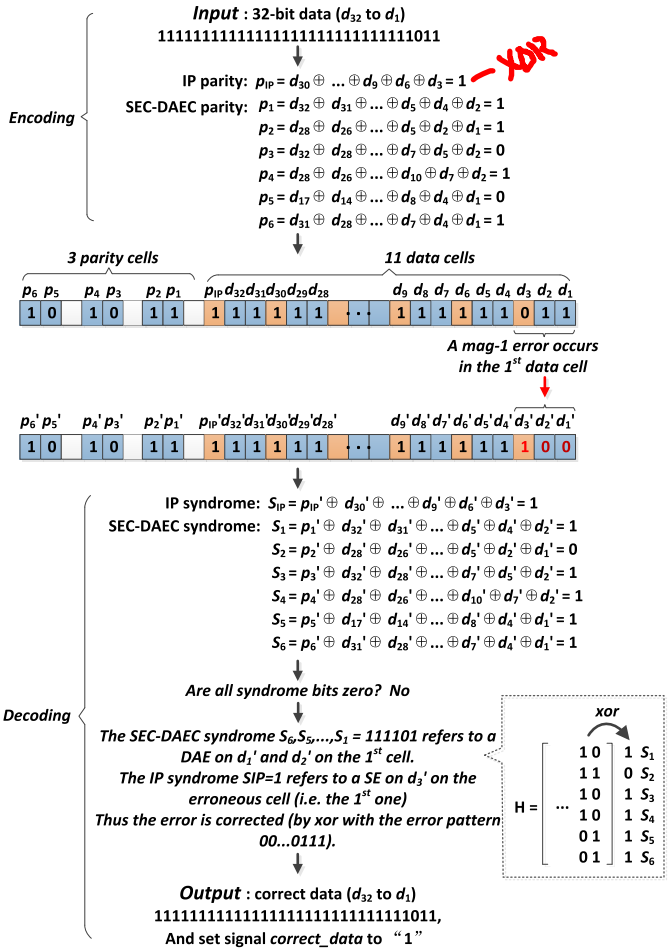



Fig. 7. An example when utilizing the proposed scheme to correct limited magnitude errors.

codes deal with a small data length (twice the number of data cells). However, the proposed scheme may incur in a large number of additional memory cells to store the parity bits, because we combine two ECCs and the number of SEC-DAEC parity bits stored in each cell is limited to 2. To address this issue, two approaches for low redundancy SEC-DAEC codes used for the proposed scheme are proposed in the next subsection.

### *B. Low Redundancy SEC-DAEC Codes*

To reduce the number of parity bits needed for the proposed scheme, two approaches are presented next for designing low redundancy SEC-DAEC codes (i.e., with a smaller number of parity bits) that can be used in the proposed scheme.

*Approach 1:* As discussed before, all magnitude-3 errors affect at least one of the lowest and second lowest bits in a memory cell, therefore any error on these bits is a correctable single or double adjacent bit error if these bits are protected by SEC-DAEC codes to locate the erroneous cell. As single cell errors are considered in this paper, double adjacent bit errors can only occur in the same memory cell (rather than the entire word). This leads to a smaller number of relevant double adjacent bit error patterns. These patterns are dependent on the number of memory cells (i.e., at most one DAE in each cell), while for the entire word, it is equal to the word length minus 1 bit (i.e., $n$-1 DAEs on the $n$-bit codeword). For example, in the 32-bit MLC memory with 3-bit cells shown in Figure 4, 28 single bit errors and 27 double adjacent bit errors can be corrected by the (28, 22) SEC-DAEC code; however, only 28 single bit errors and 14 double adjacent bit errors in all memory cells need to be taken into account. Therefore, the conventional SEC-DAEC codes provide correction for some patterns that are not relevant to our case. A strategy to obtain more efficient SEC-DAEC codes with low redundancy is to design the **H** matrices that generate distinguishable syndromes only for the single bit errors and double adjacent bit errors without sharing any bit (i.e., errors on the 1st bit and 2nd bit, 3rd bit and 4th bit, 5th bit and 6th bit and so on). In this case, the size of the **H** matrices will be smaller than conventional SEC-DAEC codes in most cases, thus the number of needed syndromes is reduced. A memory cell may also be saved with a lower decoding latency, because the error location process is simplified.

*Approach 2:* In the proposed scheme, the SEC-DAEC parity bits can only be stored on the lowest and second lowest bits of each memory cell. If there are more than two parity bits (found at the lowest and second lowest bit positions) stored in a cell, limited magnitude errors that can corrupt more than these two bits in a cell may cause an uncorrectable error pattern on the parity bits. If this uncorrectable error has the same syndrome as any single bit error or double adjacent bit error, a miscorrection will occur. This can be remedied if the **H** matrix is carefully designed. To store more SEC-DAEC parity bits in the same cell (thus reducing the number of additional cells and lowering memory redundancy), a strategy for code design is to ensure that the **H** matrices generate different syndromes for all possible errors on the parity bits (stored in the same memory cell) for all correctable errors. The syndromes for additional errors on those parity bits do not need to be different from each other, because they are only used to avoid miscorrections rather than correcting those errors (only the correctness of the data bits needs to be guaranteed). In this case, more parity bits can be stored in a single memory cell without introducing a miscorrection; moreover, the number of parity bit cells can be also reduced. A potential issue is that the number of parity bits may increase, because more syndrome patterns are considered when using this second approach. However, this is unlikely to occur, because the number of additionally considered syndromes is limited and can also be avoided by a suitable placement of the stored parity bits.

Low redundancy SEC-DAEC codes have been designed by combining the two approaches presented above and meeting the following constraints in the **H** matrices.

1) The total number of "1" is reduced (so resulting in a low complexity of the encoder/decoder circuitry);
2) The largest number of "1" per row is reduced (so resulting in a reduction of the critical path for encoding/decoding);
3) All single columns are unique and also different from any *xor* result on each pair of the 1st column and 2nd

TABLE II
ERROR CORRECTION CAPABILITIES OF DIFFERENT SCHEMES

| Scheme | Error correction capability |
|---|---|
| Hamming scheme [19] | Asymmetric magnitude 1 error |
| Spotty codes [21] | Symmetric magnitude 2 error |
| OLS scheme [20] | Symmetric magnitude 3 error |
| SSEC* RS [12] | Single symbol error |
| Proposed IP-DAEC | Symmetric magnitude 3 error |

*SSEC stands for Single Symbol Error Correction

$$H = \begin{bmatrix} 1000000011110100011101010101 \\ 0100000101011001100000111110 \\ 0010001010100011010110110001 \\ 0001000100010110011110100111 \\ 0000101010001101011111001110 \\ 0000011101101010111011010001 \end{bmatrix}$$

Fig. 8. H matrix of the low redundancy (28, 22) SEC-DAEC code used for the proposed scheme.

column, 3rd column and 4th column, 5th column and 6th column and so on (this ensures that the erroneous cell in which magnitude-3 errors occur can be located and errors on the lowest and second lowest bits of such cell can be corrected);

4) The *xor* results on any combination of the columns that relate to the parity bits stored in a same memory cell, are different from the columns/results found in constraint 3) above (so avoiding miscorrection on data bits from errors on the parity bits).

A Matlab program has been used to find the **H** matrices for low redundancy SEC-DAEC codes employed in the proposed scheme. Compared to the conventional SEC-DAEC codes given in Table I, one parity bit can be saved when the data size is 6-bit and 26-bit. In the example shown in Figure 4, the (28, 22) SEC-DAEC code with the **H** matrix shown in Figure 5 can be improved to the (28, 22) SEC-DAEC code with the **H** matrix shown in Figure 8. Even if in this case the number of parity bits may not be reduced, a memory cell per word can be saved as shown in Figure 9 (from 14 cells in Figure 5 to 13 cells in Figure 9) because more parity bits can be stored in the same cell.

### C. Advantages of the Proposed Scheme

Based on the observation that errors of limited magnitude up to 3 levels always affect at least one of the lowest and second lowest bits in a memory cell, the proposed scheme uses SEC-DAEC codes to determine which cell is in error. In this case, the error locating process (which accounts for most of the complexity of the decoder) is significantly simpler, because it is applicable to only single bit errors and a partial number of double adjacent bit errors (rather than all possible patterns) as caused by the limited magnitude errors. This leads to the following advantages of the proposed scheme:

- As the number of considered error patterns in the error location process (related to the number of distinguishable syndromes) is small, the SEC-DAEC codes used in the proposed scheme require a small number of parity bits. Even if several IP bits are additionally needed; the proposed scheme has an advantage in terms of memory redundancy when compared to existing schemes at the same error correction capability.
- The small number of considered error patterns also makes the proposed scheme efficient in terms of decoding latency compared to schemes that also implement syndrome-checking in the decoding process.

These advantages will be confirmed by the evaluation results presented in the next section.

## IV. EVALUATION

The evaluation of the proposed scheme is considered in this section; it consists of two parts: memory redundancy and the encoder/decoder overhead. Data words of 8, 16, 32, 64-bit stored in MLC memories with 3, 4, 5-bit cells are considered in the evaluation. Existing limited magnitude error correction schemes in the technical literature and the conventional Reed-Solomon (RS) codes that can correct single symbol errors are summarized in Table II; they are also evaluated for a comprehensive comparison. To show the overhead introduced by error correction over only error detection, the TBP scheme of [24] is also included in the comparison.

Overhead = Check bit/ Data word

### A. Memory Redundancy

When using ECCs to protect memories, redundant cells are added to each word to store the parity bits of the codes; this has an impact on memory overhead in terms of area and power. As each multilevel memory cell can store several bits, ECCs that have different number of parity bits, may need the same number of cells per word depending on their word length.

Table III shows the number of parity bits needed by different schemes, as well as word-length in terms of number of cells. The proposed scheme always needs more parity bits than the TBP scheme of [24] and the Hamming scheme of [19] (regardless of data size), also the Spotty codes of [21] and the SSEC RS codes [12] in several cases. In terms of word-length, (that largely affects memory redundancy), the proposed scheme only introduces one cell more than the TBP scheme (that detects magnitude-2 errors) in most cases (two cells in other cases); however, the proposed scheme can correct up to magnitude-3 errors. Compared to the Hamming scheme of [19] and the Spotty codes of [21], the proposed scheme needs the same, or one more cell, but providing a stronger error correction capability. For the same error correction capability (magnitude-3 errors), the proposed scheme has a significant advantage in terms of memory redundancy over the OLS scheme of [20] that can also correct magnitude-3 errors. This occurs because the SEC-DAEC codes used in the proposed scheme are designed with low redundancy strategies, while traditional OLS codes used in [20] usually require a significant larger number of parity bits. Compared to the SSEC RS codes,

TABLE III

NUMBER OF PARITY BITS AND WORD-LENGTH NEEDED BY DIFFERENT SCHEMES

| Data size | #bits /cell | # parity bits | | | | | | Word-length (cells) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC |
| 8 bits | 3 | 2 | 3 | 5 | 11 | 6 | **5** | 4 | 4 | 5 | 7 | 5 | **5** |
| | 4 | 2 | 3 | 6 | 9 | 8 | **6** | 3 | 3 | 4 | 5 | 4 | **4** |
| | 5 | 2 | 3 | 6 | 9 | 10 | **7** | 2 | 3 | 3 | 4 | 4 | **3** |
| 16 bits | 3 | 2 | 4 | 6 | 15 | 6 | **6** | 6 | 7 | 8 | 11 | 8 | **8** |
| | 4 | 2 | 3 | 6 | 12 | 8 | **7** | 5 | 5 | 6 | 7 | 6 | **6** |
| | 5 | 2 | 3 | 7 | 12 | 10 | **8** | 4 | 4 | 5 | 6 | 6 | **5** |
| 32 bits | 3 | 2 | 4 | 7 | 21 | 6 | **7** | 12 | 12 | 13 | 18 | 13 | **13** |
| | 4 | 2 | 4 | 7 | 18 | 8 | **8** | 9 | 9 | 10 | 13 | 10 | **10** |
| | 5 | 2 | 4 | 8 | 17 | 10 | **9** | 7 | 8 | 8 | 10 | 9 | **9** |
| 64 bits | 3 | 2 | 5 | 8 | 30 | 6 | **8** | 22 | 23 | 24 | 32 | 24 | **24** |
| | 4 | 2 | 5 | 8 | 24 | 8 | **9** | 17 | 18 | 18 | 22 | 18 | **19** |
| | 5 | 2 | 5 | 9 | 24 | 10 | **9** | 14 | 14 | 15 | 18 | 15 | **15** |

TABLE IV

SYNTHESIS RESULTS OF AREA ($um^2$) FOR ENCODERS AND DECODERS

| Data size | #bits /cell | Encoder | | | | | | Decoder | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC |
| 8 bits | 3 | 24.4 | 24.8 | 59.6 | 42.4 | 325.2 | **35.2** | 33.6 | 56.0 | 147.6 | 126.0 | 3758.4 | **139.2** |
| | 4 | 20.8 | 20.0 | 60.8 | 34.4 | 414.8 | **25.2** | 29.6 | 42.0 | 179.2 | 442.4 | 4818.4 | **116.8** |
| | 5 | 16.0 | 20.0 | 67.2 | 34.4 | 512.8 | **24.4** | 22.4 | 42.0 | 188.4 | 380.8 | 5940.4 | **110.4** |
| 16 bits | 3 | 56.0 | 56.4 | 129.6 | 95.2 | 352.4 | **80.8** | 65.6 | 116.4 | 315.6 | 251.6 | 3936.0 | **283.2** |
| | 4 | 48.8 | 45.2 | 129.6 | 73.6 | 416.0 | **66.0** | 57.6 | 83.6 | 344.0 | 854.4 | 4951.6 | **231.2** |
| | 5 | 40.8 | 45.2 | 152.4 | 64.0 | 514.8 | **62.0** | 50.4 | 83.6 | 442.0 | 1004.4 | 6094.4 | **226.4** |
| 32 bits | 3 | 124.4 | 120.8 | 288.0 | 216.8 | 356.4 | **214.0** | 133.6 | 210.0 | 676.8 | 502.8 | 4216.4 | **585.6** |
| | 4 | 105.6 | 97.6 | 312.0 | 166.4 | 440.4 | **187.2** | 114.4 | 174.4 | 685.2 | 1732.0 | 5270.4 | **526.0** |
| | 5 | 89.6 | 89.6 | 311.2 | 132.8 | 522.8 | **155.6** | 99.2 | 159.2 | 846.0 | 2126.0 | 6421.2 | **470.4** |
| 64 bits | 3 | 256.8 | 258.4 | 493.2 | 480.4 | 402.4 | **344.4** | 265.6 | 442.8 | 1216.0 | 1042.8 | 4895.6 | **952.8** |
| | 4 | 217.6 | 206.8 | 687.6 | 380.8 | 482.4 | **348.0** | 227.2 | 284.0 | 1420.4 | 3144.0 | 6056.0 | **884.8** |
| | 5 | 191.2 | 183.6 | 632.0 | 309.2 | 542.0 | **332.4** | 200.8 | 229.6 | 1741.6 | 4574.0 | 7059.6 | **845.2** |

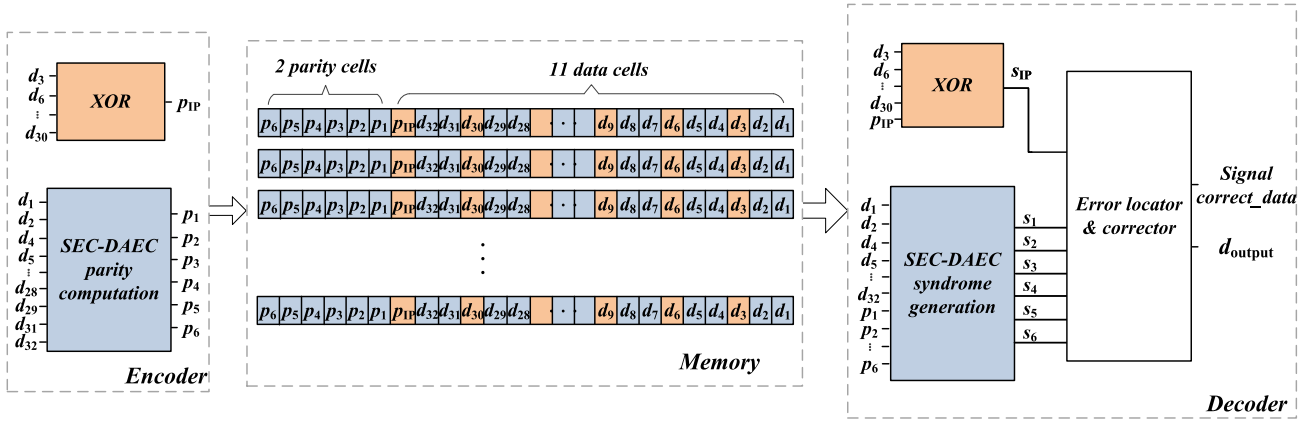



Fig. 9. Implementation of the improved design to protect a 32-bit MLC memory with 3-bit cells.

the proposed IP-DAEC scheme needs the same number of cells in nearly all cases.

Memories in most cases account for a significant fraction of the circuit area of modern digital chips, so this makes the proposed scheme very attractive for protecting MLC memories.

### B. Encoder and Decoder Overhead

To evaluate the overhead introduced by protection circuits of different schemes, encoders and decoders have been designed and implemented in HDL and mapped to a 65nm library from TSMC using the Synopsis Design Compiler. The synthesis tool has been set to area and delay optimization in the circuitry to

TABLE V

SYNTHESIS RESULTS OF DELAY (*ns*) *for Encoders and Decoders*

| Data size | #bits /cell | Encoder | | | | | | Decoder | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC |
| 8 bits | 3 | 0.20 | 0.20 | 0.31 | 0.22 | 2.61 | **0.26** | 0.25 | 0.41 | 0.58 | 0.35 | 4.55 | **0.57** |
| | 4 | 0.14 | 0.22 | 0.31 | 0.22 | 1.62 | **0.25** | 0.23 | 0.34 | 0.62 | 0.88 | 3.80 | **0.57** |
| | 5 | 0.11 | 0.22 | 0.33 | 0.22 | 1.60 | **0.25** | 0.19 | 0.34 | 0.67 | 0.82 | 4.00 | **0.56** |
| 16 bits | 3 | 0.25 | 0.26 | 0.36 | 0.26 | 4.92 | **0.32** | 0.30 | 0.47 | 0.70 | 0.40 | 7.52 | **0.68** |
| | 4 | 0.22 | 0.25 | 0.37 | 0.24 | 3.48 | **0.32** | 0.27 | 0.45 | 0.71 | 0.89 | 5.64 | **0.65** |
| | 5 | 0.20 | 0.25 | 0.39 | 0.22 | 3.16 | **0.26** | 0.25 | 0.45 | 0.76 | 0.94 | 5.88 | **0.64** |
| 32 bits | 3 | 0.33 | 0.37 | 0.45 | 0.27 | 9.68 | **0.43** | 0.36 | 0.55 | 0.82 | 0.46 | 12.22 | **0.80** |
| | 4 | 0.27 | 0.32 | 0.49 | 0.28 | 7.04 | **0.39** | 0.34 | 0.52 | 0.85 | 0.95 | 9.90 | **0.77** |
| | 5 | 0.26 | 0.31 | 0.49 | 0.27 | 5.88 | **0.36** | 0.31 | 0.51 | 0.88 | 0.98 | 8.82 | **0.73** |
| 64 bits | 3 | 0.43 | 0.42 | 0.53 | 0.32 | 19.85 | **0.51** | 0.44 | 0.68 | 0.92 | 0.49 | 22.32 | **0.90** |
| | 4 | 0.39 | 0.37 | 0.55 | 0.33 | 13.44 | **0.44** | 0.40 | 0.61 | 0.95 | 1.03 | 19.44 | **0.86** |
| | 5 | 0.38 | 0.36 | 0.55 | 0.27 | 11.05 | **0.43** | 0.38 | 0.59 | 0.97 | 1.06 | 14.25 | **0.82** |

TABLE VI

SYNTHESIS RESULTS OF POWER (*mW*) FOR ENCODERS AND DECODERS

| Data size | #bits /cell | Encoder | | | | | | Decoder | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC | TBP [24] | Hamming scheme [19] | Spotty codes [21] | OLS scheme [20] | SSEC RS [12] | Proposed IP-DAEC |
| 8 bits | 3 | 0.01 | 0.01 | 0.05 | 0.03 | 0.17 | **0.02** | 0.02 | 0.04 | 0.10 | 0.09 | 2.81 | **0.10** |
| | 4 | 0.01 | 0.01 | 0.05 | 0.02 | 0.14 | **0.02** | 0.02 | 0.03 | 0.16 | 0.39 | 2.71 | **0.09** |
| | 5 | 0.01 | 0.01 | 0.04 | 0.02 | 0.17 | **0.02** | 0.01 | 0.03 | 0.16 | 0.38 | 3.25 | **0.09** |
| 16 bits | 3 | 0.04 | 0.04 | 0.09 | 0.07 | 0.35 | **0.06** | 0.04 | 0.09 | 0.24 | 0.20 | 4.69 | **0.23** |
| | 4 | 0.03 | 0.03 | 0.09 | 0.05 | 0.28 | **0.04** | 0.04 | 0.07 | 0.33 | 1.24 | 4.19 | **0.20** |
| | 5 | 0.03 | 0.03 | 0.11 | 0.04 | 0.35 | **0.04** | 0.03 | 0.07 | 0.39 | 1.29 | 5.02 | **0.19** |
| 32 bits | 3 | 0.09 | 0.09 | 0.23 | 0.18 | 0.64 | **0.18** | 0.10 | 0.18 | 0.60 | 0.49 | 8.10 | **0.54** |
| | 4 | 0.08 | 0.07 | 0.24 | 0.12 | 0.59 | **0.17** | 0.09 | 0.16 | 0.61 | 2.39 | 7.49 | **0.48** |
| | 5 | 0.07 | 0.07 | 0.24 | 0.10 | 0.62 | **0.10** | 0.08 | 0.11 | 1.24 | 3.00 | 7.98 | **0.41** |
| 64 bits | 3 | 0.19 | 0.19 | 0.42 | 0.41 | 1.43 | **0.34** | 0.20 | 0.39 | 1.43 | 1.37 | 17.52 | **1.32** |
| | 4 | 0.18 | 0.17 | 0.61 | 0.38 | 1.25 | **0.35** | 0.20 | 0.23 | 2.20 | 4.50 | 15.70 | **1.29** |
| | 5 | 0.18 | 0.16 | 0.57 | 0.23 | 1.30 | **0.25** | 0.19 | 0.20 | 2.40 | 6.10 | 14.82 | **1.24** |

obtain the best results for these metrics. The synthesis results for area, delay and power consumption for the encoders and decoders are given in Tables IV-VI.

Consider first the overhead introduced by error correction over only error detection. From Tables IV-VI it can be seen that compared to the detection-only TBP scheme [24] the proposed scheme introduces overheads in area, delay and power for the encoder and decoder in all cases. This is expected, because the number of parity bits needed by the proposed scheme (that combines two ECCs) is larger than the TBP scheme, leading to a higher complexity of the encoder and the syndrome generation block in the decoder. Moreover, the error locating process and error correcting process also introduces additional hardware overhead compared to detection only.

Consider the schemes that can correct limited magnitude errors. Compared to [19] that uses Hamming codes to correct only asymmetric magnitude-1 errors, the proposed introduces overhead in all cases due to the additional parity bits (Table III) to account for correction and the stronger coding function. For example, the proposed scheme requires 43.3% more area, 23.1% more delay and 50.0% more power for the encoder, and 143.3% more area, 44.7% more delay and 155.6% more power for the decoder than the Hamming scheme for a 16-bit memory with 3-bit cells. Compared to the Spotty codes of [21] that correct magnitude-2 errors, the proposed scheme (that corrects magnitude-3 errors, so a stronger coding function) requires a lower hardware overhead for the encoder and decoder. For example, for a 16-bit memory with 3-bit cells, the savings are 37.7% area, 11.1% delay and 33.3% power for the encoder, and 10.3% area, 2.9% delay and 4.2% power for the decoder. For the scheme of [20] with the same correction capability (i.e. magnitude-3 errors), traditional OLS codes are used to cover three bits per cell to correct errors on those bits. Then, the corrected three bits are checked by using a truth table that lists all possible errors to identify the direction and magnitude of the error, thus the error pattern on the remaining bits is found and corrected. This decoding process is more complex than the proposed scheme for a memory with more than 3-bit per cell. For example, for a 32-bit memory with 4-bit cells, 69.6% area, 18.9% delay and 79.9% power for the decoder

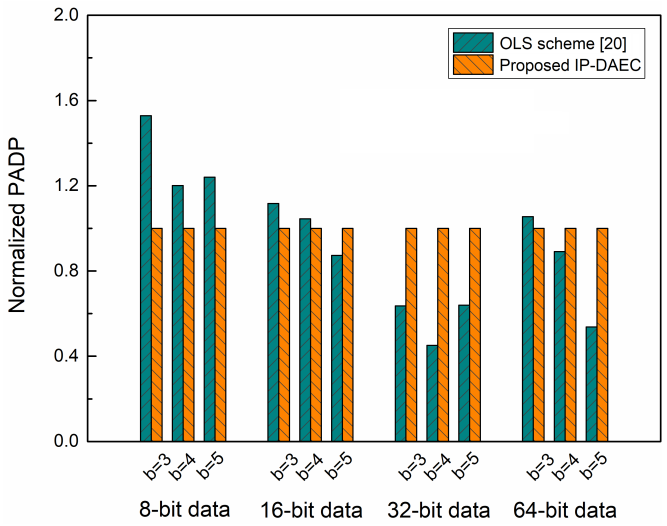



Fig. 10. Normalized PADP results for the encoders.

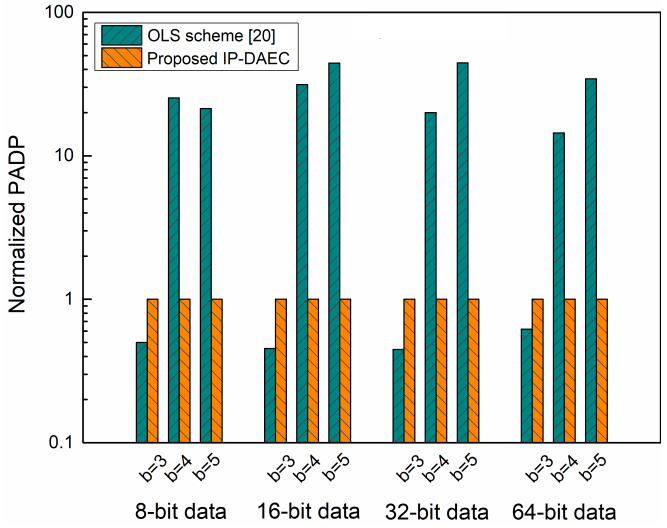



Fig. 11. Normalized PADP results for the decoders.

are saved by using the proposed scheme. When the number of bits per cell is 3, the OLS scheme of [20] incurs in a lower overhead, because in this case the OLS codes cover all bits and their majority logic decoder is fast and simple [12], [30]. For example, in a 32-bit memory with 3-bit cells, 16.5% more area, 73.9% more delay and 10.2% more power for the decoder are required by the proposed scheme. For the encoder, as the OLS codes need more parity bits, but each parity bit covers a smaller number of data bits than the proposed scheme, a larger overhead of area and power consumption is introduced by the OLS encoder in most cases, while a lower delay is achieved. For a 16-bit memory with 4-bit cells, 10.3% less area, 33.3% more delay and 20.0% less power are introduced by the proposed scheme.

Finally, when compared to the SSEC RS codes, the proposed IP-DAEC scheme incurs in a significantly lower overhead in all cases for area, delay and power dissipation.

To further show the advantage of the proposed scheme over the scheme that has the same error correction capability, the normalized combined metric of PADP (i.e., the Power Area Delay Product) results for the encoders and the decoders of the proposed scheme and the OLS scheme of [20] for different configurations are plotted in Figure 10 and Figure 11 respectively. As the decoders mostly determine the complexity of the protection circuitry (because they are the most complex block), the proposed scheme has a significant advantage (saving up to 97.7% of PADP) for MLC memories with more than 3-bits per cell.

For memory cells of a smaller size (i.e., 3-bit per cell) for which the proposed scheme has a larger PADP, the value of the PADP can be compensated by the reduction in parity bits because the parity bits account for a significant part of the overhead in large memories (they are added per word while the encoder and decoder are added to the entire memory).

## V. Conclusion

In this paper, an efficient scheme (referred to as IP-DAEC) that corrects up to symmetric magnitude-3 errors in multilevel cell (MLC) memories has been proposed; this scheme is based on the use of two simple ECCs: Interleaved Parity (IP) bits and Single Error Correction and Double Adjacent Error Correction (SEC-DAEC) codes. As all magnitude-3 errors always affect one of the lowest and second lowest bits in the cell, an SEC-DAEC code that covers these bits in each cell is used to locate the erroneous cell and correct the errors on these bits. The IP bits are used to identify the errors on the upper bits in the erroneous cell. By converting limited magnitude errors at different levels into single and double adjacent bit errors in the cell, the proposed scheme very efficiently corrects errors at a very low decoding complexity. Moreover, two approaches to design low-redundancy SEC-DAEC codes for the proposed scheme have also presented to further reduce the number of parity bits.

The advantages of the proposed IP-DAEC scheme have been verified by comparing it with existing schemes that also deal with limited magnitude errors. Evaluation results show that the proposed scheme significantly reduces memory redundancy (up to 25.0%), as well as the encoder (up to 34.6% of PADP) and decoder overhead (up to 97.7% of PADP) over existing magnitude-3 error correction schemes based on OLS codes. When compared to Spotty codes that can only correct magnitude-2 errors, the proposed scheme also performs better in terms of both memory redundancy and hardware overhead in most cases. These advantages make the proposed scheme attractive for utilization in MLC memories.

The proposed scheme can also be extended for a stronger limited magnitude error correction capability. For example, if we use codes that can correct 3-bit burst errors to cover the lowest, second lowest and third lowest bits per cell (instead of the SEC-DAEC codes in the proposed IP-DAEC scheme), up to symmetric magnitude-7 errors can be corrected by combining the codes with IP bits because those errors will always affect at least one of the bits covered. This extension is left for the future work.

## REFERENCES


[1] W. A. Bhat, "Is a data-capacity gap inevitable in big data storage?" *Computer*, vol. 51, no. 9, pp. 54–62, Sep. 2018.

[2] G. Atwood, S.-I. Chae, and S. S. Y. Shim, "Next-generation memory," *Computer*, vol. 46, no. 8, pp. 21–22, Aug. 2013.

[3] N. Papandreou *et al.*, "Multilevel phase-change memory," in *Proc. IEEE Int. Conf. Electron. Circuits Syst.*, Dec. 2010, pp. 1017–1020.

[4] G. S. Sandhu, "Emerging memories technology landscape," in *Proc. 13th Non-Volatile Memory Technol. Symp.*, Jul. 2014, pp. 1–5.

[5] X. P. Wang, Z. Y. Wang, and Y. Shen, "A high reliable design of memristor-based multilevel memory," in *Proc. IEEE 34th Chin. Control Conf.*, Sep. 2015, pp. 5615–5618.

[6] M. Stanisavljevic, H. Pozidis, A. Athmanathan, N. Papandreou, T. Mittelholzer, and E. Eleftheriou, "Demonstration of reliable triple-level-cell (TLC) phase-change memory," in *Proc. IEEE 8th Int. Memory Workshop (IMW)*, May 2016, pp. 1–4.

[7] ExtremeTech. (Aug. 2014). *Western Digital's HGST Division Creates New Phase-Change SSD that's Orders of Magnitude Faster than any NAND Flash Drive on the Market*. [Online]. Available: https://www.extrem-etech.com/extreme/187577-hitachis-new-phase-change-ssd-is-orders-of-magnitude-faster-than-any-nand-flash-drive-on-the-market

[8] (Nov. 2019). *Intel Optane Memory*. [Online]. Available: https://www.intel.com/content/www/us/en/support/products/99745/memory-and-storage/intel-optane-memory.html

[9] A. Sebastian *et al.*, "Temporal correlation detection using computational phase-change memory," *Nature Commun.*, vol. 8, no. 1, p. 1115, Oct. 2017.

[10] J. Li, B. Luan, and C. Lam, "Resistance drift in phase change memory," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2012, p. 6C-1.

[11] J. J. Yang *et al.*, "High switching endurance in $TaO_x$ memristive devices," *Appl. Phys. Lett.*, vol. 97, no. 23, Dec. 2010, Art. no. 232102.

[12] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

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

[14] Z. N. Wu, P. A. McEwan, K. K. Fitzpatrick, and J. M. Cloffi, "Interleaved parity check codes and reduced complexity detection," in *Proc. IEEE Int. Conf. Commun.*, Aug. 2002, pp. 1648–1652.

[15] M. Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. Res. Develop.*, vol. 14, no. 4, pp. 395–401, Jul. 1970.

[16] 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.

[17] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in *Proc. IEEE VLSI Test Symp.*, May 2007, pp. 349–354.

[18] Z. Ming, X. L. Yi, and L. H. Wei, "New SEC-DED-DAEC codes for multiple bit upsets mitigation in memory," in *Proc. IEEE/IFIP 19th Int. Conf. VLSI Syst.-Chip*, Aug. 2011, pp. 254–259.

[19] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multilevel flash memories," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1582–1595, Apr. 2010.

[20] A. Das and N. A. Touba, "Limited magnitude error correction using OLS codes for memories with multilevel cells," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 391–394.

[21] S. S. Liu, P. Reviriego, K. Namba, S. Pontarelli, L. Y. Xiao, and F. Lombardi, "Low redundancy double error correction spotty codes combined with gray coding for 64 data bits memories of 4-bit multilevel cells," in *Proc. 32nd IEEE Int. Symp. Defect Fault Tolerance (DFT) VLSI Nanotechnol. Syst.*, Oct. 2019, pp. 1–4.

[22] A. Das and N. A. Touba, "Efficient non-binary Hamming codes for limited magnitude errors in MLC PCMs," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2018, pp. 1–6.

[23] K. Namba and F. Lombardi, "Non-binary orthogonal latin square codes for a multilevel phase charge memory (PCM)," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 2092–2097, Aug. 2014.

[24] S. S. Liu, P. Reviriego, and F. Lombardi, "Detection of limited magnitude errors in emerging multilevel cell memories by one-bit parity (OBP) or two-bit parity (TBP)," *IEEE Trans. Emerg. Topics Comput.*, to be published.

[25] P. Junsangsri, J. Han, and F. Lombardi, "A system-level scheme for resistance drift tolerance of a multilevel phase change memory," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, Amsterdam, The Netherlands, Oct. 2014, pp. 63–68.

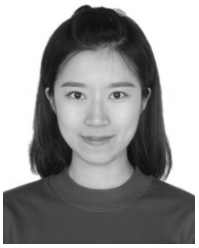
[26] P. Junsangsri and F. Lombardi, "A new comprehensive model of a phase change memory (PCM) cell," *IEEE Trans. Nanotechnol.*, vol. 13, no. 6, pp. 1213–1225, Nov. 2014.

[27] Z. Zhang, W. Xiao, N. Park, and D. J. Lilja, "Memory module-level testing and error behaviors for phase change memory," in *Proc. IEEE 30th Int. Conf. Comput. Design (ICCD)*, Sep./Oct. 2012, pp. 358–363.

[28] N. An, R. Wang, Y. Gao, H. Yang, and D. Qian, "Balancing the lifetime and storage overhead on error correction for phase change memory," *PLOS ONE*, vol. 10, no. 7, Jul. 2015, Art. no. e0131964.

[29] N. H. Seong, S. Yeo, and H. S. Lee, "Tri-level-cell phase change memory: Toward an efficient and reliable memory system," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, Jun. 2013, pp. 440–451.

[30] M. Y. Hsiao, D. C. Bossen, and R. T. Chien, "Orthogonal Latin square codes," *IBM J. Res. Develop.*, vol. 14, no. 4, pp. 390–394, Jul. 1970.

**Shanshan Liu** (Member, IEEE) received the M.Sc. and Ph.D. degrees in microelectronics and solid-state electronics from the Harbin Institute of Technology, Harbin, China, in 2012 and 2018, respectively. She is currently a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, Northeastern University, Boston, USA. Her current research interests include fault tolerant design in high-performance computer systems.

**Pedro Reviriego** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in telecommunications engineering from the Technical University of Madrid, Madrid, Spain, in 1994 and 1997, respectively. From 2007 to 2018, he was with Nebrija University. He is currently with the Universidad Carlos III de Madrid, where he has been working on high-speed packet processing and fault tolerant electronics.

**Fabrizio Lombardi** (Fellow, IEEE) received the Ph.D. degree from the University of London in 1982. He is currently the International Test Conference (ITC) Endowed Chair Professorship with Northeastern University, Boston, USA. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He is also the Editor-in-Chief of the IEEE TRANSACTIONS ON NANOTECHNOLOGY.