International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-2566

January 2017
VOLUME-1
ISSUE-1
Page:7928-32

HIGH SPEED FAULT TOLERANT ARCHITECTURE FOR MULTI LEVEL PHASE CHANGE MEMORY

1. PALURI ANUSHA, 2. R SANTHI SRI
1. PG Scholar, Dept of ECE, Prasiddha College of Engineering & Technology, Anathavaram, Amalapuram
2. Assistant Professor, of ECE, Prasiddha College of Engineering & Technology, Anathavaram, Amalapuram

ABSTRACT:
The main objective of this project is to design a non-binary OLS code as applicable to multilevel a PCM. A PCM utilizes a multilevel scheme that permits to increase the storage density using ternary, quaternary and in the near future, octal cells. The resistance drift that occurs in a multilevel PCM due to the resistive characteristics of GST may cause errors in the stored information, thus degrading data integrity. The proposed codes utilize a non-binary scheme that is capable of correcting multi-symbol errors with a parallel decoder. Memory cells have been protected from soft errors for more than a decade; due to the increase in soft error rate in logic circuits, the encoder and decoder circuitry around the memory blocks have become susceptible to soft errors as well and must also be protected. A new approach to design fault-secure encoder and decoder circuitry for memory designs is introduced. The key novel contribution of this project is identifying and defining a new class of error-correcting codes whose redundancy makes the design of fault-secure detectors (FSD) particularly simple. OLS codes satisfies a new, restricted definition for ECCs which guarantees that the ECC codeword has an appropriate redundancy structure such that it can detect multiple errors occurring in both the stored codeword in memory and the surrounding circuitries. The parity-check Matrix of an FSD-ECC has a particular structure that the decoder circuit, generated from the parity-check Matrix, is Fault-Secure. Further this is enhanced with parallel corrector logic by combining both majority gate and row density blocks. Scalable encryption Algorithm is used as extension

KEYWORDS: Fault-secure detectors (FSD), Error Correcting Codes (ECC), Fault tolerant, Phase Changed Memory (PCM), SEA.

INTRODUCTION:

LATIN SQUARE:

Lattices are regular arrangements of points in Euclidean space. They Naturally occur in many settings, like crystallography, sphere packings (stacking oranges), etc. They Have many applications in computer science and mathematics, including the solution of integer programming problems, Diophantine approximation, cryptanalysis, the design of error correcting co des for multi antenna systems, and many more. Recently, Lattices have also attracted much attention as a source of computational hardness for the design of secure cryptographic functions. In combinatorics and in experimental design, a Latin square is an $n \times n$ array filled with n different symbols, each occurring exactly once in each row and exactly once in each column. An example of a 3x3 Latin square is:

A	B	C
C	A	B
B	C	A

The name "Latin square" was inspired by mathematical papers by Leonhard Euler (1707–1783), who used Latin characters as symbols,[1] but any set of symbols can be used: in the above example, the alphabetic sequence A, B, C can be replaced by the integer sequence 1, 2, 3.Orthogonal array representation

If each entry of an $n \times n$ Latin square is written as a triple (r,c,s), where r is the row, c is the column, and s is the symbol, we obtain a set of n^2 triples called the orthogonal array representation of the square. For example, the orthogonal array representation of the following Latin square is

1	2	3
2	3	1
3	1	2

{ (1,1,1),(1,2,2),(1,3,3),(2,1,2),(2,2,3),(2,3,1),(3,1,3),(3,2,1),(3,3,2) },

where for example the triple (2,3,1) means that in row 2 and column 3 there is the symbol 1. The definition of a Latin square can be written in terms of orthogonal arrays:

A Latin square is the set of all triples (r,c,s), where 1

≤ r, c, s ≤ n, such that all ordered pairs (r,c) are distinct, all ordered pairs (r,s) are distinct, and all ordered pairs (c,s) are distinct. Memory system which can tolerate errors in any part of the system, including the storage unit, encoder and corrector circuit, using the fault-secure detector is shown below. There is a fault secure detector that can detect any combination of errors in the received code-word along with errors in the detector circuit. This fault-secure detector can verify the correctness of the encoder and corrector operation.

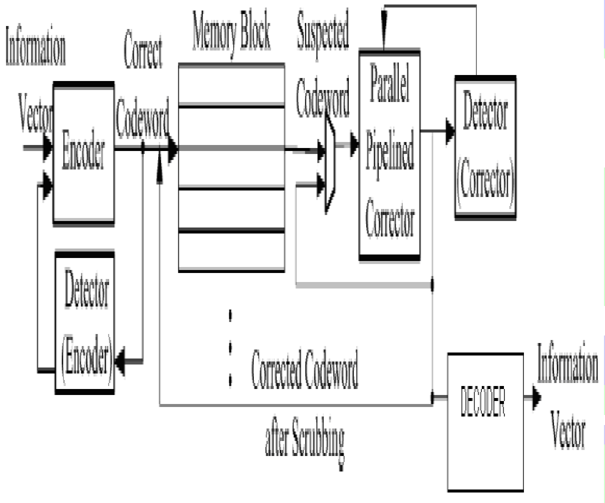


Figure Block diagram of Fault Secure Encoder and Decoder.

Design Structure:
In this section the design structure of the encoder, corrector, and detector units of our proposed fault secure encoder and decoder is provided.

Encoder
An n-bit code-word c, which encodes k-bit information vector i is generated by multiplying the k-bit information vector with k × n bit generator matrix G, i.e., c = i · G. Figure 6.8 shows the generator matrix of (15, 7) EG-OLS code. all the rows of the matrix are cyclic shifts of the first row. This cyclic code generation does not generate a systematic code and the information bits must be decoded from the encoded vector, which is not desirable for our fault-tolerant approach due to the further complication and delay that it adds to the operation.
The generator matrix of any cyclic code can be converted into systematic form (G = [I : X])

$$\begin{array}{c|ccccccccccccccc}
 & C_0 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} & C_{14} \\
\hline
i_0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
i_1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
i_2 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
i_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
i_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
i_5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
i_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}$$

Figure. The generator matrix of EG-OLS code of (15, 7) in cyclic format

$$\begin{array}{c|ccccccccccccccc}
 & C_0 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} & C_{11} & C_{12} & C_{13} & C_{14} \\
\hline
i_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
i_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
i_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
i_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
i_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
i_5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
i_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1
\end{array}$$

Figure. The generator matrix of EG-OLS code of (15, 7)
Figure Shows the systematic generator matrix to generate (15, 7) EG-OLS code. The encoded vector, which is generated by the inner product of the information vector and the generator matrix, consists of information bits followed by parity bits, where each parity bit is simply an inner product of information vector and a column of X, from G = [I : X].

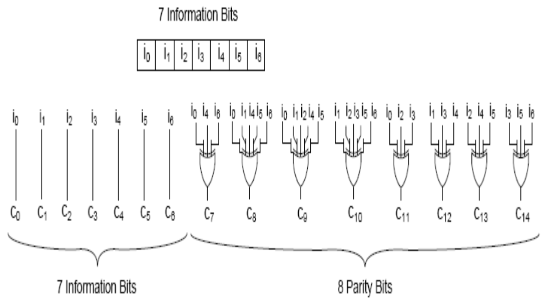


Figure . The structure of an encoder circuit for (15, 7) EG-OLS code.

Figure shows the encoder circuit to compute the parity bits of the (15, 7) EG-OLS code. In this figure i = (i0, ..., i6) is the information vector and will be copied to c0, ..., c6 bits of the encoded vector, c, and the rest of encoded vector, the parity bits, are linear sums (xor) of the information bits. If the building block is two-input gates then the encoder circuitry takes 22 twoinput xor gate. Since the systematic generator matrix of EG-OLS and PG-OLS codes does not have the standard row and column density, To compute the area of an encoder circuitry the corresponding systematic generator matrix has to be constructed. Once the systematic generator matrix is constructed the fanin size of the xor gates can be determined by the column densities of the generator matrix.

Fault Secure Detector:

The core of the detector operation is to generate the syndrome vector, which is basically implementing the following vector-matrix multiplication on the received encoded vector c and parity-check matrix H.

c H^T = S.

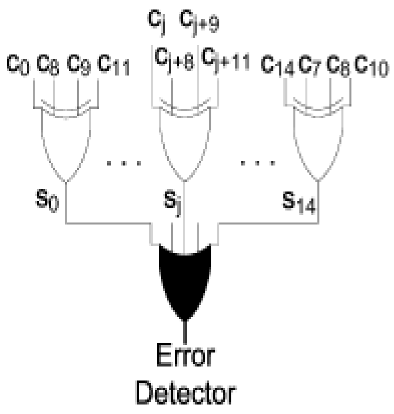


Fig. Fault-secure detector for (15, 7, 5) OLS code

Therefore each bit of the syndrome vector is the product of C with one row of the parity-check matrix. This product is a linear binary sum over digits of C. where the corresponding digit in the matrix row is 1. This binary sum is implemented with an XOR gate. Fig. shows the detector circuit for the (15, 7, 5) EG-OLS code. Since the row weight of the parity-check matrix is , to generate one digit of the syndrome vector we need a –P Input XOR gate. An error is detected if any of the syndrome bits has a nonzero value. The final error detection signal is implemented by an OR function of all the syndrome bits. The output of this -input OR gate is the error detector signal.

CORRECTOR :

One-step majority-logic correction is a fast and relatively compact error-correcting technique [15]. There is a limited class of ECCs that are one-step-majority correctable which include type-I two-dimensional EG-OLS. In this section, we present a brief review of this correcting technique. Then we show the one-step majority-logic corrector for EG-OLS codes.

1) One-Step Majority-Logic Corrector*:* One-step majority logic correction is the procedure that identifies the correct value of a each bit in the codeword directly from the received codeword; this is in contrast to the general message-passing error correction strategy (e.g., [2]) which may demand multiple iterations of error diagnosis and trial correction. Avoiding iteration makes the correction latency both small and deterministic. This technique can be implemented serially to provide a compact implementation or in parallel to minimize correction latency. This method consists of two parts: 1) generating a specific set of linear sums of the received vector bits and 2) finding the majority value of the computed linear sums. The majority value indicates the correctness of the code-bit under consideration; if the majority value is 1, the bit is inverted, otherwise it is kept unchanged.

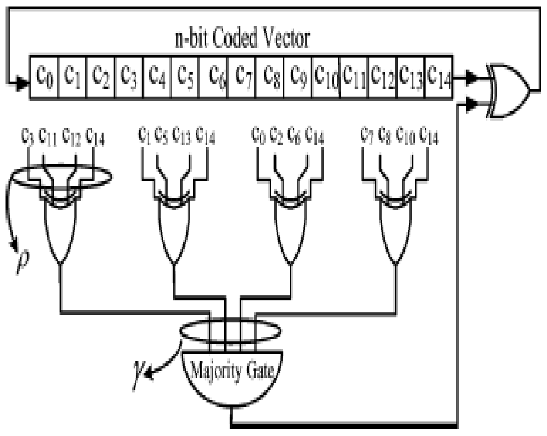

n-bit Coded Vector
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14}
C_3 C_{11} C_{12} C_{14}
C_1 C_5 C_{13} C_{14}
C_0 C_2 C_6 C_{14}
C_7 C_8 C_{10} C_{14}
ρ
γ
Majority Gate

Fig. Serial one-step majority logic corrector structure

A linear sum of the received encoded vector bits can be formed by computing the inner product of the received vector and a row of a parity-check matrix. This sum is called Parity-Check sum. A set of parity-check sums is said to be orthogonal on a given code bit if each of the parity-check sums include the code bit but no other code bit is included in more than one of these parity-check sums. If for each code bit there are j parity-check sums that are orthogonal on it, then the code is one-step majority-logic correctable up to bj/2c bit errors. In a cyclic code, a set of j parity-check sums orthogonal on a code-word bit is orthogonal on all the n code-word bits. Therefore, using one set of parity-check matrix rows orthogonal on one code bit, we can design a majority circuit that corrects all the other bits, serially.

The one-step majority logic error correction is summarized in the following procedure.

These steps correct a potential error in one code bit, e.g., cn−1.

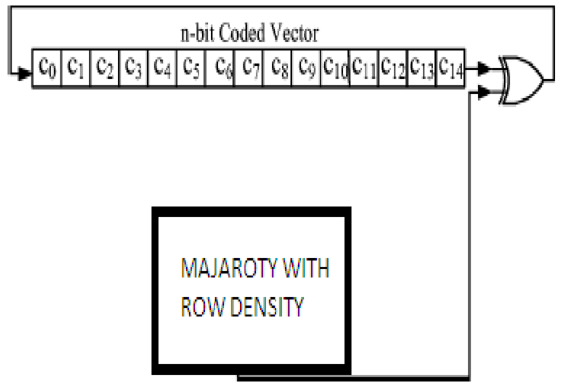
1. The j parity-check sums orthogonal on cn−1 are formed by computing the inner product of the received vector and the appropriate rows of parity-check matrix.

2. The J orthogonal check sums are fed into a majority gate. The output of the majority gate corrects the bit cn−1, by inverting the value of cn−1 if the output of majority gate is "1".

The circuit implementing a serial one-step majority logic corrector for (15, 7) EGOLS code is shown in figure is circuit generates parity-check sums with xor gates and then computes, the majority value of the parity-check sums. Since each parity-check sum is computed using a row of the parity-check matrix and the row density of EG-OLS codes are _ then each xor gate that computes the linear sum has inputs. The single xor gate on the right, corrects the code bit cn−1, using the output of the majority gate. Once the code bit cn−1 is corrected the code-word is cyclic shifted and code bit cn−2 is placed at cn−1 position and will be corrected. The whole code-word can be corrected in n rounds.

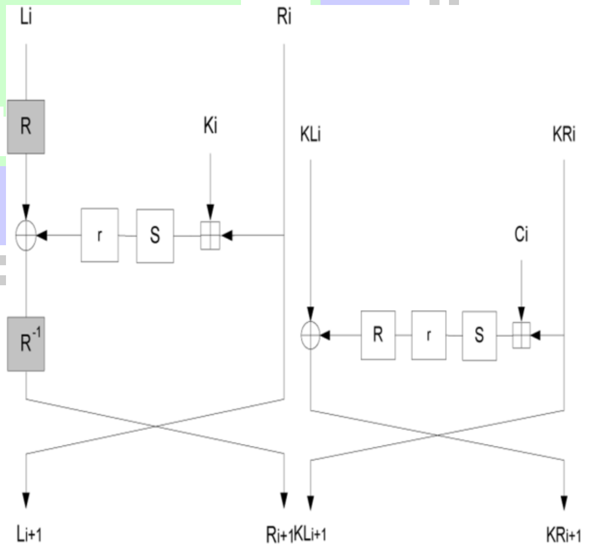
PARALLEL CORRECTOR:

Above one step serial majority logic corrector is enhanced by combining majority gate and row density to produce pipelining concept. In order to combining these two blocks, resource utilization technique is used in this concept,


n-bit Coded Vector
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14
MAJAROTY WITH ROW DENSITY

SEA:

Most present symmetric encryption algorithms result from a trade-off between implementation cost and resulting performances. In addition, they generally aim to be implemented efficiently on a large variety of platforms. In this paper, we take an opposite approach and consider a context where we have very limited processing resources and throughput requirements. For this purpose, we propose low-cost encryption routines (i.e. with small code size and memory) targeted for processors with a limited instruction set (i.e. AND, OR, XOR gates, word rotation and modular addition). The proposed design is parametric in the text, key and processor size, allows efficient combination of encryption/decryption, "on-the-fly" key derivation and its security against a number of recent cryptanalytic techniques is discussed. Target applications for such routines include any context requiring low-cost encryption and/or authentication.



Due to its simplicity constraints, SEAn;b is based on a limited number of elementary operations (selected for their availability in any processing device) denoted as follows: (1) bitwise XOR ©, (2) substitution box S, (3) word (left) rotation R and inverse word rotation R^{-1}, (4) bit rotation r, (5) addition mod 2b ¢.These operations are formally defined as follows:

1. Bitwise XOR ©: The bitwise XOR is defined on n 2 -bit vectors:

$\oplus : Z_2^{n\backslash 2} \times Z_2^{n\backslash 2 \rightarrow} Z_2^{n\backslash 2} : x, \vec{y} z = x \oplus y, \rightarrow z(i) = x(i) \oplus y(i); 0 \le i \le n/2-1$

2. Substitution box S: SEAn,b uses the following 3-bit substitution table

ST: = {0, 5, 6, 7, 4, 3, 1, 2}

in C-like notation. For efficiency purposes, it is applied bitwise to any set of three words of data using the following recursive definition:

$S: Z_{2b}^{nb \rightarrow} Z_{2b}^{nb} : x \rightarrow x = S(x) \rightarrow$

$x_{3i} = (x_{3i+2} \wedge x_{3i+1}) \oplus x_{3i}$,
$x_{3i+1} = (x_{3i+2} \wedge x_{3i}) \oplus x_{3i+1}$,
$x3i+2 = (x3i \,_{v}\, x3i+1) \oplus x_{3i+2}$,

Where ^ and $_{v}$ respectively represent the bitwise AND and OR.

3. Word rotation *R*: The word rotation is defined on *nb*-word vectors

$R: Z_{2b}^{nb \rightarrow} Z_{2b}^{nb} : x \rightarrow y = R(x) \Leftrightarrow y_{i+1} = x_i, \quad 0 \le i \le n_b-2$

$Yo = x_{nb}-1$

4. Bit rotation r: The bit rotation is defined on n_b-word vectors

$r: Z_{2b}^{nb \rightarrow} Z_{2b}^{nb} : x \rightarrow y = r(x) \Leftrightarrow \quad y_{3i} = x_{3i} >>> 1$,

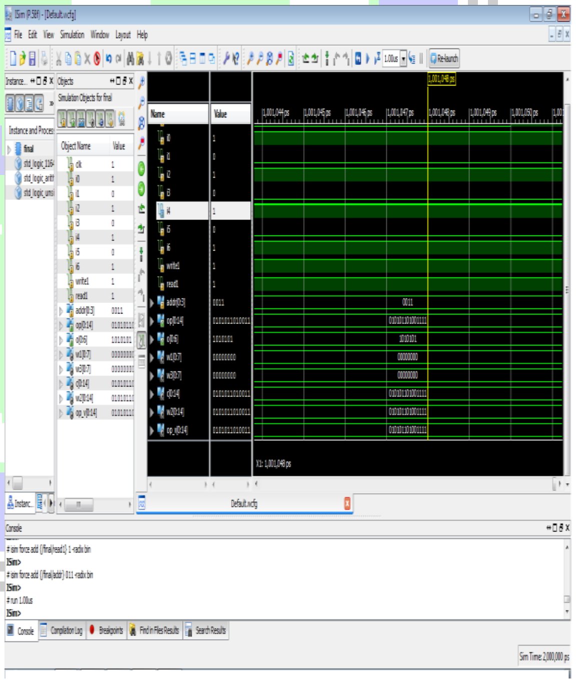
$y_{3i+1} = x_{3i+1}$,

$y3i+2 = x3i+2 <<< 1, 0 \le i \le n_{b/3}-1$

4. Addition mod2b ⊞: The mod 2b addition is defined on n_b-word vectors

$\boxplus : Z_{2b}^{nb \rightarrow} Z_{2b}^{nb} : x, y \rightarrow z = x \boxplus y, \Leftrightarrow zi = xi \boxplus yi, , \quad 0 \le i \le n_b-1$

RESULT:



```
# isim force add {/final/read1} 1 -radix bin
ISim>
# isim force add {/final/addr} 011 -radix bin
ISim>
# run 1.00us
ISim>
```

CONCLUSION:

In this report, a fully fault-tolerant memory system that is capable of tolerating errors not only in the memory bits but also in the supporting logic including the ECC encoder and decoder. OLS codes are proved as part of a new subset of ECCs that have FSDs. Using these FSDs a fault-tolerant encoder and corrector is designed with parallelism providence is also main criteria in communication systems. Here, in this concept OLS is designed with parallel correction algorithm. Finally, an enhanced secured and fault tolerant memory is designed and simulated in Xilinx ise14.5 version.

REFERENCES:

[1] N. Papandreou, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, H. Pozidis, and E. Eleftheriou, "Multilevel phase-change memory," in Proc. IEEE Int. Conf. Electron. Circuits Syst., 2010, pp. 1017–1020.

[2] X. Q. Wei, L. P. Shi, R. Walia, T. C. Chong, R. Zhao, X. S. Miao, and B. S. Quek, "HSPICE macromodel of PCRAM for binary and multilevel storage," IEEE Trans. Electron. Dev., vol. 53, no. 1, pp. 56–62, Jan. 2006.

[3] R. A. Cobley and C. D. Wright, "Parameterized SPICE model for a phasechange RAM device," IEEE Trans. Electron. Dev., vol. 53, no. 1, pp. 112–118, Jan. 2006.

[4] D. Ielmini, A. L. Lacaita, and D. Mantegazza, "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories," IEEE Trans. Electron. Dev., vol. 54, no. 2, pp. 308–315, Feb. 2007.

[5] S. Kim, B. Lee, M. Asheghi, F. Hurkx, J. P. Reifenberg, K. E. Goodson, and H. S. P. Wong, "Resistance and threshold switching voltage drift behaviour in phase-change memory and their temperature dependence at microsecond time scales studied using a micro-thermal stage," IEEE Trans. Electron. Dev., vol. 58, no. 3, pp. 584–592, Mar. 2011.

[6] I. V. Karpov, M. Mitra, D. Kau, G. Spadini, Y. A. Kryukov, and V. G. Karpov, "Fundamental drift of parameters in chalcogenide phase change memory," J. Appl. Phys., vol. 102, no. 12, pp. 124503, Dec. 2007.

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

[8] R. Datta and N. A. Touba, "Designing a fast and adaptive error correction scheme for increasing the lifetime of phase change memories," in Proc. IEEE VLSI Test Symp., 2011, pp. 134–139.

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

[10] B. Rajendran, R. W. Cheek, L. A. Lastras, M. M. Franceschini, M. J. Breitwisch, A. G. Schrott, J. Li, R. K. Montoye, L. Chang, and C. Lam, "Demonstration of CAM and TCAM using phase change devices," in Proc. IEEE Int. Memory Workshop, 2011, pp. 1–4.

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

[12] P. Reviriego, C. Argyrides, J. A. Maestro, and D. K. Pradhan, "Improving memory reliability against soft errors using block parity," IEEE Trans. Nucl. Sci., vol. 58, no. 3, pp. 981–986, Jun. 2011.

[13] M. Boniardi, D. Ielmini, S. Lavizzari, A. L. Lacaita, A. Redaelli, and A. Pirovano, "Statistical and scaling behavior of structural relaxation effects in phase-change memory (PCM) devices," in Proc. IEEE Int. Rel. Phys. Symp., 2009, pp. 122–127.