Non-Binary Orthogonal Latin Square Codes for a Multilevel Phase Charge Memory (PCM)

Kazuteru Namba, *Member, IEEE* and
Fabrizio Lombardi, *Fellow, IEEE*

Abstract—This manuscript proposes non-binary orthogonal Latin square (OLS) codes that are amenable to a multilevel phase change memory (PCM). This is based on the property that the proposed (*n* symbols, *k* symbols) *t*-symbol error correcting code uses the same H matrix as an (*n* bits, *k* bits) binary *t*-bit error correcting OLS code. The new codes are shown to have a shorter check bit length and better probability in encoding/decoding than conventional binary OLS codes. Extensive results are provided for assessment and comparison. The proposed codes are also shown to be always better than the matrix codes, i.e. independently of the metric and the parameters employed in the comparison.

Index Terms—Error correcting code (ECC), phase change memory (PCM), orthogonal Latin square (OLS) codes, multi-symbol error correcting code, parallel decoder

1 Introduction

The phase change memory (PCM) has emerged in recent years as one of the most promising technologies for future non-volatile solid-state memories with significant implications on the entire storage hierarchy [1]. PCM has attracted considerable attention due to its low latency, good endurance, long retention and high scalability compared to other non-volatile memories. PCM relies on the reversible thermally-assisted phase transformation of the chalcogenide alloy $Ge_2Sb_2Te_5$, (GST), as occurring between two structurally different phases of electrical properties: the amorphous phase with a high resistivity and the poly-crystalline phase with a low resistivity [1]. These two phases are usually referred to as the RESET and SET states, respectively [1], [2], [3]. There is a *large resistance margin* between the amorphous and the crystalline phases, so a PCM can store multiple bits of information in a single cell; this is accomplished through a *multilevel* storage implementation based upon incomplete phase transitions [1]. Advantages such as increased storage density and hence lower cost are of primary importance for the successful development of multilevel memory systems using PCM [4]. However, the resistance of a phase change material such as GTS tends to drift over time [4], [5]. The change in resistance severely degrades the margin between adjacent levels, leading to a serious data integrity challenge [6].

For PCM, the occurrence of the drift requires both compensation techniques in the management of the resistance range of the cells and efficient error correcting codes (ECC) [7], [8]. [8] has proposed error correction using orthogonal Latin square (OLS) codes; the OLS code provides both multiple-bit error correction and high-speed decoding [9]. However, the OLS code is a binary code and thus, it targets only bit-errors. Advances in PCM technology have already made possible the design and commercialization of quaternary cells [1], [10]; it is envisioned that in the next few years, an octal cell could be ready available as multilevel PCM. It is well known that to control non-binary data with binary codes is not very efficient. Hence, the following three features are highly desirable for PCM storage: 1) non-binary, 2) multiple-error correction and 3) parallel decoding.

The objective of this manuscript is to propose non-binary OLS codes that are amenable to a multilevel PCM addressing the above three features. The new codes are shown to have a shorter check bit length and better probability in encoding/decoding than the conventional binary OLS codes of [8]. In addition, the proposed non-binary OLS codes provide better parallel implementations for the encoder and decoder circuits in terms of area, power consumption and delay. Extensive results are provided for assessment and comparison. The proposed codes are also shown to be always better than the matrix codes of [11], [12], i.e. independently of the metric and the parameters employed in the comparison.

- K. Namba is with the Graduate School of Advanced Integration Science, Chiba University, Chiba, Japan. E-mail: namba@ieee.org.
- F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: lombardi@ece.neu.edu.

Manuscript received 18 July 2013; revised 14 Mar. 2014; accepted 16 June 2014. Date of publication 6 Aug. 2014; date of current version 10 June 2015.
Recommended for acceptance by A. Nannarelli.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2014.2346182

2 Review

This section provides a brief review of the basic operational features of a PCM and issues related to resistance drift and multilevel storage as affecting its data integrity. Also a brief review of existing codes as related to the proposed scheme is presented.

2.1 Phase Change Memory

This section reviews different aspects as related to PCM, resistance drift behavior and multilevel storage. As described previously, PCM is regarded as one of the most viable candidate for the next generation of non-volatile memories [1]. A PCM relies on the reversible phase transformation of the chalcogenide alloy (e.g. $Ge_2Sb_2Te_5$, GST) between the amorphous and the crystalline states. The amorphous state has a high resistance and is commonly referred to as the *reset state*; the crystalline phase has a low resistance and is referred as the *set state*. If the PCM is in the Reset state (amorphous) and the voltage across the PCM cell is higher than the threshold value, then a snapback behavior occurs and the resistance of the PCM is switched to the ON state value. If the PCM is in the ON state, it will switch back to the OFF state if and only if the voltage across the PCM is less than the so-called *ON/OFF Intersection Point*. A PCM cell can be used as a multilevel memory to increase capacity; this is made possible by its high resistance range, i.e. the difference between the resistances of the SET and RESET states.

However, after a PCM cell is programmed, its resistance increases with time; this phenomenon is generally known to as the resistance drift. The *resistance drift* is believed to be the result of structural relaxation (SR) phenomena that are thermally activated as an atomic rearrangement of the amorphous structure [10], [13]. A multilevel PCM experiences a difference in resistance drift over time, leading to a significant degradation in data integrity [13]. Fig. 1 shows the basic principles of resistance drift in a multilevel PCM cell. The resistance at each level varies according to a Gaussian distribution and accounts for less (more) drift when the PCM is in the (amorphous) crystalline phase. The resistance of each level changes during T (Fig. 1) and it could pass the threshold value (as separating two adjacent levels). This results in an erroneous output following a read. The erroneous effects of the resistance drift in a PCM cell can be alleviated if the threshold resistances could also vary with time. In [14], a time-aware fault-tolerant scheme is used for correcting the resistance drift of a PCM. The drift behavior of the threshold resistance is taken into account by keeping the so-called *lifetime* of the PCM in the form of *time tag* bits and using them to find the threshold resistances.

Multilevel storage is achieved through an accurate programming of the PCM cell into intermediate resistance values, i.e. the values between the SET and RESET states [1]. This scheme however, is susceptible to process and material variability; for example, the temperature that is generated in the PCM by using the same programming pulse, varies from cell to cell. Therefore, a single pulse

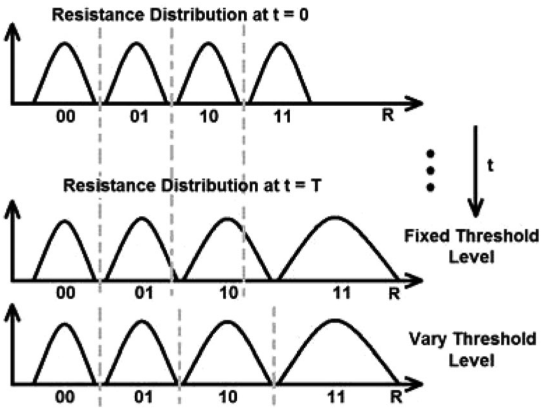

Fig. 1. PCM resistance distribution over time.

programming arrangement is not a viable option for multilevel PCM storage, because the resulting resistance level distributions are rather broad and difficult to predict [15]. A possible solution is to employ an *iterative programming strategy* that starts by reading the most recent resistance value of a PCM and comparing with a reference value; then, a programming current is utilized to bias and adjust the resistance of the PCM cell to the desired value. However, mitigation schemes such as the ones proposed in [14] are not sufficient to deal with the issue of drift once a multilevel PCM is considered. This is mostly caused by the reduced resistance values between two adjacent levels and the characteristic that the drift is more pronounced at higher values of resistance in the PCM range. Moreover, the additional number of operations to be performed may further degrade the endurance of a non-volatile memory cell, such as a PCM [13].

2.2 Coding Techniques

The Hamming code is one of the most frequently ECCs used in a memory system [7]; it covers both binary and non-binary data, but it only corrects single errors. To control multiple errors, BCH codes and Reed-Solomon codes have been widely adopted [7]. A universal parallel decoding method has been proposed in [16]; any multiple-symbol error correcting codes, including RS codes, can be decoded in parallel using this method. However, this method incurs in a large hardware overhead when utilized for multiple-symbol error correction. Better codes can be obtained by arranging the data in the memory array and using codes with a weak capability for each row and column; these are generally known as matrix codes (also referred as product codes) [7]. Decoding of matrix codes is generally difficult; recently, a decoding algorithm for a double-error correcting matrix code has been proposed in [11], [12] that partially alleviates this negative feature.

The low- density parity check (LDPC) code is another well-known class of codes capable of correcting multiple errors. LDPC codes are usually decoded by iterative decoding schemes, such as the one used for PCM in [14]. However, this scheme requires a long time for decoding. Chen et al. have proposed a high-speed decoding scheme for LDPC codes by satisfying the so-called row-column (RC) constraint [17]. It has been proved [17] that if and only if a code satisfies the RC constraint in the parity check H matrix, no two rows or two columns have more than one place where they both have nonzero elements. This scheme is based on one-step majority-logic decoding (OSMLGD) [7]. Hsiao et al. [9] has proposed an OLS code capable of correcting multi-bit errors. As it satisfies the RC constraint, this code can be decoded at high-speed using OSMLGD [18], thus confirming the excellent suitability of the OLS code for a memory system [18]. Datta and Touba [8] has shown that the OLS code can be used in a PCM; it should be noted that the OLS code is a binary code targeting only bit-errors.

Next, the binary OLS code [9] is reviewed. The OLS code uses orthogonal Latin squares, which are $m \times m$ square arrays of digits $0, 1, \ldots, m-1$. Each digit occurs exactly once in each row and exactly once in each column. Two Latin squares $L = [l_{x,y}]$ and $L' = [l'_{x,y}]$ are said to be orthogonal if every pair of elements $(l_{x,y}, l'_{x,y})$ appears exactly once. For example the following two squares L and L′ are orthogonal Latin squares:

$$L = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix}, \quad L' = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}. \tag{1}$$

The following matrix H is an H matrix (i.e. a parity check matrix) of a binary t-bit error correcting OLS code:

$$H = \begin{bmatrix} M_1 & \\ M_2 & \\ M_3 & I_{2tm} \\ \vdots & \\ M_{2t} & \end{bmatrix},$$

where $M_i (1 \le i \le 2t)$ is an $m \times m^2$ matrix defined below:

$$M_1 = \begin{bmatrix} 11\cdots1 & & & O \\ & 11\cdots1 & & \\ & & \ddots & \\ O & & & 11\cdots1 \end{bmatrix}_{m\times m^2}$$

$$M_2 = [\, I_m \quad I_m \quad \cdots \quad I_m \,]_{m\times m^2}.$$

The matrices $M_i (3 \le i \le 2t)$ are generated from the Latin squares L_i such that any two Latin squares are orthogonal. Moreover, the check bit length of the OLS code (denoted by r) is equal to $2tm$; m^2 must be equal to or larger than the information bit length k. In such cases it is not always true that $r = 2t\sqrt{k}$ because $2t-2$ orthogonal Latin squares of size $m \times m$ not always exist. Colbourn and Dinitz [19] has discussed the maximum number of orthogonal Latin squares. The condition that $r \ge 2t\sqrt{k}$ is still applicable. The OLS code is capable of correcting t-bit errors, because it satisfies the RC constraint and the minimum column weight of its H matrix is $2t$ [7]. The minimum distance of a code satisfying the RC constraint is equal to $\gamma + 1$, where γ is the minimum column weight of its H matrix. Thus, the minimum distance of the OLS code is $2t+1$, i.e. the code is capable of correcting t-bit errors.

OLS codes can be decoded using OSMLGD [7]. There are two types of OSMLGD, namely type-I and II. In type-I, the syndrome is generated and then the error pattern is calculated from the syndrome. In type-II, the error pattern is directly calculated from a received word. It is well known that type-II accomplishes better serial decoding than type-I. However, type-I is better suited for parallel decoding than type-II (as discussed next). Decoding for PCM requires a high-speed parallel scheme; so this paper deals with only type-I. A type-I based decoder requires two steps namely syndrome generation and error pattern calculation; the decoder must control these steps. A type-II based serial decoder is simpler than a type-I. A type-II based parallel decoder incurs in a larger overhead than a type-I based parallel decoder, because these two steps in type-I are accomplished by serially connecting a syndrome generator and an error pattern calculator without complex control. In addition, the direct calculation in a type-II incurs in a larger overhead than the combination of syndrome generation and error pattern calculation; this overhead is rather large for the entire decoder.

OLS codes can be decoded as follows. Let S_j be a vector that contains all i-th elements s_i in a syndrome $(s_0, s_1, \ldots, s_{(r-1)})$ such that $h_{i,j}$ in the H matrix is 1. Suppose that a t-bit error occurs on a received

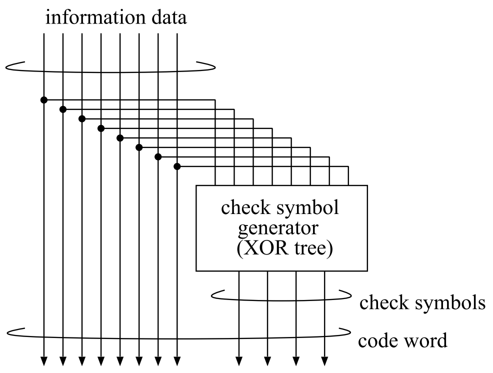

Fig. 2. Proposed encoder design.

word v. If the i-th bit in v is erroneous, then the values of at least $(t+1)$ bits in S_j are 1's. If not, the values of at least t bits are 0's. Based on these conditions, errors can be corrected by flipping all received bits, such that at least $(t+1)$ bits in S_j are 1's, i.e. by adding the majority of all values in S_j and a value 0 to all received bits.

3 Proposed Approach

This section presents the proposed non-binary multi-symbol error correcting OLS codes to accomplish the following features as highly desirable for PCM storage: 1) non-binary, 2) multiple-error correction and 3) parallel decoding. They are reflected in the H matrix and the encoder/decoder designs as treated next. The proposed non-binary OLS code can be constructed over finite rings. However, PCM requires codes over only $GF(2^b)$; so, the hardware amount of decoder can be reduced for codes over $GF(2^b)$.

Consider first the H matrix. The proposed code is over a finite ring. The proposed (n symbols, k symbols) t-symbol error correcting code uses the same H matrix as an (n bits, k bits) binary t-bit error correcting OLS code (the H matrix consists of only additive and multiplicative identities, 0 and 1). This condition is valid because the H matrix of the non-binary OLS code satisfies the RC constraint. In addition, the minimum column weight of the H matrix is the same, i.e. $2t$. Therefore the minimum distance of the proposed non-binary OLS code is $2t+1$, and the proposed code is capable of correcting t-symbol errors. The relation between the information symbol length k, the check symbol length r and the number of correctable error symbols t is the same as that for the binary OLS code, i.e. $r \geq 2t\lceil\sqrt{k}\rceil$. Figs. 2 and 3 illustrate the block diagrams of the encoder and decoder for the proposed code. These circuits have a traditional structure, so the check symbol generator in the encoder, and the syndrome generator and the adder in the decoder can also be implemented in a traditional scheme using additional circuitry for the ring (i.e. the XOR gates for codes over GF(2^b)).

Next, the construction of the error pattern calculator in the decoder is described. This uses OSMLGD [7]. Let S_j be a vector that contains all ith elements s_i in a syndrome $(s_0, s_1, \ldots, s_{(r-1)})$, such that $h_{i,j}$ in the H matrix is 1. The error magnitude e_j in the jth symbol of the received word is found by calculating the majority in S_j and a value 0 when a t-symbol error occurs. This is valid because if the value of the jth symbol changes from v_j to $v_j + e_j$, at least $t+1$ symbols in S_j are equal to e_j. Moreover, if the value of the jth symbol does not change, at least t symbols are 0. Next consider GF(2^b). A circuit of reduced complexity can be used as majority circuit in a decoder for a non-binary OLS codes over GF(2^b). The circuit does not use b-to-2^b decoders and 2^b-to-b encoders. Instead, it counts the number of 1's in the binary-coded digits. It does not always work as a majority circuit; however, it has sufficient functionality for use in the decoder. For example, if $S_j = (3,3,4,5) = (011,011,100,101)$ over GF(2^3), then it outputs $1 = (001)$ although the majority is 3.

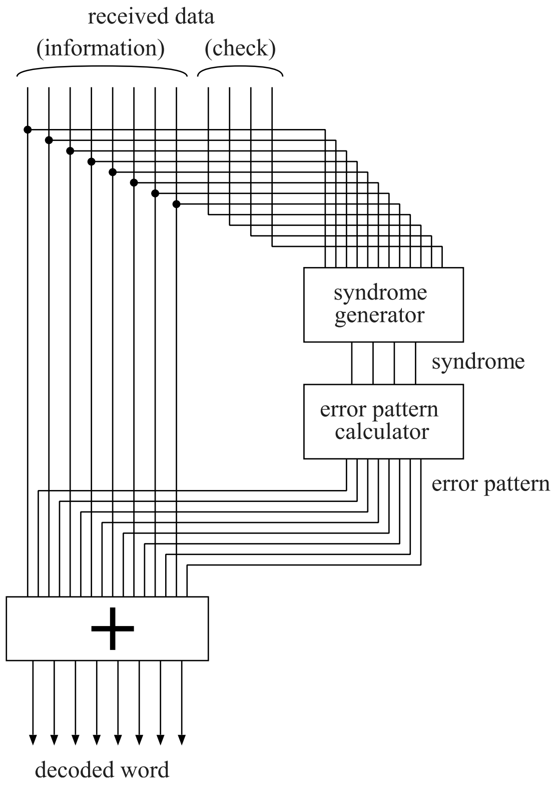

Fig. 3. Proposed decoder design.

However, such S_j does not appear when a t-symbol error occurs (as previously discussed). At least $t+1$ symbols in S_j are equal to the error magnitude e_j, and thus, every bit in the binary-coded e_j is selected as the majority for each digit.

4 Evaluation

This section evaluates the proposed 2^b-ary t-symbol error correcting OLS codes and compares them to the binary tb-bit error correcting OLS codes [9], capable of correcting t-symbol errors, for $b = 2$, 3, 4 and 5. These values for b reflect current and expected multilevel PCM capabilities as utilized for communication systems (at $b = 3$ for ternary content addressable memory operation) [10] and large storage systems (at higher values of b for increased density) [5]. This section uses the lower bound of the maximum number of orthogonal Latin squares presented in [19] as the maximum value. The obtained evaluation results are given as follows (note that RS codes and matrix codes are discussed and evaluated later.)

Table 1 shows the relation between the length of the information and the check symbols for 2^b-ary double symbol error correcting OLS codes; the proposed codes are better than the conventional binary codes for any parameters. Fig. 4 shows the relation between the number of correctable symbol errors and the check symbol length; Fig. 5 shows the legend of the figure. The curves of the proposed code for $b = 2$ and four completely overlap, because the check symbol length of the proposed code is independent of b. The proposed code is much better than a conventional scheme for larger number of correctable symbol errors.

Fig. 6 shows the plot of the symbol error rate before decoding and the probability that the information data (of 256 symbols) is erroneous even after decoding. The curves of the binary and non-binary OLS codes nearly overlap for every b.

The binary OLS codes are capable of correcting not only any double symbol errors, but also any random $2b$-bit errors unlike the proposed non-binary OLS codes. Therefore, the binary codes have

TABLE 1
Check Symbol Length *r* of $(k+r,k)2^b$-ary Double Symbol Error Correcting Codes

b	code	$k=256$	1,024	4,096	16,384
2	non-binary OLS	64	128	256	512
	binary OLS	92	187	364	728
	RS code	15	18	21	24
	matrix code	170	364	770	1,620
3	non-binary OLS	64	128	256	512
	binary OLS	116	234	448	
	RS code	10	12	14	16
	matrix code	216	441	975	2,034
4	non-binary OLS	64	128	256	512
	binary OLS	128	256	512	
	RS code	8	10	11	13
	matrix code	256	516	1,148	2,316
5	non-binary OLS	64	128	256	512
	binary OLS	148	292	594	
	RS code	7	8	9	10
	matrix code	280	640	1,285	2,865

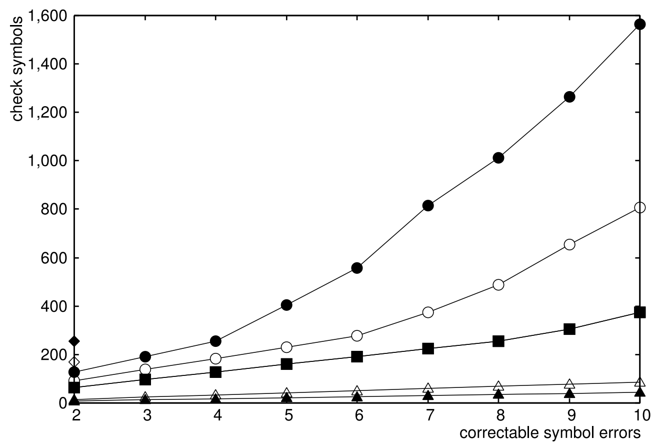

Fig. 4. Number of correctable symbol errors versus check symbol length for information data with length of 256 symbols.

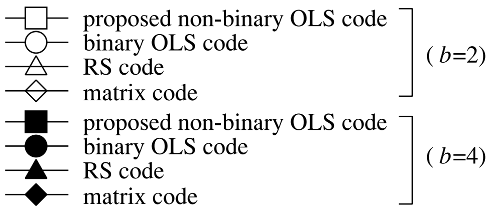

Fig. 5. Legend for Figs. 4, 7, and 8.

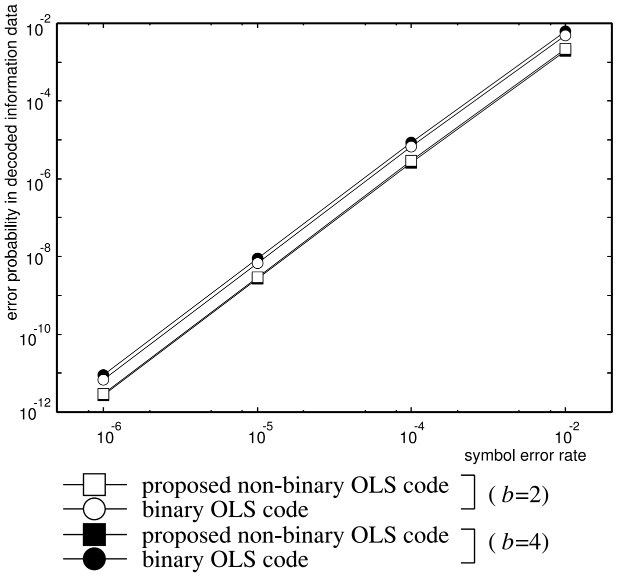

Fig. 6. Symbol error rate before decoding versus error probability in decoded information data for 2^b-ary double symbol error correction (information data of 256 symbols).

TABLE 2
Error Probability in Decoded Information Data for 2^b-ary Double Symbol Error Correction for Symbol Error Rates of 1×10^{-4}

b	code	$k=256$	1,024	4,096	16,384
2	non-binary OLS	2.89×10^{-6}	1.26×10^{-4}	4.94×10^{-3}	8.24×10^{-2}
	binary OLS	6.61×10^{-6}	2.59×10^{-4}	7.80×10^{-3}	5.78×10^{-2}
	RS code	3.22×10^{-6}	1.74×10^{-4}	8.56×10^{-3}	2.20×10^{-1}
	matrix code	2.23×10^{-6}	6.09×10^{-5}	1.57×10^{-3}	2.74×10^{-2}
3	non-binary OLS	2.56×10^{-6}	1.11×10^{-4}	4.38×10^{-3}	7.31×10^{-2}
	binary OLS	7.96×10^{-6}	2.25×10^{-4}	1.62×10^{-3}	
	RS code	3.04×10^{-6}	1.71×10^{-4}	8.52×10^{-3}	2.20×10^{-1}
	matrix code	1.79×10^{-6}	3.62×10^{-5}	9.49×10^{-4}	2.32×10^{-2}
4	non-binary OLS	2.60×10^{-6}	1.11×10^{-4}	4.45×10^{-3}	7.43×10^{-2}
	binary OLS	8.68×10^{-6}	1.73×10^{-4}	2.56×10^{-4}	
	RS code	2.97×10^{-6}	1.70×10^{-4}	8.50×10^{-3}	2.20×10^{-1}
	matrix code	1.46×10^{-6}	3.73×10^{-5}	1.14×10^{-3}	1.68×10^{-2}
5	non-binary OLS	4.19×10^{-6}	1.82×10^{-4}	7.17×10^{-3}	1.20×10^{-1}
	binary OLS	9.97×10^{-6}	9.59×10^{-5}	3.19×10^{-5}	
	RS code	2.94×10^{-6}	1.69×10^{-4}	8.49×10^{-3}	2.20×10^{-1}
	matrix code	1.29×10^{-6}	3.79×10^{-5}	9.40×10^{-4}	1.98×10^{-2}

a higher error correction function. However, the probability is worse than for the non-binary codes because of their long check symbol length; so it is more likely that errors occur in check symbols leading to miscorrection on information symbols when using binary OLS codes. Table 2 shows the relation between error probability and information data length *k*. While the proposed non-binary OLS code is better for short information data, this is worse for long information data. This occurs because the probability of miscorrection is low (instead of the long check symbol length in binary OLS codes).

Next the evaluation of parallel encoders and parallel decoders is pursued with respect to area, power consumption and gate depth. Circuits are obtained by using Verilog-HDL (RTL-level) and synthesized by using the Synopsys Design Compiler under an industrial 180 nm CMOS technology. All designed circuits are combinational and the presented evaluation reports area, power consumption and delay time normalized by those of an inverter (thus making it feature size independent). They are denoted by the ratios of A_C/A_I, P_C/P_I and D_C/D_I where A_C and A_I are the areas of the evaluated circuit and an inverter, P_C and P_I are the power consumptions of the evaluated circuit and an inverter, D_C and D_I are the delay time of the evaluated circuit and an inverter. Fig. 7 and Tables 3 and 4 show the area, power consumption and delay time of the parallel encoders for 2^b-ary double symbol error correcting code (Fig. 5 shows the legend for Fig. 7). The values for the proposed non-binary OLS codes are better than for the binary OLS codes due to the shorter check symbols. Fig. 8 and Tables 5 and 6 show the area, power consumption and delay time of the parallel decoders for 2^b-ary double symbol error correcting codes (Fig. 5 shows the legend for Fig. 8). The decoders for binary OLS codes presented in [18] do not use syndrome generators. However, this circuit requires a larger number of XOR gates (as for Fig. 1 in [8]), thus incurring in a significant increase in area as well as causing an increase in power consumption.

Table 7 gives the number of transistors required for a PCM memory (at expected sizes) as well as the total number of

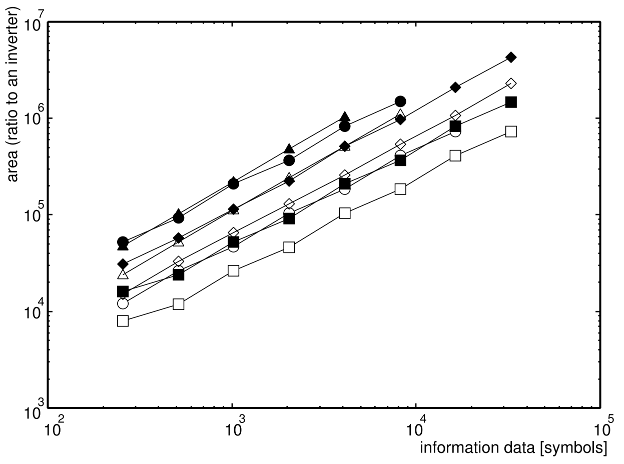

Fig. 7. Area of encoder for 2^b-ary double symbol error correcting codes.

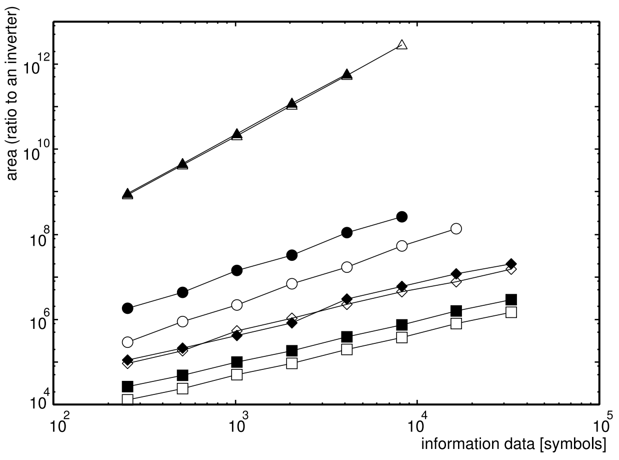

Fig. 8. Area of decoder for 2b-ary double symbol error correcting codes.

TABLE 3
Power Consumption of Encoder for 2^b-ary Double Symbol Error Correcting Codes

b	code	$k = 256$	1,024	4,096	16,384
2	non-binary OLS	1.64×10^4	5.35×10^4	1.86×10^5	6.80×10^5
	binary OLS	2.87×10^4	9.69×10^4	3.35×10^5	1.26×10^6
	RS code	3.88×10^4	1.69×10^5	7.58×10^5	
	matrix code	3.22×10^4	1.27×10^5	4.44×10^5	1.77×10^6
3	non-binary OLS	2.46×10^4	8.02×10^4	2.79×10^5	1.02×10^6
	binary OLS	6.10×10^4	2.33×10^5	7.44×10^5	
	RS code	5.27×10^4	2.37×10^5	1.08×10^6	
	matrix code	4.74×10^4	1.50×10^5	5.93×10^5	2.72×10^6
4	non-binary OLS	3.28×10^4	1.07×10^5	3.72×10^5	1.36×10^6
	binary OLS	1.07×10^5	3.72×10^5	1.36×10^6	
	RS code	7.33×10^4	3.29×10^5	1.51×10^6	
	matrix code	6.12×10^4	2.03×10^5	8.76×10^5	3.30×10^6
5	non-binary OLS	4.10×10^4	1.34×10^5	4.65×10^5	1.70×10^6
	binary OLS	1.65×10^5	5.69×10^5	2.11×10^6	
	RS code	8.77×10^4	4.02×10^5		
	matrix code	6.59×10^4	2.71×10^5	9.78×10^5	4.25×10^6

TABLE 4
Delay Time of Encoder for 2^b-ary Double Symbol Error Correcting Codes

b	code	$k = 256$	1,024	4,096	16,384
2	non-binary OLS	15.8	19.6	21.8	24.7
	binary OLS	17.6	20.5	22.5	25.6
	RS code	28.5	34.9	40.9	
	matrix code	21.1	24.0	27.1	30.2
3	non-binary OLS	15.8	19.6	21.8	24.7
	binary OLS	19.6	21.8	24.0	
	RS code	30.0	36.0	41.6	
	matrix code	21.8	22.5	26.5	31.8
4	non-binary OLS	15.8	19.6	21.8	24.7
	binary OLS	19.6	21.8	24.7	
	RS code	31.8	38.0	44.0	
	matrix code	22.5	25.6	28.9	30.9
5	non-binary OLS	15.8	19.6	21.8	24.7
	binary OLS	19.6	22.5	25.6	
	RS code	31.8	38.0		
	matrix code	22.5	25.8	28.9	32.5

TABLE 5
Power Consumption in Decoder for 2^b-ary Double Symbol Error Correcting Codes

b	code	$k = 256$	1,024	4,096	16,384
2	non-binary OLS	3.18×10^4	1.16×10^5	4.29×10^5	1.66×10^6
	binary OLS	6.70×10^5	4.38×10^6	3.07×10^7	2.28×10^8
	RS code	2.31×10^9	5.79×10^{10}	1.42×10^{12}	
	matrix code	1.59×10^5	8.71×10^5	3.62×10^6	1.68×10^7
3	non-binary OLS	4.77×10^4	1.73×10^5	6.44×10^5	2.49×10^6
	binary OLS	1.72×10^6	1.26×10^7	8.33×10^7	
	RS code	2.38×10^9	6.06×10^{10}	1.48×10^{12}	
	matrix code	1.79×10^5	6.50×10^5	3.73×10^6	1.88×10^7
4	non-binary OLS	6.36×10^4	2.31×10^5	8.58×10^5	3.32×10^6
	binary OLS	3.64×10^6	2.47×10^7	1.77×10^8	
	RS code	2.51×10^9	6.38×10^{10}	1.55×10^{12}	
	matrix code	2.02×10^5	7.42×10^5	4.55×10^6	1.78×10^7
5	non-binary OLS	7.95×10^4	2.89×10^5	1.07×10^6	4.15×10^6
	binary OLS	6.12×10^6	4.16×10^7	2.97×10^8	
	RS code	2.65×10^9	6.70×10^{10}		
	matrix code	2.03×10^5	1.19×10^6	4.55×10^6	2.22×10^7

TABLE 6
Delay Time of Decoder for 2^b-ary Double Symbol Error Correcting Codes

b	code	$k = 256$	1,024	4,096	16,384
2	non-binary OLS	36.7	39.8	42.7	45.8
	binary OLS	45.6	48.5	50.5	53.6
	RS code	82.5	92.9	108.7	
	matrix code	43.8	51.5	54.9	56.2
3	non-binary OLS	36.7	39.8	42.7	45.8
	binary OLS	47.1	49.3	51.5	
	RS code	83.1	97.1	108.4	
	matrix code	45.3	46.0	53.3	60.2
4	non-binary OLS	36.7	39.8	42.7	45.8
	binary OLS	58.7	60.9	63.8	
	RS code	83.1	97.1	111.5	
	matrix code	45.5	48.5	55.3	57.3
5	non-binary OLS	36.7	39.8	42.7	45.8
	binary OLS	54.7	57.6	60.7	
	RS code	84.4	96.4		
	matrix code	46.4	54.2	57.3	60.9

transistors for the encoder and the decoder. Double symbol error correcting codes over GF(2^2) and GF(2^3) are used assuming four- and eight-level memory cells. So, a row includes 32 words; a word includes 16 bits (eight cells). Coding is independent of memory size; it depends on the number of bits in a row. The number of transistors required for the encoders and decoders are significantly smaller than for the memory (except in the case of the RS code).

TABLE 7
Number of Transistors of Memory, Encoder and Decoder for Four-Level and Eight-Level PCMs

(a) Four-level PCM

Memory size (word)	Number of transistors in memory	encoder and decoder			
		non-binary OLS	binary OLS	RS code	matrix code
1G	1.02×10^9	7.16×10^4	1.84×10^6	8.46×10^9	4.30×10^5
5G	5.12×10^9	7.16×10^4	1.84×10^6	8.46×10^9	4.30×10^5
10G	1.02×10^{10}	7.16×10^4	1.84×10^6	8.46×10^9	4.30×10^5

(b) Eight-level PCM

Memory size (word)	Number of transistors in memory	Number of transistors for encoder and decoder			
		non-binary OLS	binary OLS	RS code	matrix code
1G	7.68×10^8	1.07×10^5	4.56×10^6	8.83×10^9	4.70×10^5
5G	3.84×10^9	1.07×10^5	4.56×10^6	8.83×10^9	4.70×10^5
10G	7.68×10^9	1.07×10^5	4.56×10^6	8.83×10^9	4.70×10^5

The proposed 2^b-ary double-symbol error correcting OLS code is compared to the Reed-Solomon code and the matrix code. The Reed-Solomon code is well-known as a multiple- error correcting non-binary code. The correction by the RS code cannot be accomplished in parallel, however the RS code can be decoded by a universal decoding method [16], [20]. The RS code can be utilized for PCM that requires a high-speed parallel decoding. The matrix code of [11], [12] is a double-symbol error correcting code; information data is arranged in an array; each row and each column are encoded with a single-symbol error correcting and double-symbol error detecting (SEC-DED) code and a single-symbol error detecting (SED) code.

While the code of [11], [12] is a binary code, it can be easily extended to non-binary codes. This comparison is based on utilizing Reed-Solomon and parity codes for SEC-DED and SED; moreover it also utilizes the row and column lengths to provide the minimum total number of check symbols.

Figs. 4, 7, and 8 and Tables 1, 2, 3, 4, 5, and 6 show the comparison results between these two codes and the OLS code. The check symbol length of the proposed code is longer than the RS code. However, the parallel decoder [16] requires a considerable hardware overhead. The same condition is also valid for any multiple-error correcting non-binary codes known to be not detectable in parallel. The proposed codes are always better than the matrix codes of [11], [12] i.e. independently of the metric and the parameters employed in the comparison of the check symbol length and the hardware overhead. In addition, it should be pointed out that the construction of the decoders for matrix codes with long Hamming distance is rather difficult; this is not encountered in the proposed OLS codes (this is the reason that Fig. 4 does not show the check symbol length of the matrix codes except for the correctable symbol error of two).

5 Conclusion

This paper has presented 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. The proposed (n symbols, k symbols) t-symbol error correcting code uses the same H matrix as an (n bits, k bits) t-bit error correcting binary OLS codes. It utilizes a shorter check length and higher reliability for information symbol length of approximately 10^3 or less than binary OLS codes. In addition, the parallel schemes for encoding and decoding in the proposed codes have been shown to be better than those for binary OLS codes in terms of area, power consumption and delay.

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 phase-change 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 behavior 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.

[14] W. Xu and T. Zhang "A time-aware fault tolerance scheme to improve reliability of multilevel phase-change memory in the presence of significant resistance drift," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 8, pp. 1357–1367, Aug. 2011.

[15] P. Junsangsri, J. Han, and F. Lombardi, "Macromodeling a phase change memory (PCM) cell by HSPICE," in *Proc. IEEE/ACM Int'l Symp. Nanoscale Archit.*, 2012, pp. 77–84.

[16] E. Fujiwara, K. Namba, and M. Kitakami, "Parallel Decoding for Burst Error Control Codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2002, p. 429.

[17] C.-Y. Chen, Q. Huang, C.-C. Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC codes," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3140–3147, Nov. 2010.

[18] Z. Chishti, A. R. Alameldeen, C. Wilkerson, W. Wu, and S.-L. Lu, "Improving cache lifetime reliability at ultra-low voltages," in *Proc. Annu. IEEE/ACM Int. Symp. Microarchit*, 2009, pp. 89–99.

[19] C. J. Colbourn and J. H. Dinitz, *The CRC Handbook of Combinatorial Designs*. Boca Raton, FL, USA: CRC Press, 1996.

[20] E. Fujiwara, *Code Design for Dependable Systems: Theory and Practical Applications*. New York, NY, USA: Wiley-Interscience, 2006.