IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 23, 2020, accepted February 3, 2020, date of publication February 14, 2020, date of current version March 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974013

Cost-Effective Reliable MLC PCM Architecture Using Virtual Data Based Error Correction

TAEHYUN KWON[1,2], (Student Member, IEEE), MUHAMMAD IMRAN[3], (Student Member, IEEE), AND JOON-SUNG YANG[4], (Senior Member, IEEE)

[1]System LSI Division, Samsung Electronics, Hwaseong 18448, South Korea
[2]Department of Semiconductor and Display Engineering, Sungkyunkwan University, Suwon 16419, South Korea
[3]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea
[4]Department of Systems Semiconductor Engineering, Yonsei University, Seoul 03722, South Korea

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea by the Ministry of Education under Grant NRF-2018R1D1A1B07049842 and Grant NRF-2015R1D1A1A01058856, and in part by the Ministry of Trade, Industry and Energy through the Korea Semiconductor Research Consortium support program for the development of the future semiconductor device under Grant 10080594.

ABSTRACT The existing charge-based memories like DRAM and flash are reaching their scaling limits. Phase change memory (PCM) is one of the most promising emerging memory technologies due to its good scalability and low leakage power. Multi-level cell (MLC) operation in PCM, which stores two or more bits in a single cell, is necessary to achieve a high storage density. However, a reduced resistance range in multiple storage levels of MLC PCM, introduces a lot of soft errors because of the resistance drift phenomenon. The poor reliability of MLC PCM requires strong error correction codes (ECC) which could severely degrade the storage density and performance. In this paper, we propose a cost-effective reliable MLC PCM architecture for improving MLC PCM reliability, storage density and performance. The proposed architecture exploits the data-dependent nature of resistance drift problem to reduce the ECC overhead. A simple state mapping is used to generate *virtual data*, which is half of the actual data size. ECC parity bits are generated based on virtual data bits instead of actual message bits, thus resulting in a reduced number of cells for parity bits. This improves the reliability and storage density of MLC PCM. The performance is also improved by minimizing the ECC overhead. Simulation results show an improvement of about 10^4 times in reliability and 10.9% in storage density compared to a conventional MLC PCM which uses a typical error correction scheme. Performance and energy efficiency are also improved up to 13.7% and 10%, respectively, by the proposed architecture.

INDEX TERMS Emerging memories, phase change memory (PCM), MLC PCM, resistance drift, error correction code (ECC).

I. INTRODUCTION

Morden computer systems utilize large memory for computing and storing data and the required memory size continues to increase steadily. The existing charge-based memories like DRAM and flash have successfully been used over decades due to their good scalability. However, they are reaching their scaling limits in deep sub-micron technology [1], [2]. Moreover, leakage energy becomes more and more critical since the proportion of leakage energy grows rapidly with an increasing memory size and technology scaling. This has motivated research on several new nonvolatile memory technologies. Among these emerging memories, Phase Change Memory (PCM) is one of the most promising candidates, with a good scalability, low leakage power and good compatibility with the current CMOS fabrication technology [3]–[6].

PCM uses chalcogenide material for storing data. The GST ($Ge_2Sb_2Te_5$) alloy is one of the materials practically used in PCM. This material has two different stable states and has a wide resistance range (about 10^3 Ω) between the two stable states [7]. The two different states are an amorphous state with a high resistance (about 10^6 Ω) and a crystalline state with a low resistance (about 10^3 Ω). Using these two states, a single-level cell (SLC) PCM stores one bit in a single cell. When the GST material is heated above its melting temperature (around 600°C) by a large amplitude current pulse for a short period of time, the material changes its state to amorphous state. This constitutes the RESET operation and the corresponding state

The associate editor coordinating the review of this manuscript and approving it for publication was Lorenzo Ciani.

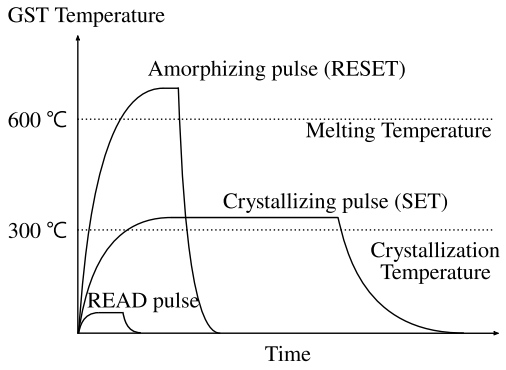

FIGURE 1. Current pulses for read and write operation in a SLC PCM.

represents logic 0. In contrast, a relatively small amplitude and long duration current pulse, with a temperature above the crystallization temperature (around 300°C) but below the melting temperature, changes the material into crystalline state. This state represents logic 1 and the operation is the SET operation. A smaller and shorter than SET current pulse, is used for read operation. Fig. 1 depicts the three current pulses for write and read operations.

A wide resistance range of the phase change material has motivated the multi-level cell (MLC) PCM. MLC PCM divides the full resistance range into n levels so that it can store $log_2(n)$ bits in a single cell. MLC operation is necessary to achieve a high storage density (data bits/cells) of PCM and to successfully replace the conventional charge-based memories. Resistance drift in phase change material increases its resistance over time [1], [8], [9]. Unlike SLC PCM, it may introduce a number of soft errors due to the narrower resistance range for each level in an MLC PCM. The poor reliability by resistance drift is a critical issue which limits the practical usage of MLC PCM. To mitigate the drift-induced errors, there has been some research done. In [1], [10]–[13], various architectural approaches have been introduced to improve the reliability of MLC PCM. Different encoding ideas have also been proposed which can considerably enhance the reliability [8], [14], [15]. However, these architectural and encoding methods cannot remove the impact of resistance drift completely and require a multiple-error correction scheme like Bose-Chaudhuri-Hocquenghem (BCH) code to achieve a soft error rate (SER) similar to that of DRAM or flash.

Error correction code (ECC) for multiple errors usually requires a large amount of additional storage for parity bits. It decreases the storage density of an MLC PCM. Multiple RESET or SET current pulses for a precise MLC PCM programming and a long encoding and decoding latency of ECC degrade performance and energy efficiency significantly. Therefore, an ECC scheme should be carefully selected and optimized if possible for better storage density, reliability, performance and energy efficiency.

In this paper, we introduce a cost-effective reliable MLC PCM architecture for MLC PCM. Based on the characteristics of resistance drift errors in MLC PCM, the proposed

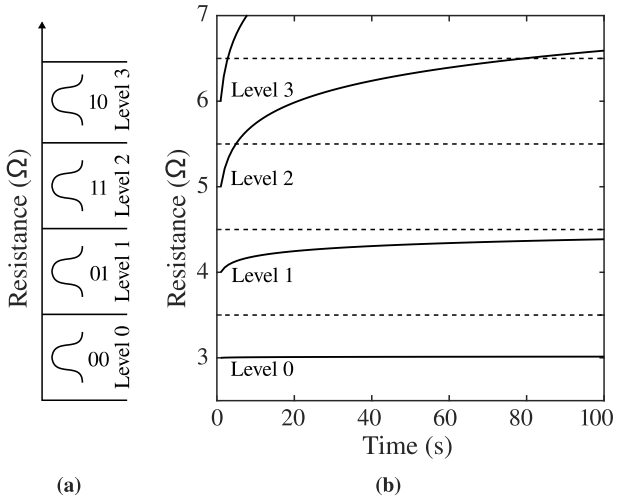

FIGURE 2. (a) Resistance distribution of a 4LC PCM (b) Resistance drift in a 4LC PCM.

architecture generates virtual data with reduced data size than that of original data and uses it for parity generation. Since the number of parity bits depends on the data size, the virtual data minimizes the additional cells for storing parity bits. The reduced number of cells for parity bits improves both the reliability and storage density of MLC PCM. Moreover, the use of a lighter ECC scheme improves performance and energy efficiency with the reduced ECC latency and cell programming. Since the virtual data generation and error correction is performed by simple state mapping, the performance overhead is negligible. Moreover, there is no storage overhead because the virtual data does not need to be stored.

The rest of this paper is organized as follows. Background of this work is summarized in Section II. In Section III, various techniques for improving the reliability and storage density of MLC PCM are described. The proposed efficient error correction architecture is explained in detail in Section IV. Section V analyzes the impact of the proposed architecture on reliability, storage density and performance. Section VI concludes the paper.

II. BACKGROUND

A. READ AND WRITE IN MLC PCM

Multi-level cell (MLC) operation allows a single PCM cell to store two or more bits. An MLC PCM divides the resistance range of phase change material into several levels. The number of bits that can be stored in a single PCM cell is decided by the number of resistance levels between the fully amorphous state and the fully crystalline state. For example, by dividing the resistance range into four (n) levels, a cell can store two ($log_2(n)$) bits. Fig. 2(a) shows the resistance distribution of a 4-level cell (4LC) PCM. In this figure, the lowest resistance level corresponding to fully crystalline state represents logic 00 while the highest resistance level corresponding to fully amorphous state represents logic 10. Two intermediate states represent 01 and 11, respectively. Even though the programming pulses target the mean resistance of each level,

the initial resistance could be placed within a resistance range of each level and this follows a normal distribution. When the resistance crosses the level boundary by drift, a soft error occurs. To minimize the soft errors by resistance drift, a gray encoding (in the order of 00, 01, 11, 10) for state mapping is generally used. This guarantees only a single-bit error when a level change by resistance drift occurs. Next subsection discusses this in more detail.

MLC PCM normally requires multiple current pulses for a precise programming. There are two programming methods for MLC PCM [16]. In S2R (SSMR) programming, a single SET pulse, initially changes the phase change material to the fully crystalline state. Following multiple RESET current pulses partially amorphize the material and increase the resistance until reaching the target resistance. On the contrary, in case of R2S (SRMS), a single RESET pulse initializes the material to the fully amorphous state. Multiple SET pulses then follow to decrease the initial resistance until it reaches the target resistance. These multiple SET and RESET current pulses result in performance and energy efficiency degradation. Even though we consider an adaptive approach [15], [16], the latency and energy consumption for the two intermediate levels are still a big concern. Moreover, a program and verify (P&V) sequence, required for more accurate programming, further degrades the performance and energy efficiency. Therefore, a large number of MLC cells for ECC parity bits could be a considerable burden to performance and energy efficiency.

B. RESISTANCE DRIFT

There are four sources of soft errors in PCM. They are resistance drift, write disturbance, read recovery disturb and read disturb [17]. Write disturbance which may decrease the resistance, occurs when a cell is heated by a neighboring cell programming. However, it could be completely eliminated by keeping enough inter-cell space within the word-lines and across the bit-lines [2], [18]. Read recovery disturb, which occurs when a cell is read immediately after it has been programmed to a high resistance state, could also be removed with using appropriate read-after-write timing [19], [20]. The voltage based sensing (M-metric) approach avoids the read disturb which decreases resistance [21], [22]. However, resistance drift cannot be completely removed and still requires a heavyweight error correction scheme which significantly degrades reliability, performance and energy efficiency in MLC PCM.

Resistance drift is a phenomenon in which the resistance of the chalcogenide material increases over time. Resistance drift in MLC PCM is critical since MLC PCM has narrower resistance boundaries than that of SLC PCM. The resistance by drift, $R_{drift}(t)$ at time t, can be described by the following equation [23]:

$$R_{drift}(t) = R_0(\frac{t}{t_0})^{\alpha} \quad (1)$$

R_0 is an initial resistance while t_0 is an initial time. α is a drift exponent. In Equation 1, $log_{10}R_0$ follows a normal distribution $\mathcal{N}(\mu_{R_0}, \sigma^2_{R_0})$ and the initial resistance range is $10^{\mu_{R_0} \pm 2.75\sigma_{R_0}}$. Each level has $10^{\mu_{R_0} \pm 3\sigma_{R_0}}$ as the level threshold. α follows a normal distribution $\mathcal{N}(\mu_\alpha, \sigma^2_\alpha)$ as well [24].

TABLE 1. Parameters (R_0 and α) for a 4LC PCM.

Storage Level	Data	$\log_{10} R_0$		α	
		μ_{R_0}	σ_{R_0}	μ_α	σ_α
0	00	3.0	$\frac{1}{6}$	0.001	$0.4 \times \mu_\alpha$
1	01	4.0		0.02	
2	11	5.0		0.06	
3	10	6.0		0.10	

TABLE 2. The probabilities of soft error for a 4LC PCM.

Time (s)	Level 1 (01)	Level 2 (11)	Average SER
2^1	Negligible	5.88E-6%	1.47E-6%
2^2	1.59E-12%	0.02%	5.35E-3%
2^3	5.89E-6%	0.12%	0.03%
2^4	7.50E-4%	0.29%	0.07%
2^5	6.51E-3%	0.53%	0.13%
2^6	2.14E-2%	0.86%	0.22%

Fig. 2(b) demonstrates the drift rate based on Equation 1 with parameters listed in Table 1 [23]. Y-axis represents the resistance $R_{drift}(t)$ at time t in log-scale while X-axis shows time t. We assume that the initial resistances is the mean resistance of each storage level. From Fig. 2(b) xit is obvious that the resistance drift rates depend on the initial resistance R_0. With the lowest initial resistance (corresponding to Level 0), the resistance drift is negligible and it does not cause any error over decades. Even though, there is the largest resistance increase in Level 3, it also does not cause any error in the given resistance range of the material. Both Level 1 and Level 2 can introduce soft errors by resistance drift. Especially, the resistance in Level 2 can easily cross the level boundary, leading to an error in a very short time. Table 2 shows the probability of a soft error of 4LC by time t for the two intermediate levels. The average soft error rate of a single 4LC PCM is described in the fourth column and it is clear that a 4LC PCM requires a strong multiple error correction scheme with frequent memory scrubbing [1]. In [13], [24], a conventional 4LC requires strong BCH codes such as BCH-24 (for detecting and correcting 24-bit errors) or even stronger to achieve an SER similar to that of DRAM ($25 \times 10^{-12} \sim 75 \times 10^{-12}$ per bit-hour) for a given scrubbing frequency. Although, some architectural methods or encoding techniques improve the reliability significantly, they still require strong BCH codes (around BCH-16), which adds a non-negligible overhead.

C. ERROR CORRECTION SCHEME FOR MLC PCM

MLC PCM requires a multiple-error correction scheme due to the poor reliability and Bose-Chaudhuri Hocquenghem

(BCH) codes are widely used in this regard. It is a subclass of linear cyclic codes. The codeword after BCH encoding consists of message bits and parity bits. The capability of BCH codes required in MLC PCM is determined by the soft error rate of word-line (64 Byte) for a given frequency of memory scrubbing operation [1], [24]. The SER of word-line in a 4LC PCM is affected by the bit error probability of a single 4-level cell and the number of required 4-level cells. The lower bit error probability or smaller number of 4LC cells results in a lower word-line SER. The detailed explanation and mathematical equation to estimate the word line SER is described in Section IV. BCH codes require additional 4-level cells to store parity bits and the number of required parity bits can be found using following relation [25]:

$$r = t \cdot ceil(log_2 k) + 1 \tag{2}$$

where k is the size of message bits while t is the number of correctable errors by the considered BCH code. The size of parity bits is determined by message bits and correctable errors. For example, if a system with 64 byte word-line utilizes BCH-8, the number of parity bits required is 73. Total 37 4-level cells are required for storing parity bits. It is approximately 14% of the message bits and it significantly reduces the word-line SER. If the memory system uses BCH-24, the fraction of the additional cells for parity is up to 42% of the message bits.

Encoding and decoding logic becomes more complicated as a BCH error correction capability increases. The long latency of strong BCH codes substantially degrades the performance. Especially, the decoding latency comprising of multiple cycles in the performance-critical read operation impacts more than the BCH encoding latency for write operation. The latency of BCH decoder, τ_{BCH} (in terms of the delay of a fanout-of-4 inverter so that it is technology-independent. $\tau_{FO4} = 9ps$ at 22nm CMOS technology) is estimated in [26] as:

$$\begin{aligned}\tau_{BCH} \approx\ & 0.7tlog_2 n + 0.7t + 8tlog_2 log_2 n + 3.6tlog_2 t \\ & + 2.5log_2 log_2 n + 1.8log_2 t + 1.8\end{aligned} \tag{3}$$

where t is the number of correctable errors while n is the code length including parity bits. When we consider BCH-24 for conventional 4LC PCM, the latency overhead by BCH decoding is approximately 11ns and this is more than 9% of the MLC read latency of 120ns [24]. This is not a negligible latency overhead and degrades the overall performance. Therefore, minimizing both the parity storage and decoding overhead of BCH codes is crucial for improving MLC PCM reliability, performance and storage density.

III. RELATED WORK

A number of researches have been done to tackle issues related to MLC PCM reliability, storage density and performance. The research directions can be mainly categorized into two groups:

One is architectural methods which improve MLC PCM reliability and performance by changing the memory architecture. An efficient scrubbing mechanism which adopts both a lightweight error detection mechanism and an optimal error correction code is introduced in [1]. A variable resistance spectrum assignment method in [3] improves MLC PCM reliability along with the improvement in performance and energy efficiency. In [10], [23], time-aware read and write operations increase MLC PCM reliability significantly. A hybrid architecture that can adapt between LLPCM (low-latency) and HDPCM (high-density with high latency) modes is proposed for better performance [11]. In [12], a write truncation and form switch reduces the number of write iterations with a reduced storage overhead. Heterogeneous PCM array architecture which uses both SLC and MLC in a single word-line is introduced in [13]. In [22], a hybrid sensing approach of current-based sensing and voltage-based sensing is introduced to minimize the impact of resistance drift. A state-aware and PV-aware adaptive programming technique is proposed in [16]. The encoding techniques for STT-MRAM may possibly be integrated with the proposed method to address drift-induced soft errors in PCM. [27] proposes a read/write error-rate aware coding technique which decreases the read disturbance and write failure rates of STT-MRAM by reducing the number of 1s and 0->1 transitions. An online pattern recognition technique in [28] is proposed to improve the endurance of NVM-based caches.

The other group enhances the MLC PCM reliability by using various encoding techniques. A resistance drift resilient encoding technique is introduced in [8] and it achieves significant reliability improvement. The flip-and-write technique flips data to reduce error-prone data patterns (corresponding to Level 2) and this improves both performance and energy efficiency [14]. Tri-level-cell phase change memory improves MLC PCM reliability significantly with ternary encoding [24]. The encoding technique introduced in [15], reduces energy consumption by minimizing two intermediate states for better energy efficiency and it also improves MLC PCM reliability as well. A data compression technique can be adopted with other techniques for improving reliability, performance and energy efficiency [12], [29]. A thermally adaptive ECC design, which can adjust the ECC protection strength considering temperature to reduce the ECC storage overhead of STT-MRAM, is introduced in [30]. In [31], a read error accumulation preventing cache is proposed to completely eliminate the accumulation of read disturbances. ROBIN in [32] more evenly distributes the total number of transitions between codewords to optimize ECC capability. [33] introduces an organized interleaved ECC scheme to improve the reliability of STT-MRAM.

The methods listed above considerably enhance reliability, storage density and performance, using architectural changes or encoding techniques. However, most of conventional works still require strong ECC schemes to achieve acceptable reliability similar to the SER of DRAM. To minimize the ECC overhead, this paper proposes a cost-effective

reliable MLC PCM architecture. Since the proposed architecture is orthogonal to all architectural and encoding methods, the proposed method can be integrated with existing approaches to further enhance MLC PCM reliability.

There has been some works done which modify algorithms of error correction codes to enhance reliability. In [34], a two dimensional parity check with variable length error detection code is proposed to reduce ECC redundancy. Progressive ECC technique in [35] has two types of progressive coding techniques of local progressive ECC (LPE) and global progressive ECC (GPE). An adaptive way allocation for reconfigurable ECCs (AWARE), which uses simple ECC for the majority of writes and strong ECC for high error rate writes, is proposed in [36]. The code in [37] is an enhanced version of OLS codes which is based on H matrix. Similarly, in [38] and [39], the enhanced versions of OLS code are proposed which consider limited magnitude. Each method reduces a code length by considering the limited magnitude at cost of a slight increase in decoding and encoding latency. Two different error detection schemes which consider limited magnitude errors are introduced in [40]. One is a code to detect magnitude-1 error with one-bit parity while the other one uses two-bit parity for magnitude-1 and magnitude-2 error detection. Unlike the methods listed above, a main focus of the proposed method is not an error correction coding (ECC) method or algorithm itself. The proposed method rather adds a simple preprocessing step in a conventional PCM architecture and utilizes an existing systematic cyclic codes without any algorithmic change, to enhance the reliability of PCM. Any of enhanced versions of systematic code which do not make any change in data input, could be used instead of conventional BCH code for the proposed architecture.

IV. PROPOSED COST-EFFECTIVE RELIABLE MLC PCM ARCHITECTURE

A. PROPOSED ENCODER WITH VIRTUAL DATA

For multi-error detection and correction, linear cyclic codes like BCH codes are widely used due to their good capability and simple hardware implementation. In BCH codes, for example, parity bits are first generated with shift and simple logical operations and then stored in memory along with the original message bits. In Equation 2, the size of parity bits is determined by the number of correctable errors and the message size. Since the number of correctable errors is decided by the current soft error rate so that all the errors can be corrected with it, it is not controllable. On the other hand, the message bits, k, in Equation 2 can be minimized to reduce the parity size.

The proposed architecture utilizes the characteristic of gray state encoding in MLC PCM. With gray encoding, even though a single four-level cell (4LC) stores 2-bits per cell, there would be only 1-bit change by a resistance drift induced state change. However, both bits in one cell are considered for parity generation since we don't know which bit would be changed by the resistance drift. Because of a single bit change, it is possible to consider just one bit among the two bits of a cell during parity generation. This helps to reduce the ECC overhead, leading to a reduced latency for encoding and lesser storage overhead for storing parity bits. We therefore propose a new architecture which uses a reduced number of data for parity generation.

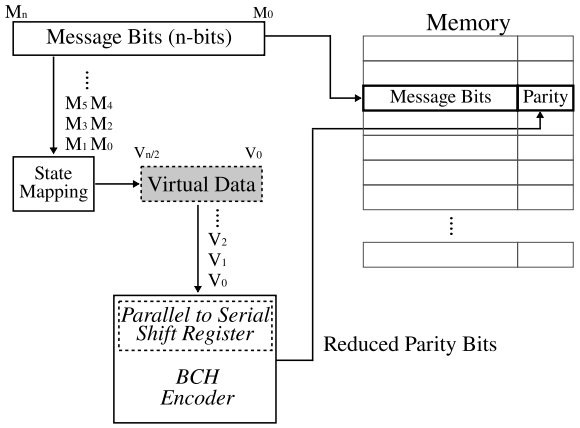

FIGURE 3. The encoder of proposed architecture.

Fig. 3 describes the encoder of the proposed architecture. It should be noted that, for the proposed error correction, any of the systematic codes which stores unaltered message bits with generated parity bits, could be used. The BCH code is considered in the shown example. In a write operation, the encoder first creates virtual data by using a simple state mapping. Each 2-bit data is converted into 1-bit virtual data then the virtual data is serially fed into a *BCH Encoder*. The length of virtual data is half of the original message size due to the 2-to-1 state mapping. The reduced parity size follows Equation 2. Once the parity generation is done, it is stored in memory along with the original message data. The details about virtual data generation are described in next subsection.

The latency overhead is negligible because the mapping table only has 4 entries and the virtual data is fed into a BCH encoder directly. Moreover, the serialized virtual data input allows skipping the parallel to serial process in a *BCH Encoder*. The virtual data in Fig. 3 is just described for clear understanding and is not required to be stored.

B. VIRTUAL DATA GENERATION

For a reduced ECC overhead, virtual data should meet the following conditions:

1) The length of virtual data should be shorter than that of the original message length because the parity size in BCH code is proportional to the size of input data of *BCH Encoder*.

2) The storage overhead for storing virtual data should be minimized for high storage density.

3) The virtual data could detect and correct errors with parity bits with the error detection and correction process.

4) Virtual data generation should be simple to minimize the performance overhead.

The virtual data generation uses a simple state mapping which is described in Table 3. The message bits are loaded by

TABLE 3. The state mapping table for virtual data generation (4LC PCM).

Storage Level	Original 2-bit Data	Virtual Data
3	10	0
2	11	1
1	01	0
0	00	1

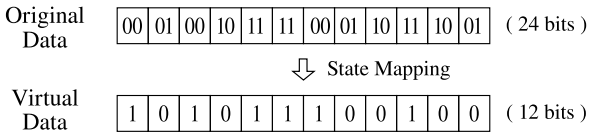

FIGURE 4. The example of virtual data generation.

2-bit and replaced by the corresponding virtual data, and then fed into *BCH Encoder* serially. Each 2-bit original data, 00, 01, 11 and 10 is substituted to each single-bit virtual data, 1, 0, 1 and 0, respectively. This state mapping can detect errors with virtual data when there is a state change by the resistance drift. For example, the original 2-bit data 01 corresponding to Level 1 represents the virtual data 0 and the parity is generated with this virtual data 0. Let's assume that an error by resistance drift occurs and the data 01 is changed to 11 (corresponding to Level 2). The virtual data for 11 is 1 and the error detection mechanism can detect the error since the parity is generated with the virtual data 0. It is the same for pattern 00 and 11. Pattern 10 for Level 3 does not cause any error. This paper only considers soft errors which is caused by the state change from the original to its adjacent state. It is not required to consider two or three level change by resistance drift because the probability of state change from Level 0 to Level 2 is zero over years since the resistance drift in Level 0 is negligible. For Level 1, the probability of two level change from Level 1 to Level 3 is also zero in a year by the Equation 4. Moreover, when we consider a periodic scrubbing operation [1], [24] which is widely considered for MLC PCM, over two level changes by resistance drift is not possible.

Fig. 4 describes an example of virtual data generation by the proposed method with 24-bit original data. By applying the state mapping, the data length of virtual data is decreased by half. The generated virtual data bits do not need to be stored in any buffer or memory since these could be regenerated during the error detection and correction process. Details are discussed in next subsection. Table 4 lists the reduction ratios of parity bits by the proposed method for 512-bit data with varying ECC capability. The required parity bits for the original BCH and the proposed method are calculated using Equation 2 and are shown in the second and third columns, respectively. The reduction ratio of parity bits by the proposed method is over 10% in all cases and this helps saving the additional cells to store parity bits. The improvement in

TABLE 4. The reduction ratio of parity bits for 512-bit data with varying ECC capability.

BCH Capability	Original Parity Size (Bits)	Proposed Parity size (Bits)	Reduction Ratio
8	73	65	10.96%
12	109	97	11.01%
16	145	129	11.03%
20	181	161	11.05%
24	217	193	11.06%

TABLE 5. The state mapping table for virtual data generation (8LC PCM).

Storage Level	Original 3-bit Data	Virtual Data
7	100	0
6	101	1
5	111	0
4	110	1
3	010	0
2	011	1
1	001	0
0	000	1

total storage density is 3.3% with BCH-24 while the parity reduction rate is about 11.1%. *Conventional 4LC* (which only uses 4LCs) needs to use BCH-24 to achieve similar SER of DRAM [24], 3.3% improvement in total storage density improvement would not be small. Moreover, since the ECC decoding latency and programming ECC parity cells also significantly affects the performance and energy efficiency, the parity reduction rate is important. It should be noted that the actual storage reduction ratio would be more increased because the proposed method can use a lighter BCH code than that of the original case, for similar SER.

The proposed virtual data generation can be extended for higher density MLC PCM like 3-bit MLC PCM (8LC) or 4-bit MLC PCM (16LC). Table 5 shows state mapping for 3-bit MLC PCM (8LC). It also uses Gray encoding for state mapping. The 6 intermediate levels from Level 1 to Level 6 are semi-amorphous state and can cause soft errors by resistance drift. If the Level 0 doesn't have not enough threshold margin due to the narrowed resistance boundary, it can cause a soft error as well. Like 2-bit MLC PCM (4LC), each 3-bit original data is alternately mapped to 1-bit virtual data 1 and 0. When an error by resistance drift occurs, the regenerated virtual data for error detection and correction will be different with the virtual data for parity generation.

C. ERROR DETECTION AND CORRECTION

The parity bits generated by the virtual data along with the stored data can detect and correct errors completely. Fig. 5

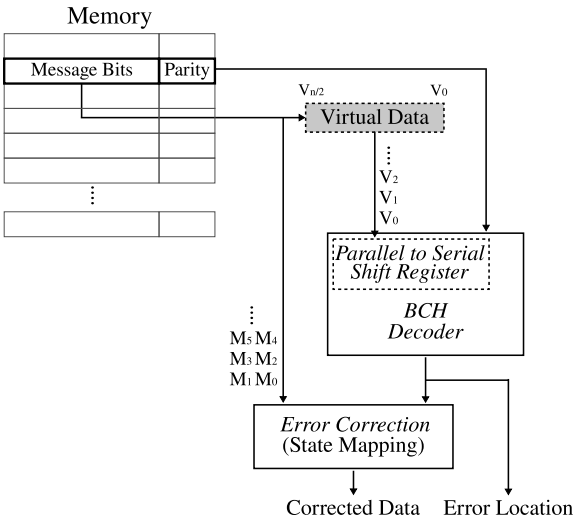

FIGURE 5. The decoder of proposed architecture.

TABLE 6. The state mapping for error correction (4LC PCM).

Error Data	Corrected Data
10	11
11	01
01	00
00	–

describes a decoder for error detection and correction by the proposed architecture. It first regenerates the virtual data using the same state mapping which is used for encoding in Table 3. Similar to the encoder, the *Parallel to Serial Shift Resister* hides the delay for state mapping, therefore there is no performance overhead by the virtual data generation. Then, *BCH Decoder* checks whether there are errors in stored data and identifies the error location.

Once errors are detected and located, a simple state mapping is used for correction and this state mapping is described in Table 6. The *Error Correction* module checks the corresponding 2-bit stored data based on the error location, then it generates the corrected data. For example, if the data in error is 01 (Level 2), it can be easily figured out that the original data is 00 because the resistance is increased from Level 0 by resistance drift. Similarly, the error data 11 and 10 leads to corrected data 01 and 11, respectively.

Fig. 6 shows an error correction example by the proposed method. The original data is 24-bit long and is stored in 12 4-level cells. By using the state mapping in Table 3, the original data is mapped into 12-bit virtual data as described in the figure. The parity is generated based on the virtual data and stored in a memory along with the original data. Let's assume that the soft errors by resistance drift occur at 1st, 8th and 10th 4LCs. The original data 00, 01 and 11 are changed to 01, 11 and 10, respectively. When data is read, the virtual data is generated from the stored data and it would be different from the virtual data generated from

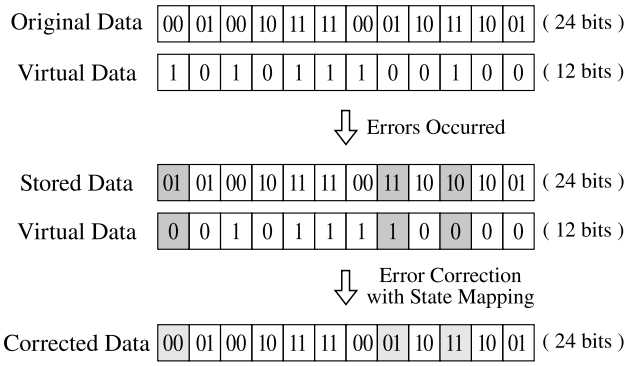

FIGURE 6. The example of error correction.

the original data. Since the parity is generated with original virtual data, the original *BCH Decoder* can detect the errors with error locations. Then, *Error Correction* in the Fig. 5 creates corrected data considering the current stored data (Error Data in Table 6). Finally, error data can be corrected by corrected data and error location.

D. SOFT ERROR RATE ANALYSIS

The reduced number of cells for storing parity bits helps to achieve higher reliability than the case which uses original BCH code (without using virtual data for parity generation). With gray encoding for 4LC PCM state mapping, there is only 1-bit change out of 2 bits stored in a single cell by a resistance drift induced soft error. In this case, the probability of uncorrectable error(s) for 4LC PCM can be calculated by Equation 4.

$$\begin{aligned} P_{error} &= 1 - P(\text{no error}) - P(\text{1-bit error}) \\ &\quad - \cdots - P(\text{n-bit error}) \\ &= 1 - \sum_{k=0}^{n} \binom{m}{k} (1 - SER_{4LC})^{m-k} (SER_{4LC})^{k} \end{aligned} \quad (4)$$

where n is the number of correctable errors by the ECC scheme considered while m is the number of 4-level cells. If a 4LC PCM architecture requires additional cells for auxiliary bits which depend on encoding methods or ECC parity, they also need to be considered for exact soft error rate calculation. SER_{4LC} represents bit error probability which is calculated by the average SERs listed in Table 2. The average SER in the table is given assuming that all 2-bit patterns (00, 01, 11 and 10) in a cache line data have the same probability of occurrence (25%). In Section V, for a more realistic scenario, we consider actual pattern distribution from real workloads from SPEC CPU2006 and PARSEC 2.1 benchmark suites to select an appropriate ECC capability. In this fashion, we can consider more practical ECC schemes for different methods.

Table 7 shows the cache line (512-bit) soft error rate with different BCH capabilities. We consider strong BCH schemes as well because the conventional 4LC PCM requires such a large ECC capability of around BCH-24 to meet the soft error rate of DRAM [13], [24]. The time listed in the first

TABLE 7. Comparison of cache line soft error rate for different ECC capability.

Time	BCH-16		BCH-24	
(s)	Original	Proposed	Original	Proposed
2^6	negligible	negligible	negligible	negligible
2^7	1.5E-12%	negligible	negligible	negligible
2^8	1.9E-10%	negligible	negligible	negligible
2^9	3.5E-08%	6.2E-12%	negligible	negligible
2^{10}	3.3E-06%	9.4E-10%	negligible	negligible
2^{11}	1.6E-04%	7.1E-08%	2.0E-09%	negligible
2^{12}	4.5E-03%	3.2E-06%	4.0E-07%	1.3E-11%

column indicates the scrubbing period required for reliable memory operation. The probabilities of uncorrectable errors with different time for conventional MLC PCM which uses an original BCH code and the proposed MLC PCM architecture are shown in the table separately. In [24], when we consider a 2GB physical PCM bank, the memory scrubbing is required within every 9.65 seconds for correct memory operation. The performance overhead by memory scrubbing could be lesser than 1% with the scrubbing period of 2^{10} seconds. The cache line SER of DRAM at time 2^{10} second is approximately 3.6×10^{-9} ($25 \times 10^{-12} \sim 75 \times 10^{-12}$ per bit-hour) [41] thus 4LC PCM with an original BCH code requires a correction capability of around BCH-24 to have a similar SER of DRAM. On the other hand, the proposed architecture allows to use an ECC of BCH-16 to achieve a similar SER. The reliability improvement is approximately 10^4 times compared to the conventional MLC PCM.

V. EVALUATION

The proposed method minimizes the additional cells required for storing parity bits and this results in improvement of both reliability and storage density of MLC PCM. The performance and energy efficiency are also improved due to the use of a light ECC scheme and reduced number of cells for programming. It should be noted that the proposed architecture can be incorporated with any other techniques using linear cyclic codes such as BCH code, to reduce the ECC overhead and further improve the reliability, storage density and performance.

To evaluate the impact of the proposed cost-effective reliable MLC PCM architecture, we perform a system-level simulation using gem5 CPU simulator [42]. For this, a 4-level cell PCM with BCH codes is considered. 18 benchmarks (6 for integer and 12 for floating point) from SPEC CPU2006 benchmark suite [43] and 12 benchmarks from PARSEC 2.1 benchmark suite [44] are used for evaluation. The system clock frequency is set as 2GHz, the L1 cache sizes are configured as 64KB for both instruction and data caches. The L2 cache size is set to 4MB. For main memory, 8GB 4LC PCM is considered with NVMain 2.0 [45]. For each benchmark, 1 billion instructions are simulated with a fast forwarding of 100 million instructions for initialization. Other state-of-the-art error correction techniques are not considered for our evaluation because the proposed method is not a part of error correction related works. The proposed architecture improves the reliability of PCM along with the performance enhancement, with a simple architectural change and not an algorithmic modification in the error correction code. To evaluate the impact of our proposed architecture, 3 different methods for MLC PCM reliability enhancement are considered along with the conventional 4LC method. The first one is an encoding technique, called *Helmet* [8] which flips and rotates data to reduce the frequency of error-prone pattern thus improving the MLC PCM reliability. Another encoding method, referred as *Mapping* [15], which also uses pattern remapping to improve write energy efficiency and reliability, is also considered. The last one is the method called *HPCM* [13] which uses both single-level cell PCM and multi-level cell PCM in a single memory array thus trading memory density to improve reliability. The conventional MLC PCM architecture with an original BCH codes and the proposed architecture are applied to each method and various aspects are analyzed and compared.

A. RELIABILITY

According to Equation 4, the error rate which represents the reliability of MLC PCM depends on three key factors, the bit error probability (SER_{4LC}), the ECC capability and the number of multi-level cells. To evaluate the impact of the proposed architecture, the key factors need to be considered. From Table 2, the bit error probability, SER_{4LC}, depends on the frequencies of the patterns which correspond to two intermediate levels (Level 1 and Level 2). We use actual pattern distributions from SPEC and PARSEC benchmarks for calculating SER_{4LC}. Fig. 7 shows the pattern distributions in the actual cache line data for 1 billion instructions of each benchmark. The average pattern frequencies for Level 1 and Level 2 are 8.82% and 11.32% with the maximum of 27.35% and 33.25%, respectively. Each method uses different BCH capabilities to have a similar SER of DRAM (i.e. 3.6×10^{-9} at time 2^{10} seconds). For the number of multi-level cells, the additional cells for parity bits or auxiliary bits (for encoding methods) are also considered.

Table 8 shows cache line SER with different BCH capabilities. For the bit error probability (SER_{4LC}) calculation, maximum pattern frequencies of Level 1 and Level 2 are used to ensure the reliability for all workloads. To meet the similar SER of DRAM, *Conventional 4LC* (which only uses 4-level cells) requires BCH-24 while *Helmet*, *Mapping* and *HPCM* require BCH-19, BCH-17 and BCH-12, respectively. The cache line SER of *Conventional 4LC* with original BCH-24 is 1.75×10^{-9}% at time 2^{10} seconds. On the other hand, the proposed architecture achieves the SER below 10^{-13} (described as ''negligible'' in the table). The reliability improvement is approximately 10^4 times and this allows to use a lighter ECC scheme leading to an

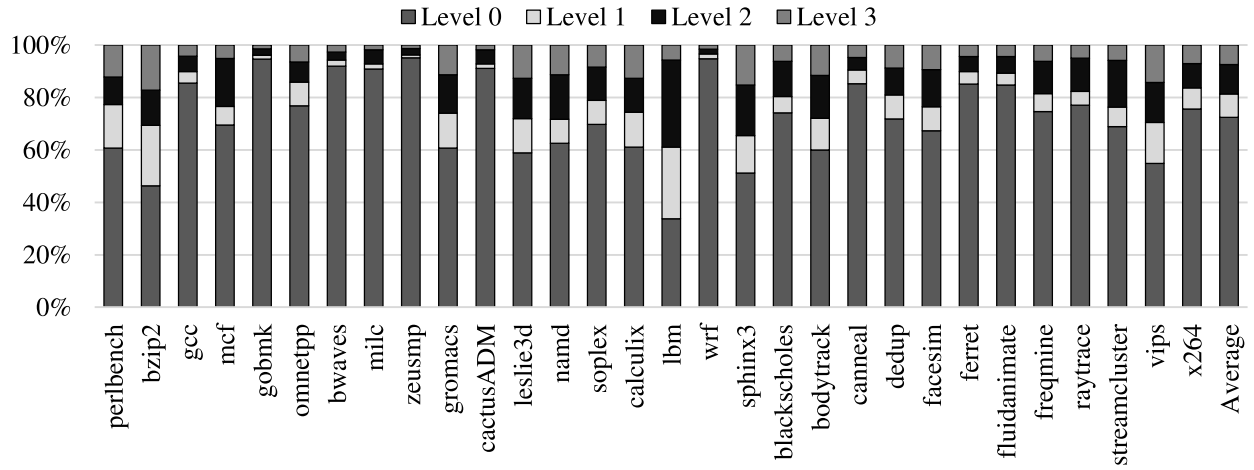

FIGURE 7. The pattern frequency of real workload (SPEC 2006 and PARSEC 2.1).

TABLE 8. The impact of cache line SER by the proposed method.

Time (s)	Conventional 4LC		
	Original BCH	Proposed Method	
	BCH-24	BCH-24	BCH-18
2^9	1.45E-12%	negligible	8.63E-12%
2^{10}	1.75E-09%	negligible	1.87E-09%
2^{11}	5.49E-07%	1.77E-11%	2.03E-07%
Time (s)	Helmet [8]		
	Original BCH	Proposed Method	
	BCH-19	BCH-19	BCH-15
2^9	7.72E-12%	negligible	1.52E-11%
2^{10}	2.27E-09%	negligible	1.59E-09%
2^{11}	2.90E-07%	5.27E-11%	9.47E-08%
Time (s)	Mapping [15]		
	Original BCH	Proposed Method	
	BCH-17	BCH-17	BCH-13
2^9	7.80E-12%	negligible	5.24E-11%
2^{10}	1.09E-09%	negligible	3.08E-09%
2^{11}	9.81E-08%	2.82E-11%	1.18E-07%
Time (s)	HPCM [13]		
	Original BCH	Proposed Method	
	BCH-12	BCH-12	BCH-9
2^9	7.48E-11%	negligible	3.23E-11%
2^{10}	3.45E-09%	2.71E-12%	8.92E-10%
2^{11}	1.06E-07%	8.05E-11%	1.77E-08%

improved performance and energy efficiency. With the proposed method, the *Conventional 4LC* can achieve a similar SER with BCH-18. The data encoding methods *Helmet* and *Mapping* reduce the number of drift-prone pattern (Level 2) to improve reliability. The average pattern frequencies for

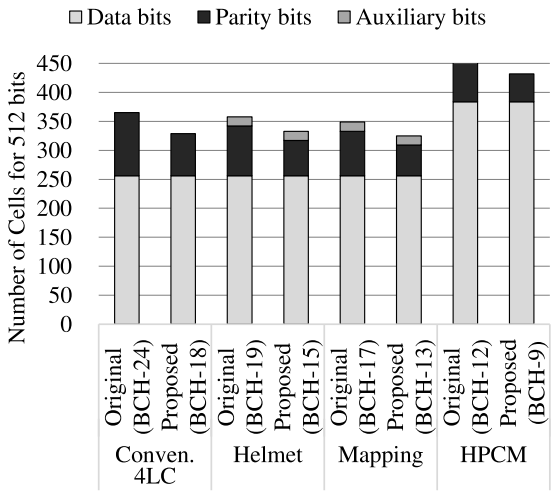

FIGURE 8. The comparison of storage density by proposed method.

Level 1 and Level 2 are decreased to 10.6% and 6.04% by *Helmet*. The reduced pattern frequencies by *Mapping* are 6.26% and 5.24% for Level 1 and Level 2, respectively. With a reduced number of drift-prone patterns, *Helmet* and *Mapping* can use lighter BCH codes (BCH-19 and BCH-17) compared to that (BCH-24) in *Conventional 4LC*. *HPCM* can use much lighter BCH code, BCH-12, for a similar to DRAM SER because this architecture significantly reduces the number of 4-level cells by using a heterogeneous memory array. When the proposed architecture is integrated with each method, BCH-19, BCH-17 and BCH-12 can be replaced by BCH-15, BCH-13 and BCH-9, respectively, while ensuring the similar level of reliability.

B. STORAGE DENSITY

The storage density (data bits/cell) of MLC PCM is given by the number of cells required for data bits, parity bits and the auxiliary bits. For correct decoding, auxiliary bits are required for the encoding based reliability enhancement methods such as *Helmet* and *Mapping*. Fig. 8 shows the

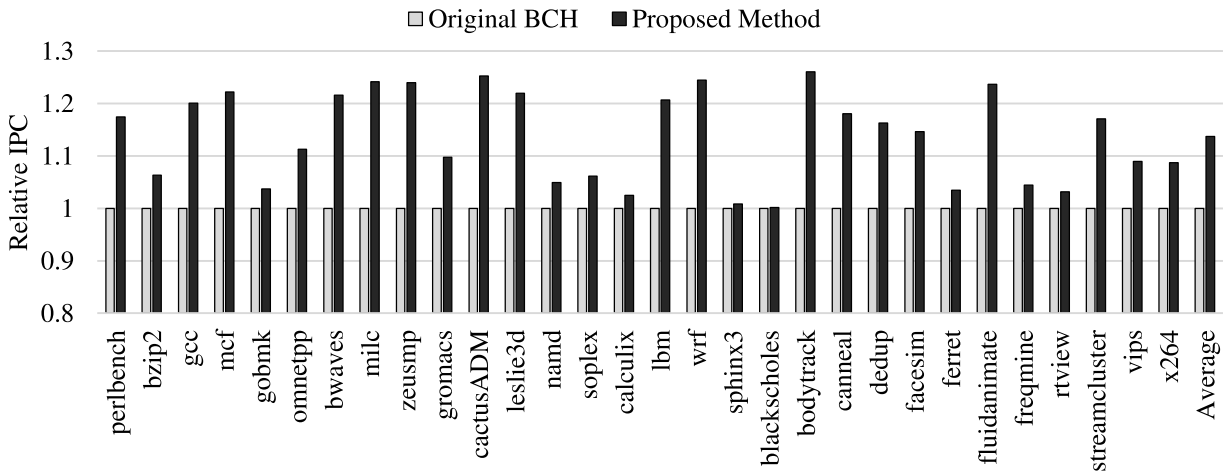

FIGURE 9. The performance (relative IPC) improvement for Conventional 4LC.

number of cells required to store a single cache line data (512-bits) for various methods. *Conventional* 4*LC*, *Helmet* and *Mapping* methods require 256 cells to store 512 message bits (i.e. actual data bits). On the other hand, *HPCM* requires relatively larger number of cells (=384) to store message bits because it includes 256 single-level cells and 128 multi-level cells. For ECC schemes, different BCH capabilities are considered such that all the methods can have SER similar to DRAM and the number of cells for parity bits are calculated using Equation 2. The encoding block size is 32-bit for both *Helmet* and *Mapping* and the additional cells for auxiliary bits are also considered for the storage density comparison.

In Fig. 8, *Conventional* 4*LC* with the original BCH code requires total 365 cells while the proposed method requires only 329 cells, and this improves the storage density from 1.40 to 1.56. Similarly, the proposed architecture also reduces the parity overhead for other methods. The total number of cells for *Helmet*, *Mapping* and *HPCM* are reduced from 358, 349 and 458 to 333, 325 and 432, respectively. The improvement in storage density is 7.9% on average with the maximum of 10.9% for *Conventional* 4*LC*.

C. PERFORMANCE AND ENERGY EFFICIENCY

The proposed cost-effective reliable architecture can improve not only the performance but also the energy efficiency since it reduces the ECC latency overhead and the number of cells for storing parity bits. To evaluate the performance improvement by the proposed architecture, we compare the Instructions-per-cycle (IPC) values for all 30 benchmarks from SPEC CPU2006 and PARSEC 2.1 benchmark suites. Since each method considers different BCH capability and different number of cells (different parity size and auxiliary bits for encoding), we vary read and write programming latencies in our evaluation. The base read latency is 120*ns* while base programming latencies for each level are 150*ns*, 370*ns*, 370*ns* and 40*ns*, respectively [15]. Based on these base latencies, we calculate the access and programming latencies for each method. The read latency is calculated by considering both the access latency and BCH decoding latency. BCH decoding latency is calculated by the actual measured latency [46] and scaling factor [47]. Since the proposed method reduces BCH decoding latency (due to a reduction in BCH capability), the access time of the proposed ECC-protected memory is much smaller than that of the conventional ECC-protected memory. Different latencies for each method are applied by modifying parameters in NVMain. For each programming latency, we have considered both actual pattern distribution of each benchmark and the total number of cells to be programmed. The equation is "*level base programming latency * (average programming latency for Conventional 4LC / average programming latency for the proposed method) * (total number of cells / 256)*" while a level base programming latency is the base programming latencies for each level. We therefore consider actual pattern distribution and additional cells to be programmed by each method, for each level programming latency. Different programming latencies for each level and each method are applied to NVMain by modifying corresponding parameters.

Fig. 9 shows the performance improvement by the proposed architecture over *Conventional* 4*LC*. We compare the relative IPC (normalized to the baseline scheme with original BCH code) for the proposed architecture. To have the SER similar to DRAM, BCH-24 and BCH-18 are used for the *Conventional* 4*LC* PCM and the one with the proposed method, respectively. For all the benchmarks, an improvement in IPC is observed because of the reduced BCH latency and the number of programmed cells. The average performance improvement is 13.7% while the maximum improvement is 26.0% for *bodytrack* benchmark. The benchmarks which have a large number of read and write operations such as *bodytrack* and *cactusADM* show larger improvements in performance compared to the other benchmarks.

Other methods also show a significant performance improvement with the proposed architecture as shown

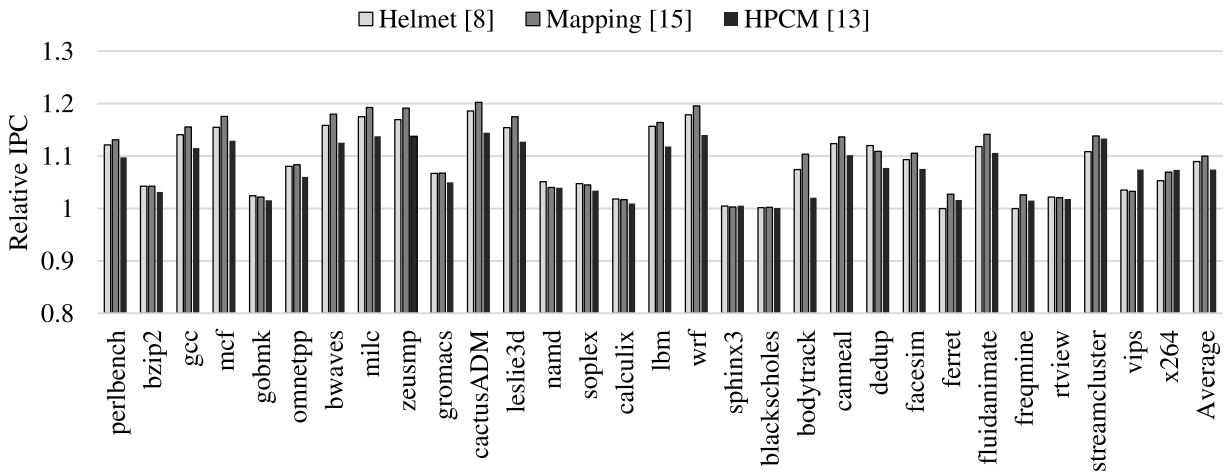

FIGURE 10. The performance (relative IPC) improvement for Helmet [8], Mapping [15] and HPCM [13].

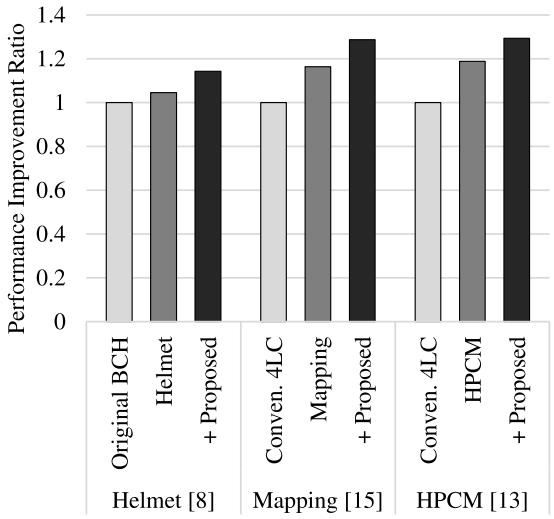

FIGURE 11. The comparison of performance improvement by the proposed architecture.

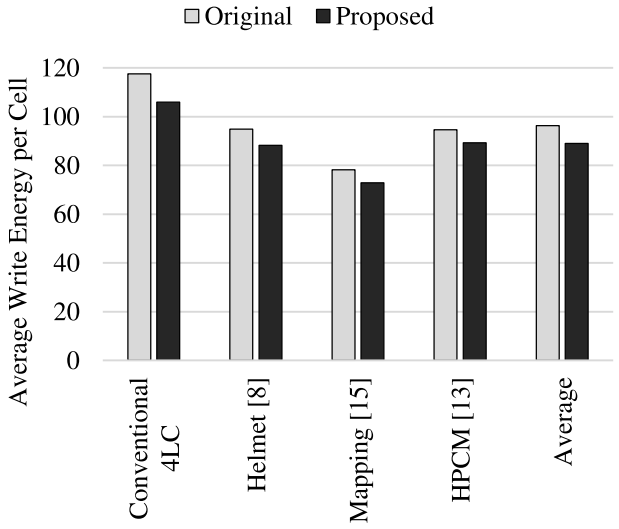

FIGURE 12. The comparison of write energy per cell.

in Fig. 10. Different BCH schemes are considered for each method based on the cache line SER in Table 8. The IPC values for each method are normalized by its reference IPC values which use original BCH code. The average performance improvements are 8.9%, 10.0% and 7.4% for *Helmet*, *Mapping* and *HPCM*, respectively. These methods show slightly lower performance improvements compared to the case of *Conventional* 4*LC*. This is because of relatively lower reduction in BCH latency overhead for these methods as compared to that of *Conventional* 4*LC*. The most notable aspect of the proposed architecture is that it can be easily incorporated with any other method to further improve the performance. Fig. 11 shows overall performance improvement for three methods compared to *Conventional* 4*LC*. *Helmet*, *Mapping* and *HPCM* show the performance improvement of 4.5%, 16.5% and 18.8%, respectively, with their encoding techniques and architectural enhancements. After applying the proposed architecture, each method can have additional performance improvement of 8.9%, 10.0% and 7.4%.

The reduced number of cells for parity bits also improves the energy efficiency. The additional cells for parity bits occupies approximately 29.9% of the total number of cells (256 cells for message bits and 109 cells for parity bits) for *Conventional* 4*LC* with BCH-24 and this is a considerable overhead. It significantly degrades the write energy efficiency. The additional cell ratio can be decreased to 22.2% (256 cells for message bits and 73 cells for parity bits) by applying the proposed architecture. Fig. 12 describes the average write energy for programming a single cell for different method. We considers the energy consumptions of $36pJ$, $307pJ$, $547pJ$ and $20pJ$ for Level 0, Level 1, Level 2 and Level 3, respectively [15]. We can calculate a relative per-bit write energy considering the pattern distribution and the number of cells. The average per-bit write energy is approximately $117.6pJ$ for *Conventional* 4*LC*. After applying the proposed architecture it is decreased to $106.0pJ$. The improvement is around 10%. *Helment* and *Mapping* has $94.9pJ$ and $78.2pJ$ per-bit write energies, which is lower than

Conventional 4LC. This is because they decrease the number of intermediate level programmings which consume more energy than other levels. *HPCM* also has reduced the per-bit write energy of $94.6pJ$ by using a number of single-level cells with higher energy efficiency. With the proposed architecture, the energy values are reduced to $88.3pJ$, $72.86pJ$ and $89.2pJ$, respectively.

VI. CONCLUSION

MLC PCM would suffer from reduced reliability due to the resistance drift phenomenon which can cause soft errors. This requires a strong error correction scheme which can handle multi-bit errors. This can significantly degrade the storage density, performance and energy efficiency. To mitigate the ECC overhead, this paper proposes a cost-effective reliable MLC PCM architecture which uses virtual data for parity generation. The reliability and storage density are improved because the data size of virtual data is just half of the original message size. The reliability is improved by approximately 10^4 times with the storage density improvement of 10.9% compared to the conventional MLC PCM. The performance and energy efficiency are also significantly improved due to the reduced ECC latency overhead and reduced number of parity bits. The performance and energy efficiency are improved up to 13.7% and 10.0%, respectively. The proposed architecture can be easily incorporated with other techniques which use linear cyclic codes like BCH code for further improvements in reliability, performance and energy efficiency.

REFERENCES

[1] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan, "Efficient scrub mechanisms for error-prone emerging memories," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.*, Feb. 2012, pp. 1–12.

[2] L. Jiang, Y. Zhang, and J. Yang, "Mitigating write disturbance in super-dense phase change memories," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 216–227.

[3] M. Asadinia, M. Arjomand, and H. Sarbazi-Azad, "Variable resistance spectrum assignment in phase change memory systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 11, pp. 2657–2670, Nov. 2015.

[4] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y. Chen, R. M. Shelby, M. Salinga, D. Krebs, S. Chen, H. Lung, and C. H. Lam, "Phase-change random access memory: A scalable technology," *IBM J. Res. Develop.*, vol. 52, nos. 4–5, pp. 465–479, Jul. 2008.

[5] Y. Xie, "Modeling, architecture, and applications for emerging memory technologies," *IEEE Design Test Comput.*, vol. 28, no. 1, pp. 44–51, Jan. 2011.

[6] B. C. Lee, P. Zhou, J. Yang, Y. Zhang, B. Zhao, E. Ipek, O. Mutlu, and D. Burger, "Phase-change technology and the future of main memory," *IEEE Micro*, vol. 30, no. 1, p. 143, Jan. 2010.

[7] A. Redaelli, A. Pirovano, F. Pellizzer, A. L. Lacaita, D. Ielmini, and R. Bez, "Electronic switching effect and phase-change transition in chalcogenide materials," *IEEE Electron Device Lett.*, vol. 25, no. 10, pp. 684–686, Oct. 2004.

[8] W. Zhang and T. Li, "Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2011, pp. 197–208.

[9] N. Papandreou, H. Pozidis, T. Mittelholzer, G. F. Close, M. Breitwisch, C. Lam, and E. Eleftheriou, "Drift-tolerant multilevel phase-change memory," in *Proc. 3rd IEEE Int. Memory Workshop (IMW)*, May 2011, pp. 1–4.

[10] Q. Wu, F. Sun, W. Xu, and T. Zhang, "Using multilevel phase change memory to build data storage: A time-aware system design perspective," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2083–2095, Oct. 2013.

[11] M. K. Qureshi, M. M. Franceschini, L. A. Lastras-Monta no, and J. P. Karidis, "Morphable memory system: A robust architecture for exploiting multi-level phase change memories," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 153–162, Jun. 2010, doi: 10.1145/1816038.1815981.

[12] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in MLC phase change memory," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.*, Feb. 2012, pp. 1–10.

[13] T. Kwon, M. Imran, J. M. You, and J. Yang, "Heterogeneous PCM array architecture for reliability, performance and lifetime enhancement," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1610–1615.

[14] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve pram write performance, energy and endurance," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, New York, NY, USA, Dec. 2009, pp. 347–357, doi: 10.1145/1669112.1669157.

[15] J. Wang, X. Dong, G. Sun, D. Niu, and Y. Xie, "Energy-efficient multi-level cell phase-change memory system with data encoding," in *Proc. IEEE 29th Int. Conf. Comput. Design (ICCD)*, Oct. 2011, pp. 175–182.

[16] M. Joshi, W. Zhang, and T. Li, "Mercury: A fast and energy-efficient multi-level cell based phase change memory system," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 345–356.

[17] S. Swami and K. Mohanram, "Reliable nonvolatile memories: Techniques and measures," *IEEE Design Test*, vol. 34, no. 3, pp. 31–41, Jun. 2017.

[18] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "SD-PCM: Constructing reliable super dense phase change memory under write disturbance," *SIGPLAN Not.*, vol. 50, no. 4, pp. 19–31, Mar. 2015, doi: 10.1145/2775054.2694352.

[19] S. Kannan, J. Rajendran, R. Karri, and O. Sinanoglu, "Sneak-path testing of crossbar-based nonvolatile random access memories," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, pp. 413–426, May 2013.

[20] M. G. Mohammad, "Fault model and test procedure for phase change memory," *IET Comput. Digit. Tech.*, vol. 5, no. 4, , pp. 263–270, 2011.

[21] G. W. Burr, M. J. BrightSky, A. Sebastian, H.-Y. Cheng, J.-Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H.-L. Lung, H. Pozidis, E. Eleftheriou, and C. H. Lam, "Recent progress in phase-change memory technology," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 6, no. 2, pp. 146–162, Jun. 2016.

[22] R. Wang, Y. Zhang, and J. Yang, "ReadDuo: Constructing reliable MLC phase change memory through fast and robust readout," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2016, pp. 203–214.

[23] 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. (VLSI) Syst.*, vol. 19, no. 8, pp. 1357–1367, Aug. 2011.

[24] N. H. Seong, S. Yeo, and H.-H.-S. Lee, "Tri-level-cell phase change memory: Toward an efficient and reliable memory system," in *Proc. 40th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2013, pp. 440–451, doi: 10.1145/2485922.2485960.

[25] C. Wilkerson, A. R. Alameldeen, Z. Chishti, W. Wu, D. Somasekhar, and S.-L. Lu, "Reducing cache power with low-cost, multi-bit error-correcting codes," in *Proc. 37th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2010, pp. 83–93, doi: 10.1145/1815961.1815973.

[26] D. Strukov, "The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories," in *Proc. 40th Asilomar Conf. Signals, Syst. Comput.*, Oct. 2006, pp. 1183–1187.

[27] E. Aliagha, A. M. H. Monazzah, and H. Farbeh, "REACT: Read/write error rate aware coding technique for emerging STT-MRAM caches," *IEEE Trans. Magn.*, vol. 55, no. 5, pp. 1–8, May 2019.

[28] S. Asadi, A. M. H. Monazzah, H. Farbeh, and S. G. Miremadi, "WIPE: Wearout Informed Pattern Elimination to Improve the Endurance of NVM-based Caches," in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 188–193.

[29] A. R. Alameldeen and D. A. Wood, "Frequent pattern compression: A significance-based compression scheme for L2 caches," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1500, 2004.

[30] B. Wu, B. Zhang, Y. Cheng, Y. Wang, D. Liu, and W. Zhao, "An adaptive thermal-aware ECC scheme for reliable STT-MRAM LLC design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1851–1860, Aug. 2019.

[31] E. Cheshmikhani, H. Farbeh, and H. Asadi, "Enhancing reliability of STT-MRAM caches by eliminating read disturbance accumulation," in *Proc. Design, Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2019, pp. 854–859.

[32] E. Cheshmikhani, H. Farbeh, and H. Asadi, "ROBIN: Incremental oblique interleaved ECC for reliability improvement in STT-MRAM caches," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, 2019, pp. 173–178.
[33] Z. Azad, H. Farbeh, and A. M. H. Monazzah, "Orient: Organized interleaved ECCS for new STT-MRAM caches," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2018, pp. 1187–1190.
[34] C.-S. Gong, Y.-C. Chang, L.-R. Huang, C.-J. Yang, K.-M. Ji, K.-L. Lu, and J.-C. Liou, "Two dimensional parity check with variable length error detection code for the non-volatile memory of smart data," *Appl. Sci.*, vol. 8, no. 8, p. 1211, Jul. 2018.
[35] S.-K. Lu, H.-P. Li, and K. Miyase, "Progressive ECC techniques for phase change memory," in *Proc. IEEE 27th Asian Test Symp. (ATS)*, Oct. 2018, pp. 161–166.
[36] Z. Azad, H. Farbeh, A. M. H. Monazzah, and S. G. Miremadi, "AWARE: Adaptive way allocation for reconfigurable ECCs to protect write errors in STT-RAM caches," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 3, pp. 481–492, Jul. 2019.
[37] 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.
[38] A. Das and N. A. Touba, "Efficient one-step decodable limited magnitude error correcting codes for multilevel cell main memories," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 575–583, 2019.
[39] 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.
[40] 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.
[41] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM errors in the wild: A large-scale field study," in *Proc. 11th Int. Joint Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*, New York, NY, USA, 2009, pp. 193–204, doi: 10.1145/1555349.1555372.
[42] N. Binkert, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, D. A. Wood, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, and T. Krishna, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, , pp. 1–7, Aug. 2011.
[43] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.
[44] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. ACM 17th Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, New York, NY, USA, 2008, pp. 72–81, doi: 10.1145/1454115.1454128.
[45] M. Poremba, T. Zhang, and Y. Xie, "NVMain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE Comput. Arch. Lett.*, vol. 14, no. 2, pp. 140–143, Jul. 2015.
[46] Y. Lee, H. Yoo, I. Yoo, and I.-C. Park, "6.4 Gb/s multi-threaded BCH encoder and decoder for multi-channel SSD controllers," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2012, pp. 426–428.
[47] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, Jun. 2017.

TAEHYUN KWON (Student Member, IEEE) received the B.S. degree from Kyung Hee University, South Korea, in 2005, and the M.S. degree from Yonsei University, South Korea, in 2007, all in electrical engineering. He is currently pursuing the Ph.D. degree with the Department of Semiconductor and Display Engineering, Sungkyunkwan University, South Korea. He has been with the SoC Design Team, Samsung Electronics, South Korea. He is a part of the team developing high-performance CPU and GPU for Samsung mobile SoC products. His research interests are high-performance processor architecture, low-power CPU and GPU implementations, high-bandwidth memory, and reliable architectures for emerging memory technologies.

MUHAMMAD IMRAN (Student Member, IEEE) received the B.S. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2012. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, South Korea. He joined Sungkyunkwan University, in 2016. His research interests include very large-scale integration design and reliable architectures for the emerging memory technologies. He was a recipient of scholarship by the Higher Education Commission of Pakistan for the M.S. and Ph.D. studies.

JOON-SUNG YANG (Senior Member, IEEE) received the B.S. degree from Yonsei University, Seoul, South Korea, in 2003, and the M.S. and Ph.D. degrees from The University of Texas at Austin, Austin, TX, USA, in 2007 and 2009, respectively, all in electrical and computer engineering. He was with Intel Corporation and Sungkyunkwan University. He is currently an Associate Professor with Yonsei University. His research interests are memory architectures and efficient deep learning architecture development. He was a recipient of the Korea Science and Engineering Foundation (KOSEF) Scholarship, in 2005. He received the Best Paper Award at the 2008 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems and the 2016 IEEE International SoC Design Conference, and was nominated for the Best Paper Award at the 2013 IEEE VLSI Test Symposium.