electronics

MDPI

Article

Error-Vulnerable Pattern-Aware Binary-to-Ternary Data Mapping for Improving Storage Density of 3LC Phase Change Memory

Jeong Beom Hong [1], Young Sik Lee [2], Yong Wook Kim [2] and Tae Hee Han [3,*]

[1] Department of Semiconductor and Display Engineering, College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16410, Korea; adfffsa@skku.edu
[2] Department of Electrical and Computer Engineering, College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16410, Korea; lty0107@skku.edu (Y.S.L.); coail@skku.edu (Y.W.K.)
[3] Department of Artificial Intelligence, Sungkyunkwan University, Suwon 16410, Korea
* Correspondence: than@skku.edu; Tel.: +82-31-299-4587

Received: 28 February 2020; Accepted: 7 April 2020; Published: 9 April 2020

Abstract: Multi-level cell (MLC) phase-change memory (PCM) is an attractive solution for next-generation memory that is composed of resistance-based nonvolatile devices. MLC PCM is superior to dynamic random-access memory (DRAM) with regard to scalability and leakage power. Therefore, various studies have focused on the feasibility of MLC PCM-based main memory. The key challenges in replacing DRAM with MLC PCM are low reliability, limited lifetime, and long write latency, which are predominantly affected by the most error-vulnerable data pattern. Based on the physical characteristics of the PCM, where the reliability depends on the data pattern, a tri-level-cell (3LC) PCM has significantly higher performance and lifetime than a four-level-cell (4LC) PCM. However, a storage density is limited by binary-to-ternary data mapping. This paper introduces error-vulnerable pattern-aware binary-to-ternary data mapping utilizing 3LC PCM without an error-correction code (ECC) to enhance the storage density. To mitigate the storage density loss caused by the 3LC PCM, a two-way encoding is applied. The performance degradation is minimized through parallel encoding. The experimental results demonstrate that the proposed method improves the storage density by 17.9%. Additionally, the lifetime and performance are enhanced by 36.1% and 38.8%, respectively, compared with those of a 4LC PCM with an ECC.

Keywords: encoding; storage density; multi-level-cell; phase-change memory; reliability; tri-level-cell

1. Introduction

Multi-level-cell (MLC) phase-change memory (PCM) has the potential to replace dynamic random-access memory (DRAM) in the main memory owing to its excellent device down-scaling and standby power consumption characteristics [1,2]. In addition, MLC PCM can be useful for the massive tensor product operations required by deep neural networks (DNN) that utilize different resistive cross-point arrays as synaptic devices, that is, a processing-in-memory (PIM) architecture [3,4]. The superior feature of PCM over other emerging non-volatile memories, such as ReRAM and STT-MRAM, is that MLC technology using a broad resistance spectrum efficiently doubles or triples the storage density and thus makes it possible to meet the increasing demands on the main memory capacity in today's big data era [5,6]. The MLC PCM stores multiple bits in a single cell using the wide variable-resistance characteristic of the GST ($Ge_2Se_2Te_5$) material, which changes its physical state when a current pulse is applied. The typical resistance spectrum ranges from 1 kΩ in the fully crystalline state to 1 MΩ in the fully amorphous state. Research has been underway for over a decade to use

MLC PCM as the main memory or a storage class memory (SCM). Indeed, various methods have been introduced to use the MLC technology effectively when targeting PCM as the main memory [7–15].

For reliable data retention, and for coping with the reduced sensing margin, iterative write-and-verify (read) operations must be executed for each memory cell in the MLC PCM, which deteriorates its lifetime and performance. In addition, MLC PCM suffers from a significantly higher soft error rate (SER) when compared to DRAM. This is mainly caused by the resistance drift phenomenon that blurs the sensing boundary between adjacent states. The use of error-correction code (ECC) is a common technique in reducing the SER of MLC PCM; however, it adversely affects the performance and area efficiency.

Scrubbing can help diminish the SER by reading, detecting, and correcting errors by utilizing the parity bit check and writing them back into the cell [16]. The shorter the scrubbing period, the lower the SER. However, frequent scrubbing accelerates the wearing out of PCM. In [10], the authors proposed an efficient scrubbing mechanism that could flexibly determine the scrubbing period considering the error-vulnerable patterns. In addition, recent studies focused on designing reliable 4LC PCM to mitigate the overhead caused by ECC and scrubbing, by expelling the most error-vulnerable pattern [7–12].

Obviously, long ECC decoding latency is a significant bottleneck in achieving a high-performance memory system [17]. Moreover, complicated ECC algorithms are unacceptable in the MLC PCM based main memory considering its longer write latency than DRAM. A four-level cell (4LC) PCM, which stores two bits in one cell, exhibits an average SER of 6.74×10^{-5}, and requires the adoption of a 16-bit error-correcting Bose–Chaudhuri–Hocquenghem (BCH) code to achieve DRAM-compatible reliability. In addition, at least $[t \times ceil\left(\log_2 k\right) + 1]$ parity bits are required to correct t-bit errors in k-bit write data in the BCH code. For example, if a BCH-16 code is used for every 512 bits of data, then a storage overhead of 160 parity bits is needed. Meanwhile, the storage density (bits/cell) of the 4LC PCM decreases from 200% to 152%.

Compared to 4LC PCM, a tri-level-cell (3LC) approach is advantageous in terms of error-resilience, can sharply reduce the ECC and scrubbing overhead; the 3LC PCM is superior to 4LC in terms of latency and lifetime, the significant factors of the main memory. Thus, recent studies have attempted to deploy 3LC PCM as a promising alternative to DRAM as the main memory. The motivations of this trial are based on the observation that 3LC PCM can show an improved storage density compared to 4LC PCM with ECC in the same SER environment [7,9,14,18–21]. However, the existing 3LC PCM based architectures have a problem that the storage density does not approach the theoretical maximum of 160% due to an additional data conversion from binary to ternary. From this motif, several studies proposed 3LC PCM specific mapping scheme to improve the storage density [7,9,21]. To summarize, the key problems in designing MLC PCM based main memory are as follows:

1. Data patterns that use intermediate resistance levels in the MLC PCM require a larger number of write-and-verify operations, resulting in a reduction in the lifetime and performance.
2. The main factor that increases the SER of the MLC PCM is the resistance drift. Hence, ECC and scrubbing are necessary to sustain the required reliability. However, a complex ECC deteriorates the storage density and performance, and frequent scrubbing degrades the lifetime.
3. The lifetime and reliability of MLC PCM are highly dependent on the most error-vulnerable pattern. Various studies have recently attempted to resolve this problem by employing 3LC PCM. However, the problem of reduced storage density compared to the 4LC PCM should be addressed further.

In this paper, we propose an error-vulnerable pattern-aware parallel data encoding method to address the storage density reduction associated with 3LC PCM-based main memory. The proposed method applies two-way encoding schemes according to the data patterns for writing data in the 3LC PCM and focuses on enhancing the storage density by maximizing cell utilization.

The remainder of this paper is organized as follows: We review the background of MLC PCM in Section 2. The detailed encoding and decoding methods are described in Section 3. We discuss the experimental results and comparisons in Section 4. Section 5 concludes this manuscript.

2. Background

2.1. MLC PCM

MLC PCM stores multiple bits per cell by dividing the wide resistance spectrum of a GST material. As the proportion of amorphous states in the PCM cell increases, the resistance value increases. During data writing operations, RESET generates a short and high-power current pulse to form an amorphous state with high resistance, and SET produces a crystalline state with low resistance by applying low power for a relatively long time [22,23].

In the case of a single-level-cell (SLC), 1-bit data can utilize the resistance values that differ by approximately 1000 times. Therefore, repeated writing operations are not necessary for data retention. However, as the number of levels in a PCM cell increases to store multiple bits, the number of required write-and-verify operations increases drastically, because the probability of causing errors sharply increases by interfering the resistance range of adjacent data values.

Accordingly, iterative write, read, and compare operations should be performed until the cell state falls within the desired resistance range in MLC PCM. This is similar to the incremental-step-pulse-programming (ISPP) technique used in MLC NAND flash memory [24]. To form the intermediate states of the 4LC PCM, the writing circuitry repeatedly generates the RESET current followed by the SET current as shown in Figure 1. For example, the actual write count in the 4LC PCM increases by 5.5 times on an average compared to that of the SLC PCM. This not only causes an increase in the write latency and energy consumption but also a reduction of the lifetime by 80% of the original.

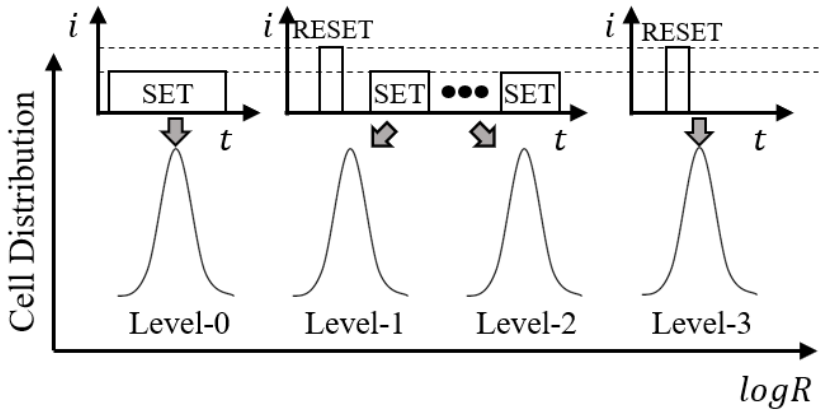

Figure 1. Iterative write operations in 4LC PCM.

2.2. Resistance Drift

Unlike DRAM, where mostly random single-bit errors occur, multi-bit errors caused by the resistance drift occur frequently in MLC PCM. In the case of 4LC PCM, cells with the most error-vulnerable pattern tend to be corrupted after just a few seconds [25,26]. The resistance drift of the MLC PCM depends on the resistance value after memory writing, which is characterized by Equation (1) [7].

$$R = R_0\left(\frac{t}{t_0}\right)^{\nu}, \tag{1}$$

where R_0 and t_0 are normalization constants, t is the time that has passed since the data has been written, and ν is the resistance drift coefficient. The resistance of the 4LC PCM cell increases exponentially with

time, and the resistance drift speed exhibits different characteristics depending on the data patterns, as shown in Table 1.

Table 1. Resistance drift parameters of 4LC PCM.

Level	Pattern	$\log R_0$		ν	
		μ_{R_0}	σ_{R_0}	μ_ν	σ_ν
0	00	3	$\frac{1}{6}$	0.01	0.004
1	01	4		0.02	0.008
2	11	5		0.06	0.024
3	10	6		0.1	0.04

The use of gray coded data patterns is common in MLC PCMs because it maintains a constant single bit difference between adjacent states. Compared to Level-0 and Level-3 data patterns, which use the smallest and largest resistance ranges, respectively, Level-1 and Level-2 patterns are much more likely to be impaired by resistance drift speed, as shown in Figure 2. In particular, the most error-vulnerable pattern (Level-2) suffers from four times more SER than the second most vulnerable pattern (Level-1) [8].

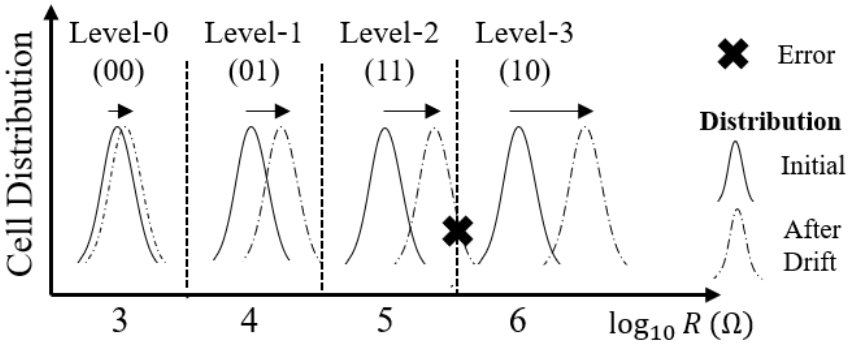

Figure 2. The highest error occurrence probability in Level-2 pattern by the resistance drift.

Several recent studies have suggested the allocation of a wider range of resistance to the most error-vulnerable data pattern in the 4LC PCM [13,27]. Nevertheless, scrubbing would still be inevitable. However, PCM with a maximum write count of 10^8 would limit the number of scrubbings, when compared to the DRAM with an almost infinite lifetime [28].

2.3. SER Analysis

To calculate the SER of the 4LC PCM, we used the analytical formula presented in [7]. In the MLC PCM, the initial resistance value of each cell, R_0, follows a Gaussian distribution as a result of the iterative write operations. With the drift parameters in Table 1, and the scrubbing period for t in Equation (1), the condition under which a soft error occurs is given by $R(t) > 10^{\mu_{R_0}+3\sigma_{R_0}}$, where the resistance of a PCM cell increases over the sensing margin. After calculating the average SER of a single memory cell, SER_{cell}, the line error rate (LER) of a 512-bit single cache line can be obtained considering the maximum bit correction capability of the applied ECC. For example, the LER of the cache line with a t-bit correctable ECC with p parity bits can be expressed as in Equation (2).

$$LER = 1 - \sum_{i=0}^{t} \binom{256+p}{i} (1 - SER_{cell})^{256+p-i} (SER_{cell})^{i} \quad (2)$$

In Equation (2), SER_{cell} depends on the ratio of each resistance level in benchmark programs. Also, the LER of the cache line is varied by the error correction capability of ECC algorithms and the

number of parity bits. Table 2 lists the LERs of 512-bit 4LC PCM calculated by Equation (2), for various scrubbing periods and ECC algorithms.

Table 2. LERs of 512-bit 4LC PCM with various scrubbing periods and ECCs.

Scrubbing Period (s)	No ECC	BCH-8 (80 Parity Bits)	BCH-16 (160 Parity Bits)	BCH-24 (240 Parity Bits)
2^5	2.89×10^{-1}	3.80×10^{-10}	Negligible	Negligible
2^6	4.28×10^{-1}	2.64×10^{-8}	Negligible	Negligible
2^7	5.65×10^{-1}	7.45×10^{-7}	Negligible	Negligible
2^8	7.04×10^{-1}	1.54×10^{-5}	1.27×10^{-12}	Negligible
2^9	8.20×10^{-1}	2.00×10^{-4}	2.32×10^{-10}	Negligible
2^{10}	9.04×10^{-1}	1.80×10^{-3}	2.15×10^{-8}	2.81×10^{-14}
2^{11}	9.56×10^{-1}	4.61×10^{-2}	1.10×10^{-6}	1.34×10^{-11}

The SERs of the conventional DRAM-based main memory for personal computers ranges 25,000–75,000 failures in time (FIT) per Mbit, which corresponds to 0.86–2.58 soft errors per hour in a 4-GB DRAM [29]. Converting this value to the error rate of one line, it can be found that the LER should at most 10^{-11} to achieve DRAM-compatible reliability. As shown in Table 2, the 4LC PCM can meet this required reliability criteria with a BCH-16 scheme and scrubbing period of 2^8 seconds.

2.4. PCM-specific Data Encoding Methods

Several studies were conducted to improve the performance and reliability of MLC PCMs considering the drift speed that depends on the data patterns. In [11], the rotation and flip technique was introduced to reduce the number of the most error-vulnerable patterns. The authors adopted a method of appending 2-bit redundancies per 8-bit data but did not completely eliminate the most error-vulnerable Level-2 pattern. Kwon et al. proposed a heterogeneous SLC and 4LC PCM architecture to improve the reliability and lifetime, with a storage density of 133% [8]. In [9], the authors presented a method that reduces the number of iterative write-and-verify operations by removing up to 48 Level-2 patterns in 256 4LC PCM cells. In [15], the authors proposed a method of compressing the data size in half to reduce the number of ECC parity bits. Seong et al. proposed a 3LC approach that eliminated all of the most error-vulnerable data patterns and achieved 10^5 times lower SER than that of the baseline 4LC [7]. However, a binary-to-ternary conversion overhead was involved, and the storage density was limited to 133%.

To significantly increase the storage density without compromising the reliability benefits of the 3LC PCM over 4LC PCM, we introduce a pattern-aware two-way binary-ternary data mapping for 3LC PCM. The number of additional storage cells required for the binary-to-ternary mapping can be reduced by maximizing the capacity of the 3LC PCM cell through appropriate data classification. Additionally, the performance degradation can be minimized by deploying a parallel encoding/decoding logic structure.

3. Error-vulnerable Pattern-aware Binary-to-Ternary Data Mapping

3.1. LC PCM Characeteristics

The 3LC PCM discards the most drift-prone pattern (Level-2 pattern; 11) in the 4LC PCM, thus can achieve comparable SER of DRAM without ECC support sustaining the scrubbing period of up to 2^{24} s ($\approx$ 194 days) [7]. Therefore, the 3LC PCM has significantly reduced ECC overhead in area and latency compared to 4LC PCM. Furthermore, the 3LC PCM requires a much smaller number of write-and-verify operations than the 4LC, resulting in an improved device lifetime and energy efficiency. Hence, 3LC with enhanced information capacity can be regarded as a promising alternative in replacing DRAM-based main memory.

As the writing data format is binary, conversion to ternary digits is required before being stored in 3LC PCM cells. Table 3 shows a binary-to-ternary mapping introduced in the existing 3LC PCM techniques. The existing 3LC PCM techniques performed the conversion method in 3-bit units where two cells store three bits, considering that the implementation complexity is exponentially proportional to the unit of numeric conversion. Thus, the storage density cannot exceed 133%.

Table 3. Three-bit unit binary-to-ternary data mapping.

3-Digit Binary Values	2-Digit Ternary Values
000	00
001	01
010	12
011	02
100	10
101	20
110	22
111	21

The storage density of 3LC PCM depends on the number of data patterns to be stored in PCM cells. An N-bit data has 2^N possible patterns, and the 3LC PCM requires at least $ceil\left(N \log_3 2\right)$ cells to accommodate all patterns; therefore, the upper bound of the storage density of the 3LC PCM is expressed as

$$Storage\ density_{3LC}(\%) \leq \frac{N}{ceil(Nlog_3 2)} \times 100 \quad (3)$$

The maximum achievable storage density of 3LC PCM is approximately 160% according to (3). For example, 19-bit data can be stored in 12 3LC PCM cells, and the storage density is at most 158.3%. In other words, the 3LC PCM can have a higher storage density than the 4LC PCM with BCH-16 (152%), as mentioned in Section 1. With this motivation, we propose a two-way encoding method to improve the storage density of 3LC PCM. For an N-bit encoding unit, 3^M [$M = floor\left(N \log_3 2\right)$; one less than the number of required cells] data patterns are first encoded into ternary values to maximize the 3LC PCM capacity, and subsequently different encoding schemes that can indicate the position of the rest of the data patterns are applied.

3.2. Overall Two-way Encoding and Decoding Process

The overall two-way binary-to-ternary data mapping is shown in Figure 3. First, a 64-bit unit data block is divided into eight encoding blocks (8 bits each) for block-wise parallel processing. The number of 3LC PCM cells required to store an 8-bit encoding block is six, but to reduce this number, each encoding block is stored in five 3LC PCM cells. For this, *Case Decision* logic classifies the pattern of an encoding block into 243 (3^5) and the remaining 13 patterns. As shown in Table 4, the data classification is based on the number of the most error-vulnerable pattern (Level-2 pattern; 11). The Case-0 encoding block processes 243 data patterns in which the number of Level-2 patterns is less than three. The remaining 13 data patterns of Case-1 are to be handled by the Case-1 encoding block. In addition, the data classification prior to the encoding has an advantage of mitigating the implementation complexity by reducing the number of patterns that the encoder should accommodate.

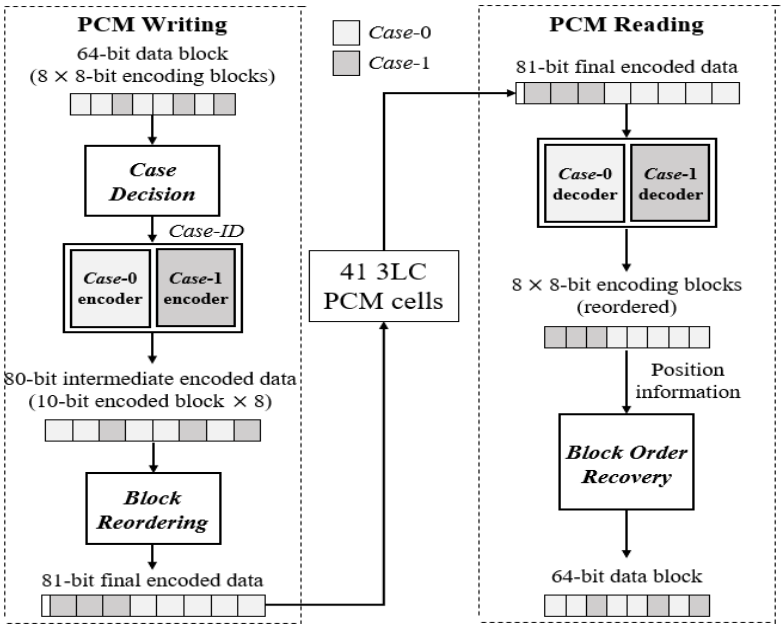

Figure 3. Overall encoding and decoding flow.

Depending on the classification by the *Case Decision* logic (Case-ID), each 8-bit encoding block is converted into Level-2 pattern-eliminated 10-bit encoded block by corresponding encoder to be stored in five 3LC PCM cells. Next, 10-bit encoded blocks are rearranged to make the best use of 3LC storage capacity. To facilitate an easy recovery of the reordered encoded blocks when data reading from the 3LC PCM, the original block-order information and an auxiliary bit for identifying the Case-ID of the encoded block is appended. After the data encoding, the 64-bit data block is stored in 41 3LC PCM cells, unlike the conventional 3LC PCM, which requires 48 cells.

Table 4. Pattern-aware case classification of 8-bit encoding block.

Case-ID	# of Level-2 Patterns	Example	# of Possible Patterns
Case-0	0	00 00 00 00	81
	1	11 00 01 10	108
	2	11 11 00 01	54
Case-1	3	11 11 11 00	12
	4	11 11 11 11	1

When reading data from the 3LC PCM, Case-0 and Case-1 decoders retrieve the original binary data. Next, the original sequence of the data blocks is restored by *Block Order Recovery* using the encoded block-order information stored in the Case-1 encoded blocks, and then the recovered 64-bit data block is transferred.

3.3. Two-way Data Encoding Method

Figure 4 shows the encoding process of the Case-0 encoding block. As presented in Table 4, 81 of 243 encoding block patterns that do not contain the Level-2 pattern are simply converted into 10-bit by appending the Level-0 pattern (00) as a prefix. The rest 162 encoding block patterns containing the Level-2 pattern are encoded via the Case-0 look-up table (LUT) to facilitate the storing of ternary-format into 3LC PCM cells.

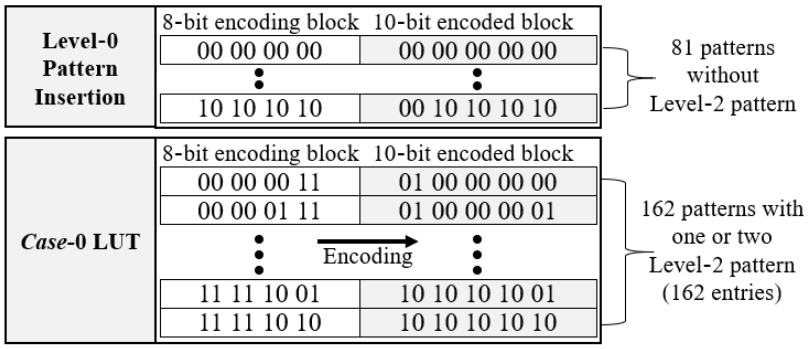

Figure 4. Case-0 encoder configuration.

The primary advantage of using LUTs instead of random logic is higher speed since the encoding rule is fixed. The number of multiplexers constituting the LUT increases exponentially in proportion to the input data size. Thus, we split the input data into 8-bit encoding blocks instead of using a single LUT. By performing parallel encoding through such small- sized LUTs, both throughput and area efficiency can be improved.

As each 10-bit encoded block to be stored into five 3LC cells for both Case-0 and Case-1 encoding block cannot hold all 8-bit possible patterns ($3^5 < 2^8$), the Case-ID must be included within the 80-bit intermediate encoded data for correct decoding. For this purpose, eight 10-bit encoded blocks are rearranged. When encoded blocks with the same Case-ID are clustered, only 1-bit information is added to indicate the Case-ID of encoded block cluster, thereby reducing the amount of extra information. As shown in Figure 5a, each 8-bit Case-1 encoding block is reconfigured to a new 12-bit block by appending 3-bit original 8-bit encoding block sequence ($1^{st} - 8^{th}$) and 1-bit Case-ID.

More specifically, the Case-1 encoding is performed by the Case-1 LUT to produce the 10-bit encoded block incorporating three pieces of following information: (1) Case-ID of next encoded block after clustering, (2) Original position of each Case-1 encoding block within the input 8 × 8-bit encoding blocks, and (3) Data pattern of Case-1 encoding block. Thus, Case-1 LUT consists of 208 (= 2 × 8 × 13) entries as shown in Figure 5b. Considering the performance degradation by the encoding latency, each encoding block is processed in parallel through the multiple *Case Decision* logics and encoders.

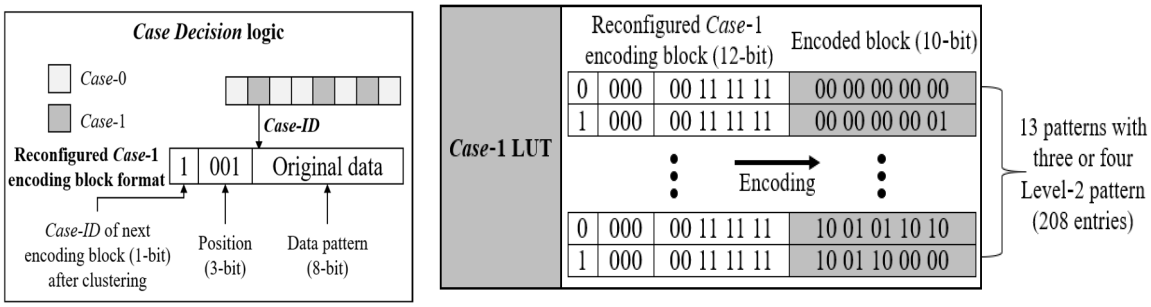

Figure 5. (**a**) Reconfigured Case-1 encoding block; (**b**) Case-1 encoder configuration.

Clustering the encoded blocks with the same Case-ID is performed *by Block Reordering* logic, as shown in Figure 6. If at least one Case-1 encoded block exists in the 80-bit intermediate encoded data, then the order of encoded blocks can be recovered with the 3-bit position information. Otherwise, i.e., when the intermediate encoded data consists of only Case-0 encoded blocks with no position information, one auxiliary bit is required for correct decoding. The role of the auxiliary bit is to indicate whether the Case-1 encoded block exists in the 81-bit final encoded data or not. If all encoded blocks have the same Case-ID, then reordering is not performed.

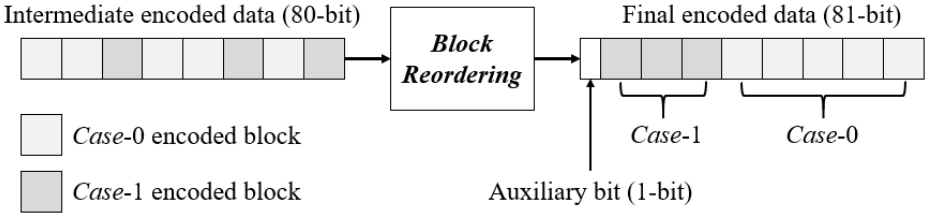

Figure 6. Function of *Block Reordering* logic.

3.4. Data Decoding Method

Algorithm 1 shows a decoding procedure that retrieves the 64-bit data block from the 81-bit final encoded data. If the auxiliary bit is 0 (Case-0), then all encoded blocks can be simply restored with the Case-0 decoder. Otherwise (i.e., Case-1), all encoded blocks are supplied to both Case-0 and Case-1 decoders in parallel. As described in Section 3.3, the Case-1 encoded block cluster is placed ahead of the Case-0 block cluster, and thus, the positions of Case-0 encoded blocks can be determined using the Case-ID of the next encoded block stored in the Case-1 encoded blocks. If the Case-ID stored in the last Case-1 encoded block is Case-0, then the rest of the encoded blocks are Case-0. After checking the Case-ID of every incoming encoded block, each decoder discards the data with mismatched Case-ID. Next, the *Block Order Recovery* logic recovers the block-order via the 3-bit position information stored in the Case-1 encoded blocks.

Algorithm 1 Decoding procedure.

```
Input: 81-bit final encoded data
Output: 64-bit data block

function Decoding
if auxiliary bit == Case-0 // all encoded blocks are Case-0
    perform Case-0 decoder (all encoded blocks)
else
    perform Case-0 decoder (all encoded blocks)
    perform Case-1 decoder (all encoded blocks)
    for i = 1 to 7 do
        if Case-ID of next encoded block == Case-0 then
            boundary = i + 1 // Set location of Case-0 encoded block cluster
            break
        end if
    end for

    for i = 1 to boundary do
        discard data_i decoded with Case-0 decoder
        place Case-1 encoding block using the 3-bit position information
    end for
    for i = boundary + 1 to 7 do // if boundary is eight, it means the auxiliary bit is Case-0
        discard data_i decoded with Case-1 decoder
        place Case-0 encoding block at the remained positions sequentially
    end for
end if
end function
```

With the pervasive 64-bit computing, the unit data transaction size from/to main memory is an integer multiple of 64-bit. Therefore, our method is designed to be easily extendable for multiple 64-bit data transactions. The logic area overhead increases only proportionally to the size of the data transaction, and the latency can be hidden by parallel processing. In addition, the storage density can

be enhanced by reducing the number of PCM cells storing auxiliary bits. For example, three final encoded data can share two PCM cells to store the auxiliary bit of each data block, as shown in Figure 7. Thus, processing multiple data blocks at the same time can further improve the storage density of the 3LC PCM, as shown in Figure 8.

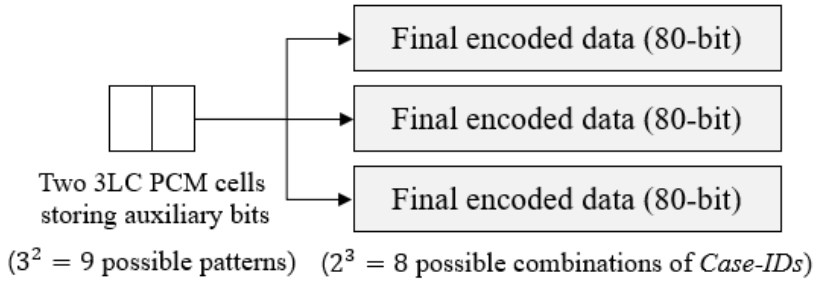

Figure 7. Example where three final encoded data share two auxiliary bits.

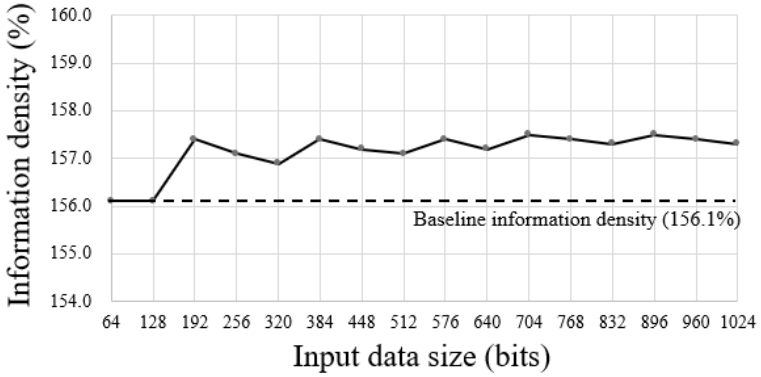

Figure 8. Storage density variation of 3LC PCM using proposed method for integer multiples of 64-bit data blocks.

4. Evaluation

4.1. Experimental Setup

4.1.1. Simulation Environment

We evaluated the storage density, lifetime, and performance improvements by the proposed method in comparison to the conventional 3LC PCM, which uses 3-bit unit data conversion, 4LC PCM with ECC, and three other related studies that reduced the number of the most error-vulnerable patterns of 4LC PCM. One is the relaxed write/read method proposed in [9] (referred to as "RXR"), one is the heterogeneous PCM architecture proposed in [8] (referred to as "HPCM"), and the other one is the cost-effective reliable MLC PCM architecture proposed in [15] (referred to as "CRPCM"). Both are focused on improving the reliability and lifetime of the 4LC PCM. We conducted a system-level simulation using Gem5 [30] and NVMain 2.0 [31] with 21 benchmarks from the SPEC CPU 2006 benchmark suite [32]. To eliminate the effect of cold cache misses, which impedes accurate performance evaluation, we first executed 100 million instructions for each benchmark. After that, actual performance evaluations were conducted based on the execution results of next 500 million instructions for each benchmark [33]. Table 5 lists the simulation parameters for the evaluation. As the number of iterative write-and-verify operations is different for each resistance level, as discussed in Section 2, the various parameters for the MLC writing operation are adopted from [34]. The size of the data transaction between the last-level cache and main memory is 512-bit, a typical cache line size.

Table 5. Simulation parameters.

Cores	4-Core, Alpha, 2 GHz
L1 I/D Cache	32 KB, 2-way, 2-cycle latency
L2 Cache	1 MB, 8-way, 20-cycle latency
L3 Cache	16 MB, 16-way, 50-cycle latency
Main Memory	MLC PCM, 16 GB, 400 MHz clock
Number of banks	8 (2 GB for each bank)
Memory Controller	FRFCFS
Memory write latency (pattern) [34]	150 ns (00), 200 ns (01), 210 ns (11), 60 ns (10)

4.1.2. ECC and Scrubbing Conditions

The storage density of 4LC PCM is affected by the number of parity bits of ECC schemes, while the 3LC PCM does not require any ECC. In addition, the lifetime and performance are deteriorated by frequent scrubbings. Thus, for a fair comparison, we selected the ECC and scrubbing period wherein each study can meet DRAM-compatible reliability. To evaluate the reliability for writing data using SPEC CPU 2006 benchmarks, we used Equation (2) to calculate the LER of a 512-bit cache line. To calculate SER_{cell}, the average SER of a single MLC PCM cell, we extracted frequencies of 2-bit patterns (00, 01, 10, and 11) in benchmark data considering the drift dependencies of each pattern. For each benchmark graph in Figure 9, the left bar indicates the original frequencies of 2-bit patterns and the right one denotes the changed frequencies after applying the proposed encoding technique. The values of SER_{cell} in the proposed method, conventional 3LC PCM [7], and "RXR" [9] are calculated considering the changed pattern frequencies caused by their different data mapping techniques. Meanwhile, 4LC PCM, "HPCM" [8], and "CRPCM" [15] do not require data conversion.

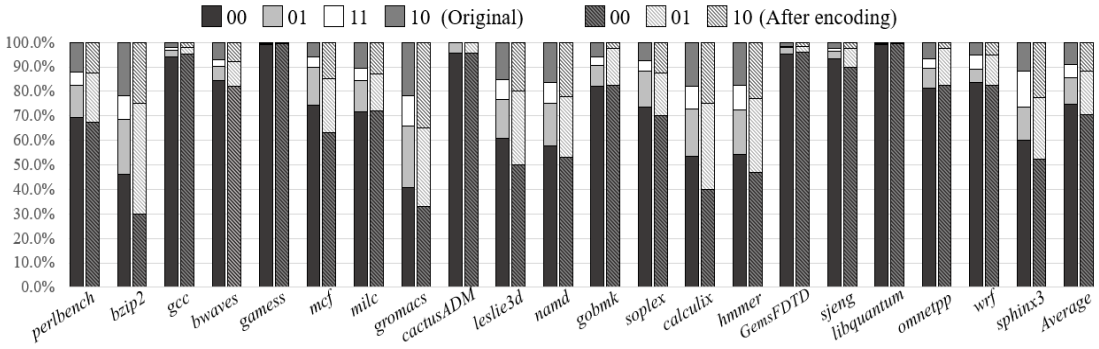

Figure 9. Frequencies of data patterns (00, 01, 11, 10) before/after encoding for the selected SPEC CPU 2006 benchmarks.

An SER comparison between the proposed method and related studies are presented in Table 6, where "Negligible" indicates the calculated SER value is lower than 10^{-13} (as mentioned in Section 2, SER criteria of DRAM-based main memory are on the order of 10^{-11}). The overhead of scrubbing is defined as the ratio of the total time used for scrubbing and the scrubbing period. In the case of 2GB memory bank, the total time used for scrubbing is 9.65 s [7]. Thus, we assumed a scrubbing period of 2^{10} seconds that has an ignorable effect on the lifetime and performance (<1.0%). In this assumption, 4LC PCM requires BCH-16, both "HPCM" and "RWR" require BCH-12, whereas, the proposed method and conventional 3LC PCM do not need any ECC. "CRPCM" uses the same ECC algorithm as 4LC PCM, the size of the encoding unit is reduced using the virtual data, and only the number of parity bits is reduced.

Table 6. Comparison of 512-bit cache line LERs between proposed method and previous studies.

Scrubbing Period (s)	4LC PCM BCH-16	HPCM [8] BCH-12	RWR [9] BCH-12	CRPCM [15] BCH-16	3LC PCM [7] No ECC	Proposed No ECC
2^6	Negligible	Negligible	Negligible	Negligible	Negligible	Negligible
2^7	Negligible	Negligible	Negligible	Negligible	Negligible	Negligible
2^8	Negligible	Negligible	Negligible	Negligible	Negligible	Negligible
2^9	4.8×10^{-13}	Negligible	Negligible	6.2×10^{-14}	Negligible	Negligible
2^{10}	5.6×10^{-11}	2.5×10^{-13}	6.9×10^{-13}	9.4×10^{-12}	Negligible	Negligible
2^{11}	3.9×10^{-9}	7.8×10^{-12}	4.6×10^{-11}	7.1×10^{-10}	Negligible	Negligible
2^{12}	1.6×10^{-7}	1.7×10^{-10}	1.1×10^{-9}	3.2×10^{-8}	Negligible	Negligible

4.2. Storage Density

Table 7 presents a comparison of the storage density considering the number of parity cells of ECC schemes and additional cells ("Add. cell" in Table 7) of each architecture when storing 512-bit data. The storage densities of "RWR (3LC+4LC)", "HPCM (SLC+4LC)", and "CRPCM (4LC)" are 151.5%, 119.9%, and 155.6%, respectively. The 3LC PCM using the proposed method has a 17.9% enhanced storage density compared to the conventional 3LC PCM, and even has a 3.1% higher storage density than the 4LC PCM that needs BCH-16.

Table 7. Comparison of storage density between proposed method and previous studies.

Related Studies	Number of PCM Cells (for 512-Bit Cache Line)				Storage Density
	Data Cells	Parity Cells	Add. Cells	Total Cells	
4LC PCM (BCH-16)	256	80	0	336	152.4%
RWR (BCH-12) [9]	256	60	32	338	151.5%
HPCM (BCH-12) [8]	384	43	0	427	119.9%
CRPCM (BCH-16) [15]	256	73	0	329	155.6%
3LC PCM [7]	384	0	0	384	133.3%
Proposed	320	0	4	324	157.1%

To calculate the area overhead of the proposed architecture, we estimated the area using Synopsys Design Compiler™ and SAED32, 32nm Synopsys Educational Design Kit. The area of the PCM cells and peripheral circuitry of each memory bank were extracted using NVSim [35] with a 32 nm process node. The estimated area overhead of the proposed encoders and decoders is approximately 0.52% of the area of a single PCM bank, which is negligible even when parallel processing is used, as presented in Table 8.

Table 8. Area overhead of proposed method.

Module	Area Per Bank (mm^2)
Memory cells	2.466
Peripheral circuits	0.326
Proposed encoder	0.007
Proposed decoder	0.008
Area overhead	0.52%

4.3. Lifetime

To evaluate the lifetime improvement by the proposed method, we estimated the normalized number of actual write counts per PCM cell considering the number of iterative write-and-verify operations for each 2-bit pattern. The estimated write counts for different benchmarks are shown in

Figure 10. "HPCM" and "RWR", which have a non-uniform lifetime distribution per each PCM cell because of using heterogeneous array architecture (3LC+4LC in "RWR" and SLC+4LC in "HPCM"), are excluded from the estimation. "CRPCM" has the same lifetime with the conventional 4LC PCM with BCH-16. The lifetime improved by 2.39 times in bzip2 which has the highest ratio of Level-2 pattern and improved by 1.04 times in libquantum which has the smallest number of level-2 pattern. On average, the lifetime of the 3LC PCM with the proposed method is enhanced by 36.1% when compared to that of the 4LC PCM.

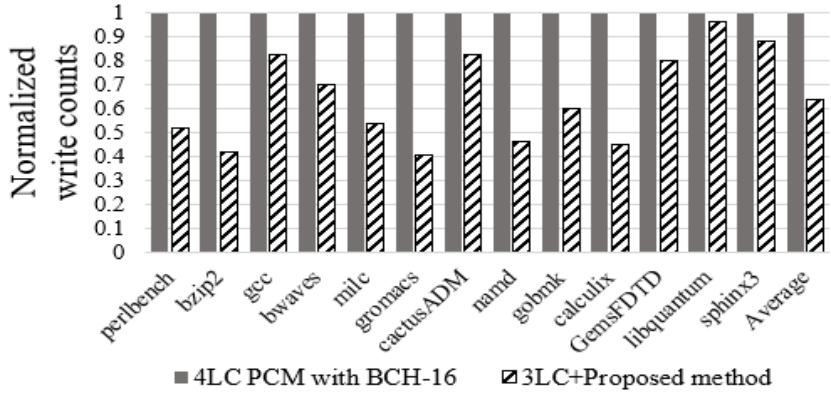

Figure 10. Comparison of lifetime between proposed method and 4LC PCM with an ECC.

4.4. Performance

In the comparative analysis of the performance, we considered both the number of iterative write operations and the encoding latency. Read latencies were not analyzed because 3LC and 4LC can be read as fast as SLC by using multiple sense amplifiers. Figure 11 presents the normalized write latency in terms of the relative instruction-per-cycle (IPC) for the benchmarks. The ECC encoding latencies for BCH-12 and BCH-16 were calculated based on [36]. "HPCM", "RWR", "CRPCM", and the conventional 3LC PCM show performance improvements of 17.8%, 29.8%, 14.5%, and 39.1% on average compared to 4LC PCM with BCH-16, respectively. The 3LC PCM with the proposed method improves its performance by 38.8% on average, and the maximum and minimum performance improvements are 69.8% and 17.6%, respectively. In particular, the proposed method improved the IPC by more than 50% in benchmarks, where the number of memory accesses are high such as GemsFDTD and cactusADM. The maximum delay of the encoding logic was less than 10ns as estimated using Synopsys PrimeTime®. The encoding logic is therefore designed as a four-stage pipeline that runs on the same 400MHz clock as the memory controller. Hence, our technique needs an additional one or two clock cycles for writing than that of the conventional 3LC PCM depending on whether the Case-1 encoding block is existed in the 64-bit data block. However, the encoding latency affects the overall IPC by only 0.7% considering the parallel processing and the inherent long write latency of the MLC PCM.

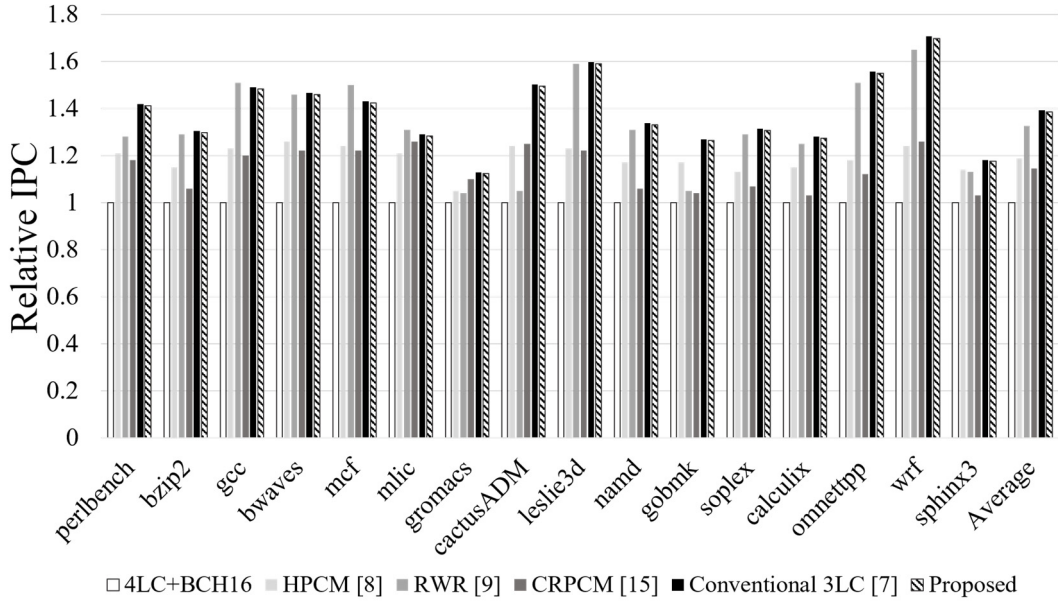

Figure 11. Comparison of performance with respect to relative IPC between proposed methods and previous studies.

5. Conclusions

MLC PCM is an attractive solution to replace DRAM-based main memory but has shortcomings with regard to reliability and performance. Several recent studies focused on reducing the number of most error-vulnerable patterns of 4LC PCM to improve the reliability. However, a complex ECC and frequent scrubbing, which cause performance and lifetime degradation, are required. To overcome these challenges, 3LC PCM-based architectures are introduced to improve the lifetime and performance without the need for an ECC. However, 3LC PCM requires binary-to-ternary data conversion, which reduces the storage density. Herein, we proposed error-vulnerable, pattern-aware two-way binary-to-ternary data mapping to improve the storage density 3LC PCM. Considering that the storage density depends on the number of data patterns, the proposed method maximizes the capacity of the 3LC PCM cell through data classification. Furthermore, the encoding latency is minimized through block-wise parallel processing. Experimental results indicated that our method achieved 17.9% higher storage density than the conventional 3LC PCM, with a negligible logic area overhead. The lifetime and performance were improved by 36.1% and 38.8%, respectively, compared with those of 4LC PCM with BCH-16.

Author Contributions: Conceptualization, J.B.H. and T.H.H.; methodology, J.B.H. and Y.S.L.; software, J.B.H. and Y.W.K.; validation, J.B.H., Y.S.L. and T.H.H.; formal analysis, T.H.H.; investigation, J.B.H.; resources, T.H.H.; data curation, J.B.H.; writing—original draft preparation, J.B.H.; writing—review and editing, Y.S.L. and T.H.H.; visualization, J.B.H. and Y.W.K.; supervision, T.H.H.; project administration, T.H.H.; funding acquisition, T.H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-00421, AI Graduate School Support Program (Sungkyunkwan University)) and in part by the Ministry of Trade, Industry and Energy (MOTIE) and Korea Semiconductor Research Consortium (KSRC) support program (10080594) for the development of the future semiconductor device.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, B.C.; Zhou, P.; Yang, J.; Zhang, Y.; Zhao, B.; Ipek, E.; Mutlu, O.; Burger, D. Phase-change technology and the future of main memory. *IEEE Micro* **2010**, *30*, 131–141. [CrossRef]

2. Lefurgy, C.; Rajamani, K.; Rawson, F.; Felter, W.; Kistler, M.; Keller, T.W. Energy management for commercial servers. *Computer* **2003**, *36*, 39–48. [CrossRef]
3. Burr, G.; Narayanan, P.; Shelby, R.; Sidler, S.; Boybat, I.; di Nolfo, C.; Leblebici, Y. Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power). In Proceedings of the 2015 IEEE International Electron Devices Meeting (IEDM), Washington, DC, USA, 7–9 December 2015. [CrossRef]
4. Burr, G.W.; Shelby, R.M.; Sidler, S.; Di Nolfo, C.; Jang, J.; Boybat, I.; Shenoy, R.S.; Narayanan, P.; Virwani, K.; Giacometti, E.U. Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses) using phase-change memory as the synaptic weight element. *IEEE Trans. Electron Devices* **2015**, *62*, 3498–3507. [CrossRef]
5. Jia, G.; Han, G.; Jiang, J.; Liu, L. Dynamic adaptive replacement policy in shared last-level cache of DRAM/PCM hybrid memory for big data storage. *IEEE Trans. Ind. Inform.* **2016**, *13*, 1951–1960. [CrossRef]
6. Tavana, M.K.; Ziabari, A.K.; Kaeli, D. Live together or die alone: Block cooperation to extend lifetime of resistive memories. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 1098–1103.
7. Seong, N.H.; Yeo, S.; Lee, H.-H.S. Tri-level-cell phase change memory: Toward an efficient and reliable memory system. In Proceedings of the 40th Annual International Symposium on Computer Architecture, Tel-Aviv, Israel, 23–27 June 2013; pp. 440–451.
8. Kwon, T.; Imran, M.; You, J.M.; Yang, J.-S. Heterogeneous PCM array architecture for reliability, performance and lifetime enhancement. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 1610–1615.
9. Rashidi, S.; Jalili, M.; Sarbazi-Azad, H. Improving MLC PCM performance through relaxed write and read for intermediate resistance levels. *ACM Trans. Archit. Code Optim.* **2018**, *15*, 1–31. [CrossRef]
10. Awasthi, M.; Shevgoor, M.; Sudan, K.; Rajendran, B.; Balasubramonian, R.; Srinivasan, V. Efficient scrub mechanisms for error-prone emerging memories. In Proceedings of the IEEE International Symposium on High-Performance Comp Architecture, New Orleans, LA, USA, 25–29 February 2012; pp. 1–12.
11. Zhang, W.; Li, T. Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system. In Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), Hong Kong, China, 27–30 June 2011; pp. 197–208.
12. Khouzani, H.A.; Hosseini, F.S.; Yang, C. Segment and conflict aware page allocation and migration in dram-pcm hybrid main memory. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2016**, *36*, 1458–1470. [CrossRef]
13. Wang, R.; Zhang, Y.; Yang, J. ReadDuo: Constructing reliable MLC phase change memory through fast and robust readout. In Proceedings of the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, 28 June–1 July 2016; pp. 203–214.
14. Imran, M.; Kwon, T.; Yang, J.-S. Enrely: A reliable MLC PCM architecture based on data encoding. In Proceedings of the 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), JeJu, Korea, 23–26 June 2019; pp. 1–4.
15. Kwon, T.; Imran, M.; Yang, J.-S. Cost-effective reliable MLC PCM architecture using virtual data based error correction. *IEEE Access* **2020**, *8*, 44006–44018. [CrossRef]
16. Thakkar, I.G.; Pasricha, S. DyPhase: A dynamic phase change memory architecture with symmetric write latency and restorable endurance. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2017**, *37*, 1760–1773. [CrossRef]
17. Das, A.; Sánchez-Macián, A.; García-Herrero, F.; Touba, N.A.; Maestro, J.A. Enhanced limited magnitude error correcting codes for multilevel cell main memories. *IEEE Trans. Nanotechnol.* **2019**, *18*, 1023–1026. [CrossRef]
18. Gang, W. Threat models and security of phase-change memory. In Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 11–13 January 2019; pp. 1–6.
19. Mittal, S. A survey of soft-error mitigation techniques for non-volatile memories. *Computers* **2017**, *6*, 8. [CrossRef]
20. Zhao, M.; Xue, Y.; Hu, J.; Yang, C.; Liu, T.; Jia, Z.; Xue, C.J. State asymmetry driven state remapping in phase change memory. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2016**, *36*, 27–40. [CrossRef]

21. Yoon, D.H.; Chang, J.; Schreiber, R.S.; Jouppi, N.P. Practical nonvolatile multilevel-cell phase change memory. In Proceedings of the SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 17–22 November 2013; pp. 1–12.
22. El-Hassan, N.H.; Kumar, T.N.; Almurib, H.A.F. Implementation of time-aware sensing technique for multilevel phase change memory cell. *Microelectron. J.* **2016**, *56*, 74–80. [CrossRef]
23. Kim, N.S.; Song, C.; Cho, W.Y.; Huang, J.; Jung, M. LL-PCM: Low-latency phase change memory architecture. In Proceedings of the 56th Annual Design Automation Conference 2019, Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6.
24. Lue, H.-T.; Hsu, T.-H.; Wang, S.-Y.; Lai, E.-K.; Hsieh, K.-Y.; Liu, R.; Lu, C.-Y. Study of incremental step pulse programming (ISPP) and STI edge effect of BE-SONOS NAND flash. In Proceedings of the 2008 IEEE International Reliability Physics Symposium, Phoenix, AZ, USA, 27 April–1 May 2008; pp. 693–694.
25. Ielmini, D.; Boniardi, M.; Lacaita, A.L.; Redaelli, A.; Pirovano, A. Unified mechanisms for structural relaxation and crystallization in phase-change memory devices. *Microelectron. Eng.* **2009**, *86*, 1942–1945. [CrossRef]
26. Kim, S.; Lee, B.; Asheghi, M.; Hurkx, F.; Reifenberg, J.P.; Goodson, K.E.; Wong, H.-S.P. 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 Devices* **2011**, *58*, 584–592. [CrossRef]
27. Sebastian, A.; Papandreou, N.; Pantazi, A.; Pozidis, H.; Eleftheriou, E. Non-resistance-based cell-state metric for phase-change memory. *J. Appl. Phys.* **2011**, *110*, 084505. [CrossRef]
28. Yu, S.; Chen, P.-Y. Emerging memory technologies: Recent trends and prospects. *IEEE Solid-State Circuits Mag.* **2016**, *8*, 43–56. [CrossRef]
29. Schroeder, B.; Pinheiro, E.; Weber, W.-D. DRAM errors in the wild: A large-scale field study. *ACM SIGMETRICS Perform. Eval. Rev.* **2009**, *37*, 193–204. [CrossRef]
30. Binkert, N.; Beckmann, B.; Black, G.; Reinhardt, S.K.; Saidi, A.; Basu, A.; Hestness, J.; Hower, D.R.; Krishna, T.; Sardashti, S. The gem5 simulator. *ACM SIGARCH Comput. Archit. News* **2011**, *39*, 1–7. [CrossRef]
31. Poremba, M.; Zhang, T.; Xie, Y. Nvmain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems. *IEEE Comput. Archit. Lett.* **2015**, *14*, 140–143. [CrossRef]
32. Henning, J.L. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Comput. Archit. News* **2006**, *34*, 1–17. [CrossRef]
33. Nair, A.A.; John, L.K. Simulation points for SPEC CPU 2006. In Proceedings of the 2008 IEEE International Conference on Computer Design, Lake Tahoe, CA, USA, 12–15 October 2008; pp. 397–403.
34. Joshi, M.; Zhang, W.; Li, T. Mercury: A fast and energy-efficient multi-level cell based phase change memory system. In Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture, San Antonio, TX, USA, 12–16 February 2011; pp. 345–356.
35. Dong, X.; Xu, C.; Xie, Y.; Jouppi, N.P. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2012**, *31*, 994–1007. [CrossRef]
36. Strukov, D. The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories. In Proceedings of the 2006 Fortieth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 29 October–1 November 2006; pp. 1183–1187.