Improving MLC PCM Performance through Relaxed Write and Read for Intermediate Resistance Levels

SAEED RASHIDI and MAJID JALILI, Sharif University of Technology
HAMID SARBAZI-AZAD, Sharif University of Technology & Institute for Research in Fundamental Sciences (IPM)

Phase Change Memory (PCM) is one of the most promising candidates to be used at the main memory level of the memory hierarchy due to poor scalability, considerable leakage power, and high cost/bit of DRAM. PCM is a new resistive memory that is capable of storing data based on resistance values. The wide resistance range of PCM allows for storing multiple bits per cell (MLC) rather than a single bit per cell (SLC). Unfortunately, higher density of MLC PCM comes at the expense of longer read/write latency, higher soft error rate, higher energy consumption, and earlier wearout compared to the SLC PCM. Some studies suggest removing the most error-prone level to mitigate soft error and write latency of MLC PCM, hence introducing a less dense memory called Tri-Level memory. Another scheme, called M-Metric, proposes a new read metric to address the soft error problem in MLC PCM.

In order to deal with the limited lifetime of PCM, some extra storage per memory line is required to correct permanent hard errors (stuck-at faults). Since the extra storage is used only when permanent faults occur, it has a low utilization for a long time before hard errors start to occur. In this article, we utilize the extra storage to improve the read/write latency in a 2-bit MLC PCM using a relaxation scheme for reading and writing the cells for intermediate resistance levels. More specifically, we combine the most time-consuming levels (intermediate resistance levels) to reduce the number of resistance levels (making a Tri-Level PCM) and therefore improve write latency. We then store some error correction metadata in the extra storage section to successfully retrieve the exact data values in the read operation. We also modify the Tri-Level PCM cell to increase its read latency when the M-Metric scheme is used. Evaluation results show that the proposed scheme improves read latency by 57.2%, write latency by 56.1%, and overall system performance (IPC) by 26.9% over the baseline. It is noteworthy that combining the proposed scheme and FPC compression method improves read latency by 75.2%, write latency by 67%, and overall system performance (IPC) by 37.4%.

CCS Concepts: • **Information systems** → **Phase change memory**; • **Hardware** → **Memory and dense storage**; • **Computer systems organization** → Multicore architectures;

Additional Key Words and Phrases: Tri-Level PCM, M-metric, write speed, read latency, energy consumption

ACM Reference format:
Saeed Rashidi, Majid Jalili, and Hamid Sarbazi-Azad. 2018. Improving MLC PCM Performance through Relaxed Write and Read for Intermediate Resistance Levels. *ACM Trans. Archit. Code Optim.* 15, 1, Article 12 (March 2018), 31 pages.
https://doi.org/10.1145/3177965

Authors' addresses: S. Rashidi and H. Sarbazi-Azad, HPCAN Lab, Computer Engineering Department, Sharif University of Technology, Tehran, Iran; emails: rashidice@ce.sharif.edu, azad@sharif.edu; H. Sarbazi-Azad, School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran; email: azad@ipm.ir; M. Jalili is now at the Department of Electrical and Computer Engineering, University of Texas at Austin Texas, USA; email: majid@utexas.edu.

https://doi.org/10.1145/3177965

1 INTRODUCTION

With the increasing number of cores and developing sophisticated applications in today's computer systems, larger main memory capacity is increasingly demanded. The large capacity of main memory results in fewer page faults and more application parallelism. Unfortunately, DRAM cannot satisfy the increasing demand for larger main memory capacity due to its power and scalability limits that make further scaling of DRAM infeasible [31]. Therefore, emerging memory technologies have been proposed to be used in the main memory level of memory hierarchy.

Phase Change Memory (PCM) is an emerging memory that is a candidate for replacing DRAM technology. A PCM device consists of Chalcogenide material (GST), capable of changing its resistance. Therefore, PCM stores data based on its GST resistance level. Compared to DRAM, PCM is more scalable [44] and denser, and consumes less standby power.

The large resistance range of GST material allows for considering more than two resistance levels that lead to storing multiple bits per cell (known as MLC) and creating even denser PCM memories. However, utilizing MLC PCM has several challenges. In the presence of multiple resistance levels, the write operation should be performed in a more exact way. Therefore, an MLC write operation is performed using multiple Program and Verify (P&V) iterations that result in longer write latency and higher energy consumption. In addition, writing to the intermediate resistance levels in MLC PCM requires a more nondeterministic number of iterations [26, 36].

PCM cells also suffer from a limited lifetime, and extra storage per line can greatly help to tolerate cells' failure. However, this extra storage is not utilized for a long time before the line faces first permanent faults [17, 39]. Another drawback of MLC PCM is its vulnerability to resistance drift, which is changing cell value based on increasing cell resistance over time. A recent study [51] proposed Tri-Level PCM to use three resistance levels (instead of four levels in a 2-bit MLC PCM) to increase the reliability and write speed. In Tri-Level PCM, the most vulnerable resistance level of MLC PCM to drift (i.e., the third level) is removed, leaving a wide resistance region between the second and fourth resistance levels. This allows the three remaining levels to be read much more reliably; it also reduces the write latency in Tri-Level PCM.

The authors in [48] proposed to use a nonresistance metric (M-Metric) for determining cell value. In M-Metric, a linear voltage ramp is applied to determine the time it takes the cell to see a predefined reference current I_R level. Cell values are differentiated by the time they require to converge to I_R. In this technique, read latency is determined by the highest reference resistance (the resistance that differentiates the two highest resistance levels). M-Metric shows significant tolerance to resistance drift. Therefore, by using M-Metric, the soft error rate due to drift is significantly decreased, making M-Metric a suitable approach as a read mechanism, although it increases the average access latency.

To address the challenges of MLC PCM while exploiting its density, in this article, we propose a morphable MLC/Tri-Level PCM architecture that enjoys the speed and reliability of Tri-Level PCM and the capacity of MLC PCM. More precisely, we make the following contributions:

- We propose a slight modification on the read mechanism of the previously proposed Tri-Level PCM to boost its read speed compared to MLC PCM (Section 2.2.1). This is done by leveraging the vast region between the two highest resistance levels in Tri-Level PCM to reduce the highest reference resistance while reliability requirements are met. Using this new design, in the proposed morphable memory architecture using the M-Metric read mechanism, the memory lines stored in Tri-Level mode can be read much faster than those kept in MLC mode.
- We propose Relaxed Write/Read (RWR) to improve the write latency of MLC PCM by merging the intermediate resistance levels ("01" and "10") to one level (i.e., making it Tri-Level),

and use the unutilized hard error recovery storage to determine the exact value of the cells with intermediate resistance levels. To be more specific, in this method, instead of writing the exact values of intermediate state levels of MLC cells, they are written to the intermediate state level of Tri-Level cells, making them approximate. A simple mapping scheme is used to map each approximate cell to a single correction bit in an unutilized part of the extra storage section. The mapped bits in the unutilized part then determine the exact values of the approximate cells in the read operation. This way, the write latency of MLC PCM is significantly decreased while the capacity of MLC PCM is maintained. We also show that the RWR method has a positive impact on the energy consumption and lifetime of MLC PCM.

The objective of this article is to provide a PCM main memory that has the capacity density of MLC PCM with much better access latency and reliability than the conventional MLC PCM.

2 PRELIMINARIES

There are several challenges in exploiting MLC PCM for mass production. In comparison with SLC PCM, the MLC write operation must spend more cycles to finish since there are a couple of intermediate states to be programmed. Basically, there are two types of write pulses in MLC PCM; SET and RESET. A RESET pulse is a short and high-power pulse that drives the cell's temperature above its melting point and then quenches its temperature quickly. This results in a larger amorphous region, thus increasing the resistance of the cell. The SET pulse is a long and moderate-power pulse that gradually shrinks the amorphous region and hence decreases its resistance. Figure 1 shows typical SET and RESET write pulses. To cope with the difficulty of the write operation in MLC PCM, the (P&V) method is proposed in [36]. The write operation is divided into multiple P&V iterations. After each iteration, the resistance of the cell is examined: if it is in the desired resistance range, the write operation is finished; otherwise, more P&V iterations are applied.

Another issue is the limited write endurance of PCM cells; that is, after a certain number of writes, the cell will be permanently stuck at a constant value, no matter which write the pulse is applied to. After each write, a comparison read is required to ensure that the data were correctly written [17, 47]. Faulty cells are detected if the checking read returns unequal data with respect to the written data. To deal with this limitation, typically an extra storage per line is required to cover such permanent cell failures. Typically, the acceptable amount of storage overhead is 12.5%. For example, for 512-bit lines (a typical line size in PCM), 64-bit storage overhead is reserved to deal with such errors. Different methods like [17, 47, 50] try to use this storage in different ways to cover more stuck-at faults.

One of the major concerns related to PCM is soft error caused by resistance drift, that is, increasing the resistance value over time. According to the study in [22], the drift could be modeled as

$$R_{drift}(t) = R \times \left(\frac{t}{t_0}\right)^{\alpha}. \tag{1}$$

In Equation (1), R is the initial resistance of the cell after programming, $R_{drift}(t)$ is the resistance of the cell after time t, t_0 is the normalization constant that is usually equal to 1, and α is the drift exponent. The drift exponent has a normal distribution of $N(\mu_\alpha, \sigma_\alpha^2)$ for each state. It is observed that the drift exponent is proportional to initial resistance (i.e., R_0), which means higher R_0 leads to higher α. Thus, special care must be taken to deal with the resistance drift problem. One common way is to use the scrubbing mechanism, that is, periodically rewriting the lines to prevent a further increase of cells' resistance. Scrubbing comes at the cost of more traffic, more energy

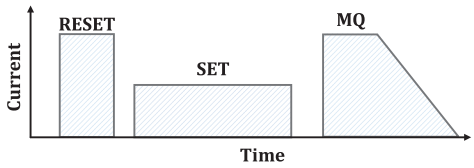


Fig. 1. SET, RESET, and moderate-quenched (MQ) pulses for programming intermediate states in MLC and Tri-Level PCM.

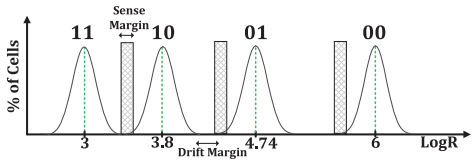


Fig. 2. MLC resistance distribution.

Table 1. MLC Resistance Distribution Parameters

		$LogR$					α	
State Level	Data	Target(μ_R)	Deviation(σ_R)	Range ($3\sigma_R$)	Sensing Margin	R Drift Margin	Mean (μ_α)	Deviation (σ_α)
0 (Reset)	00	6.00	0.098	0.295	0.2	-	0.1	$0.4\times\mu_\alpha$
1	01	4.74				0.47	0.06	
2	10	3.8				0.16	0.02	
3 (Set)	11	3				0.01	0.001	

consumption, and earlier wearout. Therefore, the memory system must be designed in a way that is reliable enough to let the scrubbing period be postponed in order to mitigate its overhead.

Resistance distribution of different states follows normal distribution $N(\mu_R, \sigma_R^2)$. In this article, we use the resistance distribution adopted from previous studies [20, 28]. Figure 2 shows the resistance bands of different states. Typically, the resistance domain is divided into three regions: target resistance band, drift margin region, and sensing margin.

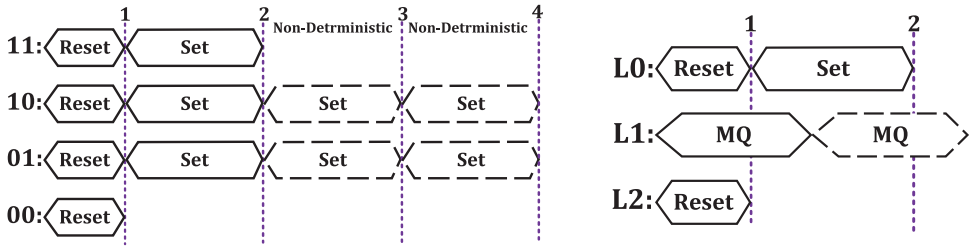
The target resistance band is the desired resistance range that needs to be obtained after cell programming. Here, the target resistance band is set to be $\mu_R \pm 3\sigma_R$. The drift margin accounts for dealing with the resistance drift problem. As stated before, the resistance of the cell is increased over time; thus, for each state, a safety margin (drift margin) is needed to let the resistance be safely increased while read is still realized correctly (as long as resistance does not pass the drift margin).

In order to address the noise in the read circuit, the sense margin is required [20]. In the read operation, the resistance of the cell is compared to the reference resistances (i.e., usually the middle of sense margins) to determine which state the cell belongs to. Table 1 presents the parameters associated with the distribution of Figure 2. Note that the parameters in Table 1 are based on [28].

2.1 Write Model

2.1.1 MLC Write Model. In MLC PCM, programming a cell to "00" and "11" states (or SET or RESET states) is done using a fixed number of iterations, but programming to intermediate states ("01" and "10" states) requires a number of nondeterministic P&V iterations. Because of the process variation and nondeterministic characteristics of PCM devices, the number of P&V iterations is cell dependent. Even for the same cell the number of required iterations varies [26]. Thus, different approaches have been proposed to model P&V-based write operations [26, 41, 46]. In this article, we adopt the write model presented in [46] and assume that intermediate states require 4.05 P&V iterations, on average, to be programmed, as reported in [36]. The detailed description of the model is presented in Appendix A.

To conclude the write scenario in MLC, Figure 3(a) shows the write operation for different states. As can be seen, the programming operation begins with a RESET pulse. A RESET pulse is enough



(a) MLC PCM write scenario

(b) Tri-Level PCM write scenario

Fig. 3. Required write pulses for writing different values in MLC and Tri-Level PCM.

Table 2. Tri-Level Resistance Distribution Parameters

State Level	LogR					α	
	Target(μ_R)	Deviation (σ_R)	Range (σ_R)	Sensing Margin	R Drift Margin	Mean (μ_α)	Deviation (σ_α)
L2	6	0.098	0.295($3\sigma_R$)	0.2	-	0.1	$0.4\mu_\alpha$
L1	4.042	0.110	0.528 ($4.8\sigma_R$)		0.13	0.0269	
L0	3	0.098	0.295($3\sigma_R$)		0.01	0.001	

for writing "00" data. By applying one SET pulse, the "11" data is written. But for intermediate states (i.e., "01" and "10"), a variable number of SET pulses should be applied for write operations.

2.1.2 Tri-Level Write Model. As stated before, MLC PCM suffers from several drawbacks. Recent studies in [51] showed that the drift problem might be so severe as to make MLC PCM infeasible. Thus, they proposed to remove the most vulnerable resistance level to drift, that is, the third level. By removing the third level, the reliability of PCM is improved. It is clear that removing a level will decrease the storage density. By using the coding technique discussed in [51], two Tri-Level cells could store 3 bits, and consequently the density of Tri-Level becomes 1.5 bits per cell. Moreover, since the third level is removed, the write operation to the intermediate state can be done using a single pulse [51]. By applying the moderate-quenched (MQ) programming pulse (proposed in [27]), the write operation to the intermediate state is done with one iteration in most cases. Figure 1 shows the MQ pulse. It is a RESET pulse with a controlled falling slope.

According to the study in [51], the latency of the MQ pulse is less than the SET pulse. This makes the latency of Tri-Level PCM close to SLC write latency. One drawback of MQ programming is its lesser precision compared to the P&V method [27, 51]. To analyze the distribution change due to MQ programming, we used a similar method as that used in [51]. Table 2 presents the associated parameters of Tri-Level PCM. As shown in the table, σ_R of the intermediate state is worsened from 0.098 to 0.110 and μ_R is increased from 3.8 to 4.042. The mean drift exponent also is increased from 0.02 to 0.0269.

Here, we redefine the resistance regions in a Tri-Level cell to improve its read/write latency. Since μ_R is increased, we increase the target resistance band of the intermediate state from $3\sigma_R$ to $4.8\sigma_R$. This results in a 100% - 1.5867E-04% success rate for write operations to the intermediate state to be finished in only one iteration. For the remaining 1.5867E-04% of writes, another MQ iteration is required, which is negligible. Thus, a verification step is required to see if the first MQ iteration is successful. Note that this verification step is not an overhead of MQ programming, since after each iteration the cell value must be checked for detecting stuck-at faults. We also set the drift margin of the intermediate state to 0.13. Later, we will show that this amount of drift

margin is sufficient to meet reliability requirements (discussed in Section 2.3). Unlike the original Tri-Level PCM design in [51], we also decrease the reference resistance between the second and third levels from 5.605 to 4.80. Decreasing the last reference resistance leads to improving the read latency as discussed in Section 2.2. As Tables 1 and 2 show, the first level of MLC PCM and first level of Tri-Level PCM, and the fourth level of MLC PCM and third level of Tri-Level PCM are equal in terms of resistance distributions.

Figure 3(b) shows the write scenario of Tri-Level PCM. Writing to the first and third levels is the same as MLC PCM, but for writing to the intermediate state, the MQ iteration is applied.

2.2 Read Model

The conventional read mechanism (called R-Metric) has severe vulnerability to resistance drift (R-Metric is described in Appendix B). As a result, a recent study [48] proposed to use a less sensitive read metric to drift, called M-Metric. In M-Metric, the cell is biased to a constant reference current (I_R) and the voltage across the cell is measured. In order to bias the cell to I_R, authors in [37, 48] proposed to apply a linear voltage ramp to the cell. Consequently, the time that it takes the cell to converge to I_R determines the cell state, because the higher resistance leads to a longer time to converge to I_R. Figure 4 explains the read process. Figure 4(a) shows a linear voltage ramp that is applied to the cell. Figure 4(b) shows the MLC read process. The three curves in Figure 4(b) are reference resistance curves of MLC PCM. According to Figure 4(b), if the cell gets to I_R between 0 and t_1, its content is interpreted as "11." If this time is between t_1 and t_2, it shows "10," and if it is between t_2 and t_3, its value represents "01." If the cell does not get to I_R by t_3, then its value is assumed to be "00."

Therefore, in this method, the read latency is determined by the highest reference resistance value. Since in Tri-Level PCM the highest reference resistance value is decreased from $402K\Omega$ to $63K\Omega$, the read latency is dramatically decreased.

2.2.1 Proposed M-Metric Read Mechanism. In the original Tri-Level PCM design [51], authors assumed parallel reads using the R-Metric mechanism; therefore, they assumed that the read latency of Tri-Level PCM is the same as SLC PCM. However, in addition to vulnerability of R-Metric to drift, parallel reading is a costly technique in terms of hardware overhead and sensing current [19]. By using M-Metric, both MLC and Tri-Level PCMs could be reliably used. Moreover, increased reliability of Tri-Level PCM compared to MLC PCM makes it possible to decrease the last reference resistance value in M-Metric, and hence the read speed in Tri-Level PCM is improved.

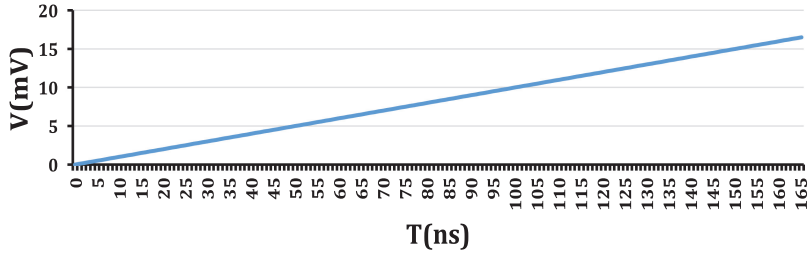
Figure 4(c) shows the Tri-Level M-Metric reading process, with the same mechanism used in MLC PCM. Note that in Tri-Level PCM, there are two resistance references.

The reference current is set to be $I_R = 1\mu A$ [37, 48]. According to [48], by applying a voltage ramp with a slope of K_{slope}, the time that takes the cell to get to I_R is given by

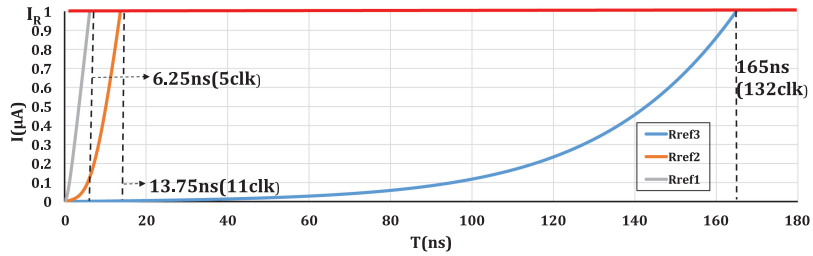
$$M(I_R) = \frac{2KTu_{A_{eff}}}{q\Delta Z K_{slope}} arcsinh\left[\frac{I_R \tau_0 e^{\frac{E_c - E_f}{kT}}}{2q\pi r_{E_{eff}}^2 NT\Delta Z}\right], \tag{2}$$

where q is the elementary charge, τ_0 is the characteristic attempt-to-escape time for a trapped electron, ΔZ is the mean intertrap distance, N_T is the trap density, K is the Boltzmann constant, T is the temperature, and $E_a = E_c - E_f$ is the activation energy. $r_{E_{eff}}$ is the effective bottom electrode radius and $u_{A_{eff}}$ is the effective amorphous thickness.

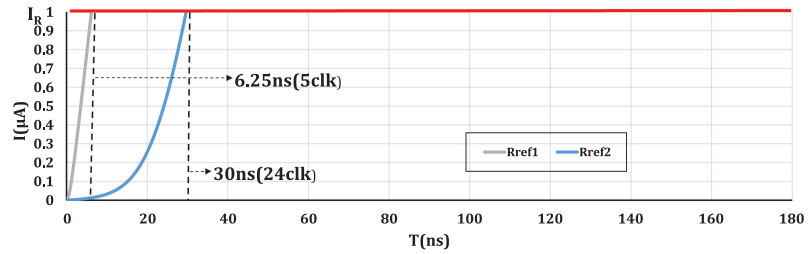
Both $r_{E_{eff}}$ and $u_{A_{eff}}$ are functions of PCM amorphous thickness, u_A. Figure 5(a) shows a PCM device with indicated amorphous thickness (u_A). Figure 5(b) shows the relationship between u_A, $r_{E_{eff}}$ and $u_{A_{eff}}$.



(a) Linear voltage ramp with $K_{slope} = 10^5$

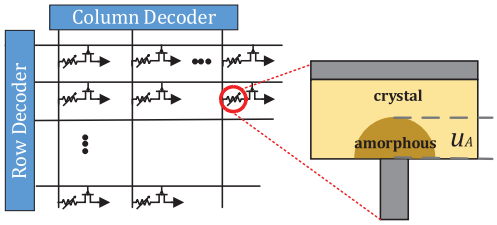


(b) MLC PCM read scheme.

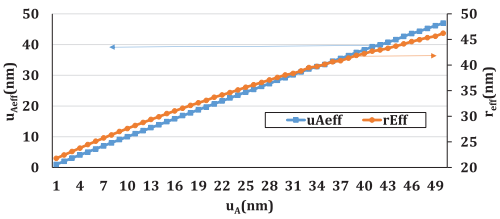


(c) Modified Tri-Level PCM read scheme.

Fig. 4. Timing details of M-Metric read in MLC PCM and proposed M-Metric read for Tri-Level PCM.



(a) Amorphous thickness, u_A, of a PCM cell.



(b) $r_{E_{eff}}$ and $u_{A_{eff}}$ curves as a function of u_A.

Fig. 5. Amorphous (mushroom) thickness parameters of PCM.

We consider $K_{slope} = 10^5$. Therefore, by using Equation (2), the sense latency of MLC PCM and Tri-Level PCM is 165ns and 30ns, respectively.[1]

[1]Note that according to the study in [48], in Equation (2), the resistance of the crystal part is assumed to be negligible compared to the amorphous part, and hence it is ignored. But at low resistances, the crystal resistance matters. Therefore, we have taken into account the crystal resistance by assuming $R_{crystal} = 500\Omega$, since the lowest resistance in our distribution is around 500Ω.

Table 3. M-Metric Distribution

State level	Data	$LogM$					α_M	
		Target(μ_M)	Deviation(σ_M)	Range ($3\sigma_M$)	Sensing Margin	M drift Margin	Mean (μ_{α_M})	Deviation (σ_{α_M})
0(Reset)	00	2.62	0.101	0.303	-	-	0.014	$0.4\mu_{\alpha_M}$
1	01	1.436	0.079	0.237	0.197	0.443	0.01	
2	10	0.893	0.030	0.09	0.124	0.087	0.003	
3(Set)	11	0.722	0.0074	0.022	0.052	0.005	0.0001	

(a) MLC PCM M distribution

State level	$LogM$					α_M	
	Target(μ_M)	Deviation (σ_M)	Range	Sensing Margin	M drift Margin	Mean (μ_{α_M})	Deviation (σ_{α_M})
L2	2.62	0.101	$0.303(3\sigma_M)$	-	-	0.014	$0.4\mu_{\alpha_M}$
L1	1.036	0.052	0.251 $(4.8\sigma_M)$	0.167	0.1	0.0038	
L0	0.722	0.0074	$0.022(3\sigma_M)$	0.052	0.005	0.0001	

(b) Tri-Level PCM M distribution

The predominant impact of temporal drift is growth in activation energy (E_a). In Equation (2), M is weakly affected by activation energy (E_a). According to the study in [48], the drift exponent of M-Metric is an order of magnitude less than R-Metric.

Applying a linear voltage ramp is a convenient and low-cost method that simplifies the read circuit logic. In the read operation, the current through the cell needs to be compared with I_R. At certain times, the read circuit checks if the cell has reached I_R and therefore determines the cell state.

However, one drawback of the linear voltage ramp is that its latency is determined by the reference resistances, meaning if the resistance range increases, the latency is increased as well. In this case, increasing the ramp slope or using a nonlinear voltage ramp may reduce the latency [37, 48]. However, in the range of interest ($1K\Omega - 1M\Omega$), the linear voltage ramp gives reasonable read latency for MLC PCM, and hence we use this method for our read operation.

2.3 Reliability Model

The probability of soft error rate (SER) should be calculated in order to determine the scrubbing period. SER is the probability for a cell to change its state due to resistance drift. Since the resistance of the cell is permanently increased due to drift, the scrubbing operation is needed eventually. The scrubbing period should be determined in a way that makes its overhead low while keeping the SER low.

Since we use M-Metric for determining cell state, resistance drift turns into M-Drift. According to [48], M-Drift can be modeled in the same way as resistance drift by

$$M_{drift}(t) = M \times \left(\frac{t}{t_0}\right)^{\alpha_M}, \tag{3}$$

where α_M is the drift exponent of M-Metric. Here, we assume the mean value of the drift exponent in M-Metric (μ_{α_M}) to be $\frac{1}{7}$ of R-Metric(μ_α), as reported in [5, 55].

In order to determine the SER, first the resistance distribution of MLC PCM (Table 1) and Tri-Level PCM (Table 2) should be converted to M distribution. Using Equations (5b) and (2), the M distribution is calculated and presented in Table 3. Table 3a shows the M distribution of MLC PCM and Table 3b shows that for the Tri-Level PCM.

Table 4. SER of Different Levels of MLC PCM and Tri-Level PCM

Time(s)	State Level 3	State Level 2	State Level 1
2^{10}	too small	too small	too small
2^{24}	too small	1.815E-16%	too small
2^{36}	too small	5.408E-07%	2.175E-16%
2^{39}	1.5379E-16%	6.279E-06%	9.7E-14%

(a) SER of MLC PCM

Time(s)	State Level L0	State Level L1
2^{10}	too small	too small
2^{24}	too small	2.0896E-16%
2^{36}	too small	7.5255E-9%
2^{39}	1.5379E-16%	5.4907E-8%

(b) SER of Tri-Level PCM

It can be seen in Table 3 that the normal distributions of resistances in MLC and Tri-Level PCMs are turned into normal distributions in the M domain, but with different deviations. At low resistance levels, the deviation of their corresponding M distribution is smaller.[2]

In order to calculate SER, we used the analytical model presented in [51]. Our model applies to Tables 3a and 3b. Here, our objective is to make the memory system as reliable as the Tri-Level PCM memory proposed in [51]. Thus, we should choose a scrubbing period that results in an SER of less than $3.6E-16\%$ for all levels. According to [51], this SER rate results in a line error rate (uncorrectable errors in a line) of PCM lines, without using soft error correction resources, being less than the line error rate of typical DRAM lines using a single-error correction and double-error detection (SECDED) mechanism. Therefore, our memory is more reliable than typical DRAMs even without using soft error correction resources.

Table 4 shows the SER rate of MLC PCM and Tri-Level PCM levels for different scrubbing periods. We choose the scrubbing period to be 2^{24}s, since in this period, the SER of both MLC and Tri-Level PCMs for all levels are less than $3.6E-16\%$.

2.4 Techniques to Reduce MLC PCM Latency

High access latency is a major issue in MLC PCM. Therefore, in recent years, many studies tried to fill the gap between MLC PCM access latency and DRAM access latency. Throughout this section, we introduce several schemes whose primary goal is to reduce the access latency of PCMs.

The authors in [43] proposed a hybrid main memory consisting of PCM main memory and a DRAM layer between PCM and the Last-Level Cache (LLC) to hide PCM limitations. The DRAM layer helps to bear the long access latency of PCM and improve its lifetime by filtering writes to the same positions. Another work [32] considered a hybrid memory consisting of various technologies (including PCM) with different characteristics and proposed a hardware-based page management scheme that intelligently places pages into different sections of the hybrid memory architecture to maximize performance. The proposed scheme in [1] introduced an OS-based approach to profile and manage pages to hide the high access latency of slow memory in a dual-technology hybrid memory.

Arjomand et al. [3] showed that a considerable percentage of write requests only update a small number of chips, leaving many chips idle. Therefore, it is possible to make use of uninvolved chips to service other read/write requests while the ongoing write is processed.

Morphable memory [40] takes advantage of both MLC PCM density and SLC PCM access latency by partitioning main memory to SLC and MLC regions. Based on the demand of running

[2]The reason is that at low resistances, the resistance of the crystal part of the PCM device matters and plays a significant role in determining M-Metric at low resistances. Since in the range of interest the crystal resistance of the PCM cell is almost constant for different amorphous thicknesses, the variation of M is small at low resistances.

workloads, morphable memory dynamically changes the SLC and MLC types of regions to maximize performance.

To address the read latency of MLC, Hoseinzadeh et al. [19] introduced the line striping method in which half of the lines are stored in the least significant bits of two lines, and the other half in the most significant bits of the two lines. Consequently, half of the lines are read only by one comparison (i.e., SLC read latency). This comes at the cost of doubling the read/write traffic, which leads to more energy consumption and shorter lifetime.

Nair et al. [35] introduced two new techniques for boosting the read latency of PCM using R-Metric: Early Read and Turbo Read. In Early Read, the sense resistance is decreased, and consequently, the sense time will be decreased. However, decreasing the sense resistance leads to wrong interpretation of a cell's state; hence, there must be some additional codes to detect such misinterpretations and repeat the read operation in the presence of such errors. In Turbo Read, the sense voltage (i.e., precharge voltage) is increased in order to decrease the sense time. Increasing the read voltage corresponds to increasing the probability of read disturb (i.e., accidental write to cell), and therefore there is a need for error correction codes (ECCs) to correct read disturb errors in subsequent reads.

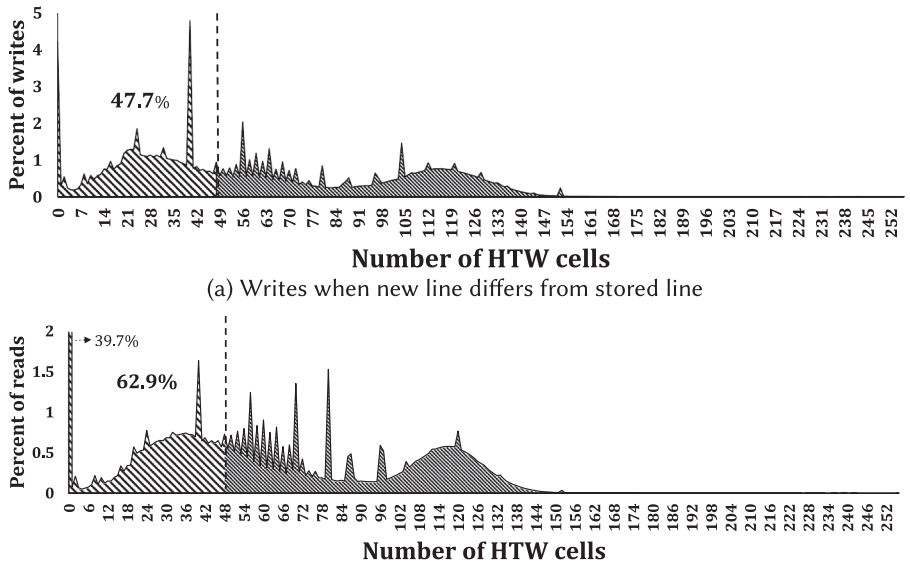
A recent structure proposed in [16] is based on the observation that in some PCM prototypes, data bits of a PCM write request are mapped to multiple cell groups. Cell groups are processed in parallel. Due to programming current and write circuit constraints, a PCM device cannot simultaneously program all data bits in a cell group. Therefore, an XOR mapping function between data bits and cell groups was proposed to evenly distribute bit-flips among cell groups. This method eliminates the asymmetric write latency of cell groups and improves write bandwidth.

The authors in [59] proposed to decouple the cache-line bits to be stored only on MSB or LSB bits of MLC PCM cells, since the MSB and LSB bits have different access latencies. Then, the cache lines that need to be read fast are stored in MSB bits and the cache lines that are needed to be written faster are stored in LSB bits.

Kim et al. [28] proposed to dynamically change the target resistance bands based on the number of intermediate cells in order to decrease the number of iterations per write, since wider resistance bands are correlated with fewer P&V iterations.

Jiang et al. [26] proposed a combination of compression with ECCs. ECCs give the opportunity to write circuit to truncate the last P&V iterations; since at the final write iterations only a few cells are not written properly, ECCs can correct them while reading. The compression helps to tolerate the ECC storage overhead, and additionally converts the MLC write/read to SLC write/read when the compression ratio is ≥ 2. They use the extra storage per line (12.5% storage overhead for hard error correction) provided by the ECP chip to store the extra bits when compression fails (the size of compressed data + compression metadata is larger than the original data size). However, adding ECCs comes at the expense of more bit-flips, which results in more consumed energy and early wearout. In addition, the ECC correction capability is limited (usually 1 bit correction) to make its implementation feasible. Additionally, for the cases when compression fails, the compression metadata with ECCs may not even fit to the extra storage, which makes the compression not an always applicable option.

To the best of our knowledge, among the proposed schemes that address read/write latency at memory line granularity, the methods introduced by Jiang et al. [26] provide the best performance. In addition, many of above techniques (e.g., line stripping, morphable memory) are orthogonal to line-level schemes and can be employed with slight modifications. Since our proposed scheme addresses the access latency of PCM at memory line granularity, we choose the schemes proposed in [26] as the state of the art, for comparison.



(a) Writes when new line differs from stored line

(b) Reads

Fig. 6. Read/write distribution of HTW cells for SPEC-CPU2006 workloads.

3 PROPOSED SCHEME

3.1 Motivation

In MLC PCM, the write latency is determined by the cells that need the most P&V iterations. For the cells to be written to intermediate states, the number of P&V iterations varies; hence, the write operation needs to continue until all the cells are written correctly. We call the cells that are written to intermediate states hard-to-write (HTW) cells, since the write latency is determined by them. The idea behind this article is based on three observations.

Observation 1. *The number of HTW cells in write/read operations is relatively small in typical workloads.*

Figure 6 shows the histogram of HTW cells in write/read operations for 20 SPEC CPU2006 workloads. They are the result of executing 15 billion instructions. Note that the line size of the PCM was set to 512 bits. As Figure 6 shows, on 47.7% of writes and 62.9% of reads, the number of HTW cells is less than or equal to 48 (9.3% of line size).

Observation 2. *Decreasing the number of HTW cells in write operations leads to a smaller average number of P&V iterations per write.*

Figure 7 shows the average required number of P&V iterations to finish the write for different numbers of HTW cells in the line based on our write model. As the figure indicates, the number of P&V iterations for a write (i.e., write latency) is proportional to the number of HTW cells. The reason is clear: the write operation needs to wait for the cell that requires the most iterations. Hence, the smaller the number of HTW cells, the lower the probability that an HTW cell requires a high number of P&V iterations. Therefore, if we could decrease the number of HTW cells for a given write, then the average write latency is decreased as well.

Observation 3. *The dedicated extra storage to correct hard errors per line is not utilized until the line encounters first permanent faults.*

In order to cover the permanent faults of PCM, some extra storage (called ES) per line is usually used to cover such errors. The ES's size is 12.5% of the line size. For a 512-bit line size, the ES is 64 bits (32 cells in 2-bit MLC). However, ES is not utilized until the line encounters first permanent faults. Therefore, this unutilized storage could be used for boosting the write/read operations, before the line faces permanent stuck-at faults.

In order to analyze the utilization of the ES section, first, we assume that the baseline system uses the well-known Error Correction Pointers (ECPs) [47] scheme for hard error correction. ECP is considered as the default correction scheme in many works (e.g., [26, 39]) because of its simplicity and low hardware overhead. In ECP, when a line faces stuck-at faults, its ES section is used to store some (six in the original proposal) ECP entries. Each entry is responsible for correction of one failed cell; consequently, each entry contains a pointer to the position of a faulty cell and a cell that stores the correct value of the failed cell. Since each line has the correction power of six faulty cells, the method is called ECP_6.

When a line encounters more than six faults, ECP entries are not sufficient to cover all; hence, the operating system retires the page containing the faulty line in the memory manager. In this article, similar to other studies, we consider a page size of 4KB (i.e., 64 memory lines).

Another issue that needs to be taken into account is different vulnerability of PCM cells to stuck-at faults, mainly because of process variation [23, 50]. The maximum number of tolerable writes for different PCM cells is expressed as normalized standard deviation (COV) around the mean value [39]. That is, some cells are more prone to stuck-at faults and they may encounter hard errors at the beginning phases of PCM memory lifetime.

Qureshi [39] conducted an extensive study on the utilization of ECP_6 entries and showed that many lines do not face stuck-at faults even at the final stages of PCM lifetime. Figure 8 shows the utilization of ECP_6 entries during the lifetime of a PCM memory with COV = 20% (it is based on Figure 2 in [39]).

Figure 8 contains four curves. The FaultyLine curve shows the probability of a memory line to have at least one stuck-at cell at any point of PCM lifetime. In other words, it shows the percentage of memory lines that were forced to use their ES section for error correction.

The LineFail curve shows the percentage of memory lines with more than six faults, where ECP_6 is not able to recover all faulty cells and, therefore, the line cannot be used.

The PageFail curve shows the probability of pages to have at least one uncorrectable fault, meaning that they have at least one line with more than six faulty cells. Consequently, it shows the percentage of retired memory pages. We define the end of PCM lifetime when the memory capacity drops to half of its size. That is the endpoint where the probability of PageFail is equal to 0.5 (as indicated in the figure).

FirstLineFail is the probability of PCM memory to face the first line failure. To be more specific, it indicates the point where PCM capacity starts to drop. The expected time to first page retirement is when the FirstLineFail curve becomes 0.5.

From Figure 8, we can conclude:

- When memory reaches its half lifetime, only 3.85% of lines have faced stuck-at faults. This implies that more than 96% of lines have not used their ES section when memory passes half of its lifetime (point A in the figure).
- When memory capacity starts to drop, still more than 72% of memory lines are healthy (point B).
- When still 50% of lines have an unused ES section, memory has passed 85.6% of its lifetime (point C).

Table 5. Ternary-to-Binary Conversion

Ternary Values (2 Cells)	Binary Values (3 Bits)
L0L0	000
L0L1	001
L1L2	010
L0L2	011
L1L0	100
L2L0	101
L2L2	110
L2L1	111

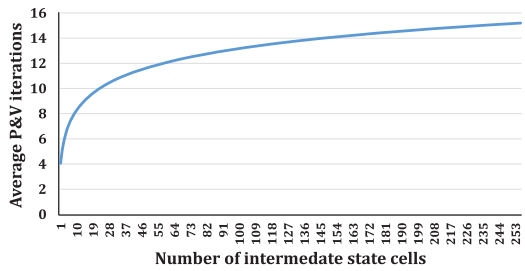


Fig. 7. Average number of P&V iterations versus number of HTW cells.

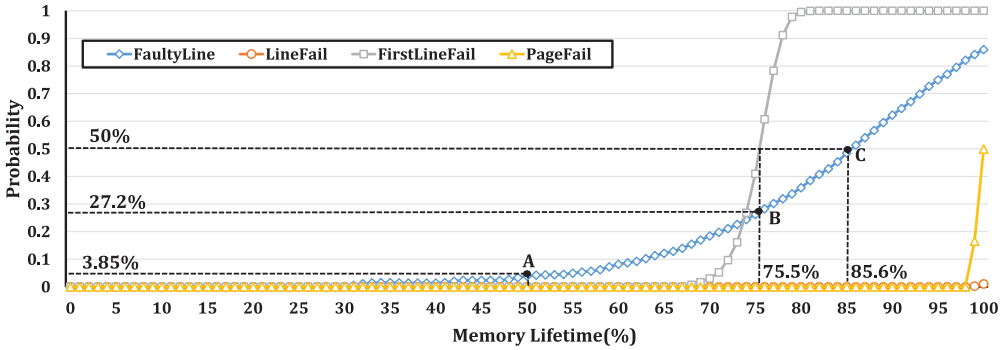


Fig. 8. Failure probability versus time (as percentage of lifetime).

The expected rate of fault-free lines during the entire memory lifetime is therefore given by

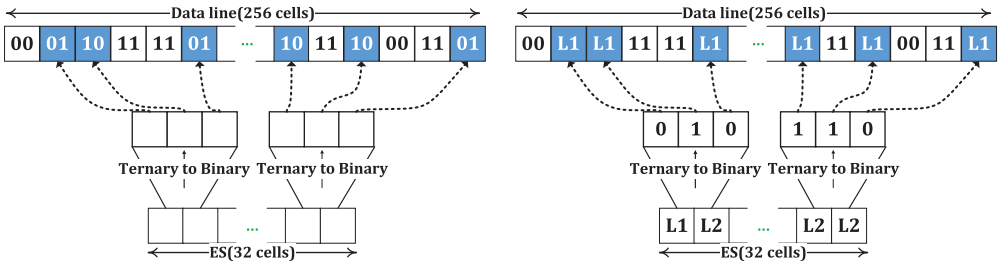
$$Average\_rate\_of\_fault - free\_lines = \int_{t=0}^{100} (1 - FaultyLine(t))dt \approx 83\%. \quad (4)$$

Equation (4) reveals a significant amount of unutilized ES sections in the baseline system, and implies a good opportunity of using ES sections for performance improvement purposes. In the next section, we will show that it is possible to use this storage for performance improvement with a positive effect on memory wearout and energy consumption.

3.2 Relaxed Write/Read (RWR) Method

In our proposed method, Relaxed Write/Read (RWR), the extra storage is used for boosting the write/read performance. The ES is written in Tri-Level format. By using the coding technique proposed in [51], each pair of two Tri-Level cells represent 3 bits. Table 5 shows ternary-to-binary conversion.

Since there are 32 ES cells per line, they could represent 48 bits in ternary format. These 48 bits are used to mitigate 48 HTW cells per line. This is done through a simple one-to-one mapping from an ES bit to an HTW cell. As Figure 9(a) indicates, we assume that the leftmost bit in ES corresponds to the leftmost HTW cell, the one after the leftmost bit in ES corresponds to the one after the leftmost HTW cell in line, and so on. Instead of writing these 48 HTW cells to their exact values, they are written to the intermediate state of Tri-Level format, and their corresponding bits are stored in the ES section to determine their exact value. Consequently, as Figure 9(b) shows, if the mapped bit in the ES section is "0," its corresponding HTW cell is "01"; otherwise, the cell



(a) Indicated mapping in RWR. (b) RWR correction.

Fig. 9. RWR mechanism.

shows "10." In other words, the intermediate state of Tri-Level is used to mark 48 HTW cells and let their corresponding bits be stored in the ES section to determine the exact state ("01" or "10").

In a write operation, up to 48 HTW cells are detected and ES is filled according to the value of these 48 cells, before the write command is issued to the memory array.

In a read operation, the corresponding ES is decoded from ternary to binary to represent the mapped bits; then, the memory controller finds up to 48 leftmost HTW cells of the line and corrects their values by their corresponding bits in the ES. Finally, the final data line is delivered to the requester.

RWR lets up to 48 HTW cells be written in only one iteration, and therefore, by decreasing the number of HTW cells, the average number of P&V iterations is decreased as well. Based on the number of HTW cells in each line, two possible scenarios can be considered:

- If the line has no more than 48 HTW cells, then all of them could be covered by ES bits. Consequently, the write operation is performed using one or two iterations (two iterations if a cell with "11" value exists or if writing to the intermediate state of Tri-Level requires another iteration). Since all line cells are in Tri-Level format, the Tri-Level read mechanism is used for subsequent reads to the line and therefore the read speed is improved. In Tri-Level read, if a data cell is in the first region, its value is "11"; if it is in the third region, its value is "00." For the cells in intermediate region, their corresponding ES bit determines their exact value ("01" or "10"). We call such line fully Tri-Level.
- If the line has more than 48 HTW cells, the 48 leftmost HTW cells are written to the intermediate state of Tri-Level format. Thus, the average number of P&V iterations is decreased. However, the subsequent read operations to the line are issued in MLC mode. In MLC read, the HTW cells that are written to the intermediate state of Tri-Level mode are interpreted as "01" or "10," but their exact value is determined by their corresponding bits in the ES part. The other HTW cells that could not be covered by ES are written exactly (in MLC mode) and hence are fully determined as in MLC read. Since the ES section is in Tri-Level format, in MLC read, if a cell in ES is determined to be "01" or "10," then it is interpreted as the intermediate state of Tri-Level format. We call such line partially Tri-Level.

Using RWR, both read and write latency to the lines with no more than 48 HTW cells are improved significantly. Moreover, for the lines with more than 48 HTW cells, the write latency is still improved. If a line encounters stuck-at failure, then its ES section is used for error correction purposes and the RWR method cannot support the line.

As the memory lines could be in different states, the memory controller needs to track each line's state. Therefore, 2 bits per line are needed to indicate whether the line is fully Tri-Level or partially

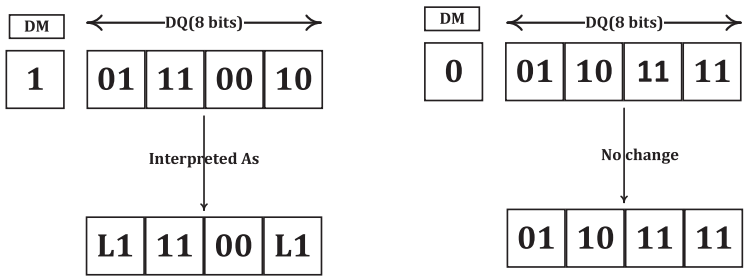


Fig. 10. Reusing DM pins for the RWR method.

Tri-Level or the ES is used for permanent failure correction. Therefore, the storage overhead of the RWR method is about 0.4% (assuming 512-bit memory line size). For example, for a 16GB memory, 64MB storage is needed to track line states. We let these state bits be stored in main memory in MLC mode, and a small cache, called line-state cache, in the memory controller is used to cache them. On every read/write request that is issued to the memory controller, the memory controller adds the request to its corresponding read/write queue and then searches the line-state cache to determine the requested line's state. If a miss occurs, the memory controller takes the state bits from main memory. If the state of the next request to be serviced in any read/write queue is not determined, the memory controller stops servicing any requests from that queue until the state of the request is determined.

The main memory also has a DRAM cache in order to hide the PCM read/write latency. Further, details on CPU & memory organization is explained in Section 4.

To make the RWR method compatible with the DIMM structure, we reuse the Write Data Mask (DM) pins to indicate the cells that need to be written to the intermediate state of the Tri-Level mode. Originally, for a write operation, each DM pin is used to mask its respective data byte (DQ) for any given cycle of the burst. In the RWR method, the DM pin is used to indicate that all intermediate values of its respective DQ should be written to the intermediate state of Tri-Level. Figure 10 shows the usage of the DM pin. If the DM pin is set to "1," then all of the intermediate values of its respective DQ are written to the intermediate state of Tri-Level. Otherwise, the DQ is written in MLC mode. It is possible that in the last burst cycle, in one DQ, some HTW cells need to be written in Tri-Level form while other HTW cells must be written in MLC mode. In such cases, the Tri-Level write in the corresponding DQ is ignored and all HTW cells are written in MLC mode for the respective DQ. Therefore, in the worst case, 45 HTW cells are covered rather than 48 cells.[3]

3.3 RWR Encoder/Decoder

The RWR is a memory-controller-level technique. Therefore, CODEC circuits should be integrated with memory controller hardware. Such circuits should have low overheads in terms of power and area. In this section, we propose the Encoder/Decoder circuits for the RWR method and evaluate their area, power, and latency overheads.

3.3.1 Encoder. The encoding process happens when a new write request is placed to its corresponding write queue in order to be serviced by the memory controller. While the write request is waiting for its turn in the write queue, the encoder circuit calculates the 48 correction bits to be

[3]This condition occurs when DQ has four HTW values; three HTW values are covered by the ES section and can be written in Tri-Level form, while the other HTW value is not covered by ES.

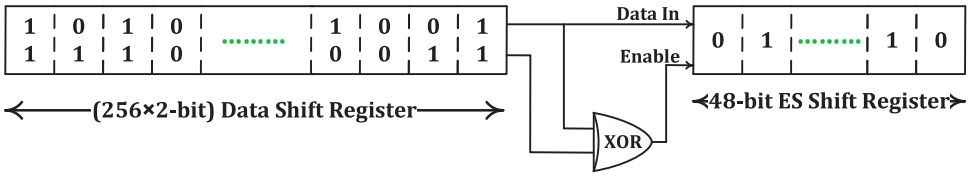


Fig. 11. Encoder circuit.

stored in the ES section in Tri-Level form. Writes are not on the critical path and have relatively high latency. In addition, reads have higher priority than writes for service. Therefore, the average queue latency of writes is sufficient to hide the encoder latency, even if the encoder is relatively slow, and hence, the encoder latency overhead does not have a considerable impact on overall IPC.

Figure 11 shows the encoder circuit. The data is first loaded into a 256×2 shift register in one clock. Then, during each clock, the rightmost cell is checked: if it is an HTW cell, the ES shift register is enabled and the MSB of the rightmost cell is fed to the ES shift register. After each clock, the data shift register is shifted one cell to the right. If the ES shift register is enabled, it is also shifted 1 bit to the right. All of the data cells should be checked; therefore, after (1 + 256) clocks, the ES shift register has the correction bits corresponding to the 48 leftmost HTW cells. Consequently, the latency of the encoder is 64.25ns per line, assuming a 4GHz CPU clock. Further details about the area and energy overheads of the encoder appear in Section 3.3.3.

3.3.2 Decoder. The decoding process begins when a new line is read from main memory by the memory controller. Before the data is passed to the last-level cache, it is first corrected by the decoder circuit.

Because the read requests are in the critical path, the decoder logic should be fast. Consequently, analyzing each data cell should be done in parallel with other cells. Figure 12 shows the decoder circuit. Let's say in each data line, $cell_n$ is the data cell in position n, where $n = 255$ is the leftmost cell index and $n = 0$ shows the rightmost cell. In order to perform parallel operations on data cells, each $cell_n$ should know how many HTW cells are among its left-side cells, that is, $cell_{255} \ldots cell_{n+1}$. Therefore, each HTW cell could determine whether it has a correction bit in the ES section or not, and if yes, it finds the location of its correction bit. In the decoder circuit, first the bits of the cells are passed through XOR gates. The output of XOR gates—called HTW bits—determines whether the cell is HTW or not. Then, a set of carry-lookahead adders is used to sum the HTW bits and form partial sums. The partial sums are in $S_n[i]$ format, where n indicates the number of cells that are in the partial sum, and i is the index of that partial sum. For example, $S_2[127]$ indicates, among $cell_{255}$ and $cell_{254}$, how many HTW cells exist. Then, the partial sums are used to calculate the number of HTW cells on the left side of each cell, called offset. For example, for $cell_{255}$, since it is the leftmost cell, the offset value is 0. For $cell_{252}$, the offset is $S_2[127] + S_1[253]$. In the worst case, the maximum of four add/subtract operations among partial sums should be performed to calculate its offset (e.g., $cell_{224}$).

When each cell determines its offset, its decoding operation can be performed independently. For each HTW cell whose offset is less than 48, a correction bit is considered in the ES section. The cell's offset is used to access its correction bit through a 48-to-1 multiplexer. Based on the cell's correction bit, the corrected data is produced and selected through a 2-to-1 multiplexer.

Compared to the encoder, the decoder circuit poses more hardware overhead and complexity. In the next section, we show that the overhead of both encoder/decoder units is negligible in terms of area, energy consumption, and latency.

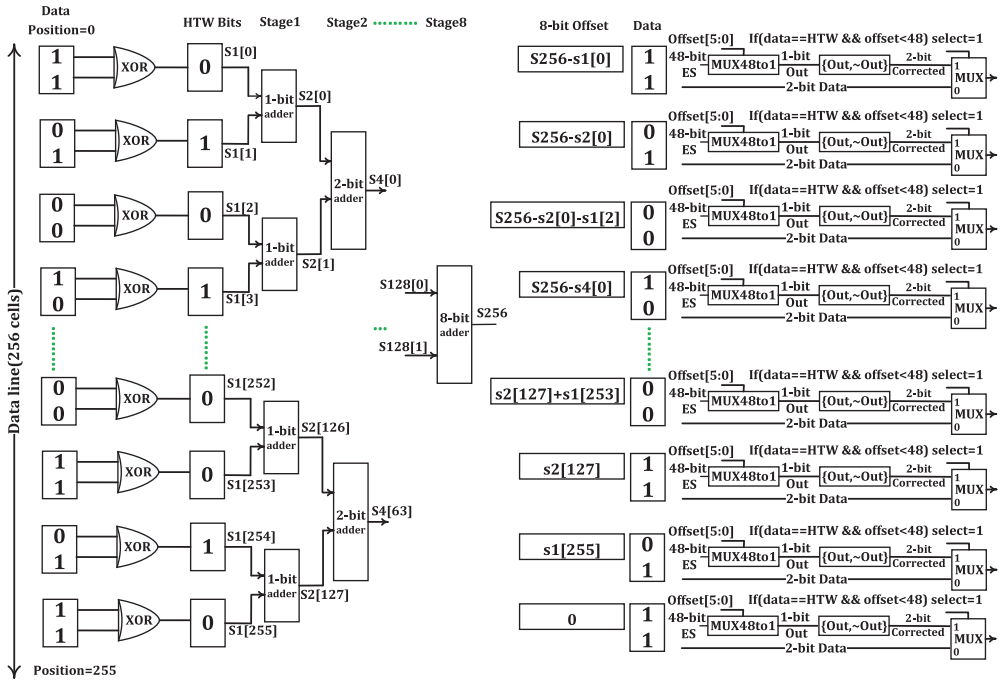


Fig. 12. Decoder circuit.

Table 6. Encoder/Decoder Overhead

Circuit	Latency	Energy	Area
Encoder	64.25ns	0.02pJ/cell	$0.0035mm^2$
Decoder	4.94ns	0.16pJ/cell	$0.036mm^2$

3.3.3 Encoder/Decoder Overhead. We synthesized the encoder/decoder circuits using a 45nm process library. Table 6 presents the encoder/decoder overhead. The encoder latency is mostly hidden since encoding is done when the write request is waiting in the queue. The decoder latency is also negligible compared to the baseline read latency.

The area overhead of encoder and decoder circuits is about 0.3% of the memory controller, based on the $12mm^2$ area overhead of the memory controller reported in [33].

The energy overhead of both encoder and decoder units is also negligible compared to overall dissipated read/write energy per cell.

4 EXPERIMENTAL RESULTS

In this section, evaluation results of the proposed scheme are reported.

4.1 Experimental Settings

We used the gem5 simulator for evaluating our proposal. We modified the simulator to model a PCM-based main memory with a DRAM cache. The detailed baseline parameters can be found in Table 7. For MQ pulse, we conservatively assumed that its latency is the the sum of SET and RESET pulses' duration, and its energy is the sum of SET energy and RESET energy. Note that according to [51], the latency of MQ pulse is shorter than SET duration; therefore, our assumption

Table 7. Baseline Parameters

System	8-core CMP, 4GHz
Processor core	single issue in-order, 32KB iL1, 32KB dL1, 4-way, 2-cycle access latency
L2 cache	2MB, 8-way, LRU, 64B line size, write back, 6-cycle access latency
Line-State cache	16KB, 8-way, LRU, 64B line size, write back, 6-cycle access latency
DRAM Cache	32MB, 16-way, 64B cache line size, 50ns (200 cycles) access latency, write back
Main Memory	LPDDR3-N-1600, 9 × 8b I/O, 18GB PCM (including 2GB ES), 4KB page, 1 channel, 1 rank per channel, 16 banks per rank, 24 entries read/write queues per bank, read priority scheduling (unless WRQ is >80% full), with differential write support
PCM write latency	RESET: 5V, 100μA, 29.7pJ per bit, 50ns operation latency, 125ns iteration latency; SET: 3V, 50μA, 22.5pJ per bit 150ns operation latency, 250ns iteration latency (including T_{off} [21] & verify), MLC Write Parameters: 2-bit MLC [25, 36, 41]; MQ: 52.2pJ, 200ns operation latency, 300ns iteration latency [27, 51]
PCM read latency	tCL/tRP = 12/2 cycles, tRCD = 132 cycles (165ns) for MLC read/24 cycles (30ns) for Tri-Level read/5 cycles(6.25ns) for SLC read

is an upper bound for MQ latency and energy. We took into account the latency overheads for the RWR encoder/decoder, Line-State cache hit/miss, and compression circuits when it is combined with RWR and WT (discussed later) in all of our simulations.

We also employ differential writes [58] to avoid redundant cell writes; therefore, a read operation is needed prior to each write operation. In addition, read priority scheduling is adopted to prioritize the read requests, since the reads are in a critical path. We also assumed there are wear-leveling techniques that evenly distribute writes at intraline (e.g., [47, 63]) and interline (e.g., [42, 49]) levels. Note that the intraline wear-leveling technique rotates data to distribute writes among data cells and the ES section in order to improve lifetime through better wear leveling [47]. Therefore, in the baseline, the ES section cells also wear out. It is shown that in the RWR method, the overall wear rate is decreased with respect to baseline.

We simulated the RWR method with 20 workloads of SPEC CPU2006. All of the results are based on 2B fast-forward, 50M warmup, and 1B real simulation instructions. We compared our scheme against a 2-bit MLC PCM baseline and the Write Truncate (WT) scheme [26]. All results are normalized to that of the 2-bit MLC PCM baseline. Note that the RWR scheme improves the read/write latency, but in the WT scheme the read latency is not improved directly. However, improving the write latency decreases the bank service time; hence, the queueing delay is decreased and the effective read latency is improved as well.

We also evaluated our scheme and WT method with FPC compression. FPC uses frequent pattern compression [2] to capture the most frequent patterns and store them in fewer bits. A study in [26] showed that the FPC hardware overhead is negligible (0.7ns for compression, 1.2ns for decompression). Note that FPC is orthogonal to both RWR and WT schemes. The compatibility of FPC with WT was shown in [26].

When RWR is used with FPC, first a line is compressed. If the compressed size is less than half of the data line size, it could be stored in SLC format. Hence, the subsequent reads to it could be performed in SLC mode. Otherwise, MLC write is needed; therefore, RWR uses the remaining unutilized cells (i.e., ES or data cells) in order to relax accessing intermediate state cells. Although

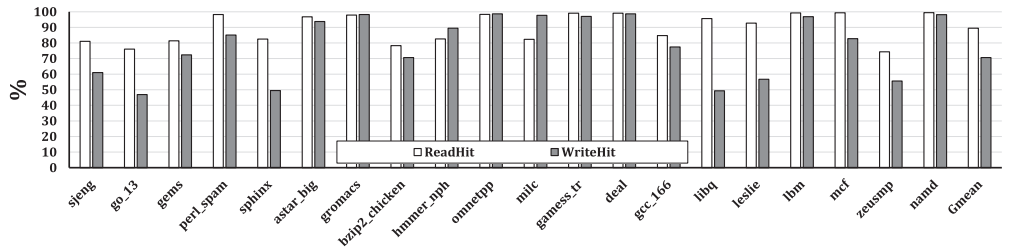


Fig. 13. Hit rate of Line-State cache for write/read requests.

compression may increase the percentage of HTW cells, on the other hand, it usually leaves more unutilized cells to be used by the RWR method.

However, one problem of using compression with RWR is the variable size of data when data is compressed. In FPC compression, metadata embodies the size of compressed data. But when FPC combines with RWR, tag bits might not be written exactly (if they have intermediate state cells approximated by RWR); hence, knowing the exact size of compressed data and correction bits becomes impossible while decoding. Note that this problem also exists when FPC is used with WT.

To address this issue, a specific alignment of data (i.e., metadata in the leftmost cells, correction bits of metadata in the rightmost cells) in the line is used to help the RWR decoder circuit find the corresponding correction bits of the metadata segment. This way, the decoding process is accomplished in two steps. In the first step, the decoder finds correction bits of the metadata segment, corrects metadata, and finds the size of compressed data. After this step, the decoder is able to distinguish data and correction bits. In the second step, the decoder corrects the compressed data. By doing so, the decoding latency is doubled (i.e., 9.88ns), but compared to the advantages offered by compression (e.g., SLC reads), this latency is shown to be tolerable. A more thorough description of this process is given in Appendix C.

The latency of the RWR encoder circuit remains constant when combined with FPC. According to the proposed architecture for the RWR encoder in Section 3.3.1, all 256 cells of a data line are checked. Therefore, the only modification needed is to increase the capacity of the shift register to hold more correction bits.

4.2 Line-State Cache Hit Rate

Figure 13 shows the hit rate of Line-State cache for SPEC2006 workloads. We considered the size of Line-State cache to be 16KB, meaning that it is capable of storing $\frac{16KB}{16Bperpage} = 1{,}024$ pages. This size of Line-State cache is relatively the same as TLB entries in an eight-core processor chip. Therefore, like TLB, its hit rate is expected to be high. As the figure indicates, the hit rate for read and write requests is 89.5% and 70.6%, respectively. For *sphinx, go_13,* and *libq* workloads, the hit rate for write requests is below 50%, but the miss penalty of Line-State cache has a negligible impact on write performance (compared to overall write latency, it is negligible). In addition, the miss penalty hides as a result of improved write latency due to application of the RWR method.

Note that according to the proposed scheme in [26], a Line-State cache is also needed, since the memory controller must be aware of those lines stored in SLC or MLC mode or those that contain faulty cells, to disable WT or FPC. Such information is essential for the memory controller to issue proper read commands and successfully decode data.

Due to the relatively high hit rate of Line-State cache in SPEC2006 workloads, we assumed that the state bits of all memory lines are stored in MLC mode in PCM memory; hence, it imposes 0.4% storage overhead. However, for workloads requiring large memory footprints, Line-State cache

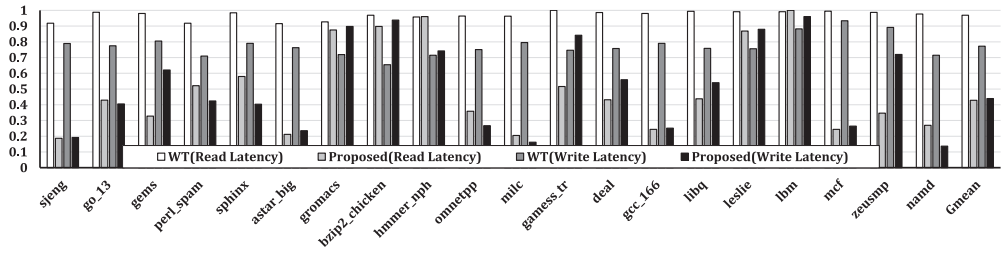


Fig. 14. Read & write latency for WT and RWR (normalized to 2-bit MLC).

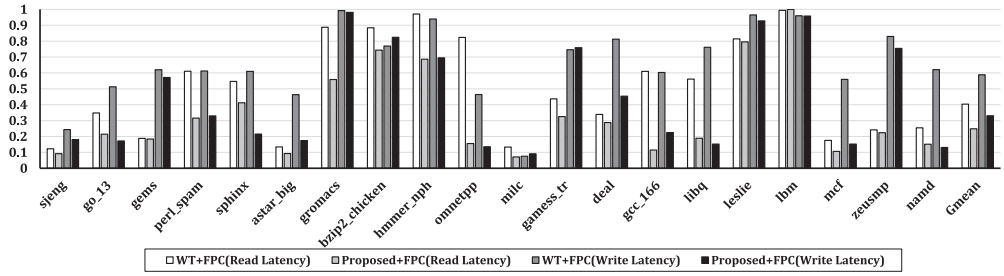


Fig. 15. Read & write latency for WT+FPC and RWR+FPC (normalized to 2-bit MLC).

might negatively impact the overall performance due to an increased miss rate. A poor hit rate might reduce read bandwidth to half since each read operation requires two accesses to memory. One possible solution is to reduce the miss penalty by storing state bits in SLC form in the main memory. Due to the high asymmetry of sense latency for different resistance levels in M-Metric, SLC read has much lower latency than MLC read (6.25ns vs. 165ns); therefore, miss requests will be responded to faster. In this case, storage overhead of state bits becomes 0.8%, which is still negligible, while a miss penalty incurs much lower latency overhead.

4.3 Read and Write Latency

Figure 14 shows the effective read and write latency for RWR and WT schemes normalized to the 2-bit MLC PCM baseline. As the figure shows, by using the RWR method, the effective read and write latency is decreased by 57.2% and 56.1%, respectively. Compared to the WT scheme that reduces the write latency by 22.8% and read latency by 3.2%, the RWR method outperforms WT in terms of read/write latency.

Figure 15 shows the effective read/write latency of the RWR+FPC and WT+FPC schemes. As the figure indicates, applying FPC enhances the overall effective access latency for different workloads. However, for some workloads (e.g., *lbm*), the effective access latency benefit of WT+FPC and RWR+FPC is less than WT and RWR, respectively. The reason in this case is that FPC increases the amount of HTW cells while it mostly fails to store the lines in SLC mode. The combination of RWR+FPC reduces the write latency by 67% and read latency by 75.2%, while WT+FPC reduces the write latency by 41.2% and read latency by 59.7%.

4.4 Write Energy

It is important that improving the read/write latency does not come at the cost of increasing write energy. Moreover, write energy relates to cell wearout; that is, increasing the write energy leads to decreasing PCM lifetime [45]. Figure 16 shows the write energy of RWR, WT, RWR+FPC, and WT+FPC methods normalized to 2-bit MLC PCM baseline. As it shown, the RWR method also

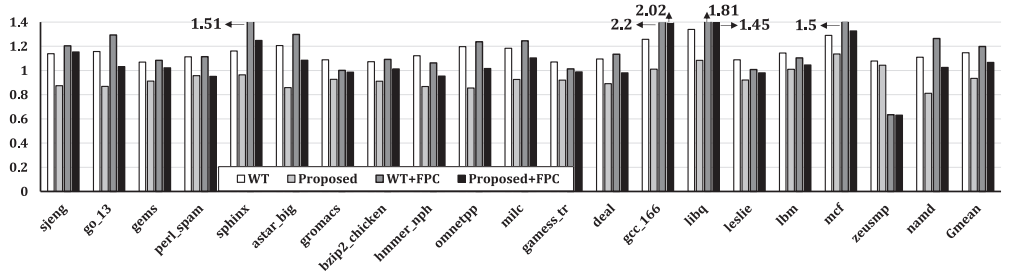


Fig. 16. Write energy (normalized to 2-bit MLC).

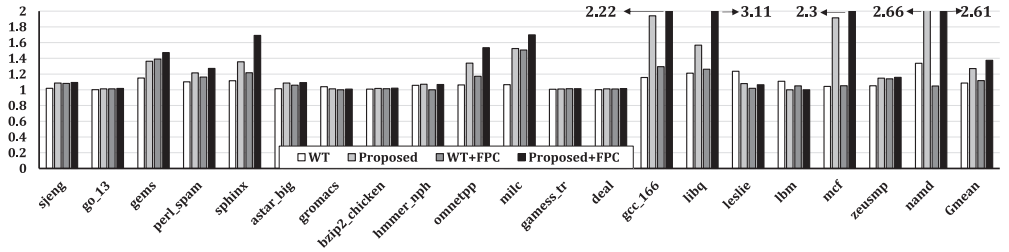


Fig. 17. IPC (normalized to 2-bit MLC).

decreases the write energy by 6.6%. This is because the number of P&V iterations for a large number of HTW cells is decreased. It is also shown that the write latency improvement in the WT method comes at the cost of increased write energy of 14.6%. It is because in WT, only the last P&V iteration of a few cells are skipped, while the ECC section itself consumes more energy. Applying FPC increases the average write energy in both RWR and WT schemes, since it increases the bit-flip rate. To be more specific, using only FPC increases the write energy by 13.7% over the baseline, while, for RWR+FPC and WT+FPC, the increased write energy is 6.7% and 19.8%, respectively.

4.5 Overall Performance (IPC)

The impact of reduced read/write latency has improved the overall system performance in terms of IPC. The improved IPC depends on the reduced read/write latency, which is workload dependent. Figure 17 shows the IPC results of RWR, WT, RWR+FPC, and WT+FPC. WT increases the IPC by 8.6%, while RWR increases the IPC by 26.9%. WT+FPC increases the IPC by 11.7% and RWR+FPC increases the IPC by 37.4%. It should be noted that although in WT+FPC, the effective access latency is close to that of RWR, the IPC benefit of WT+FPC is about half of RWR. It is because in WT+FPC, most of the improved access latency comes from the workloads that are not memory intensive. Therefore, in WT+FPC, the improved access latency does not impact IPC significantly.

4.6 Wearout

Reducing write energy corresponds to enhancing PCM lifetime [4, 45, 60]. We evaluated the effect of our proposed scheme and other implemented methods on memory lifetime. Like [30], since each bank is a separate entity and can be written independently, we focus on determining the lifetime of one bank in terms of the number of possible writes to one memory bank. The final lifetime point for each bank is where the capacity of the bank drops to half. In the evaluated configuration, each bank has 1GB capacity with 64B lines. Like similar studies in [30, 39], we assume the write endurance of each cell follows a normal distribution with a mean of 2^{25} writes and Coefficient of Variation (COV) varying from 10% to 30%. To realize the difference in wearout cost of write pulses

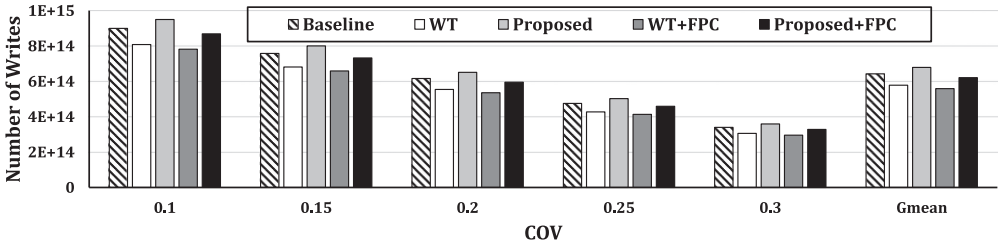


Fig. 18. Lifetime of implemented schemes for different values of COV.

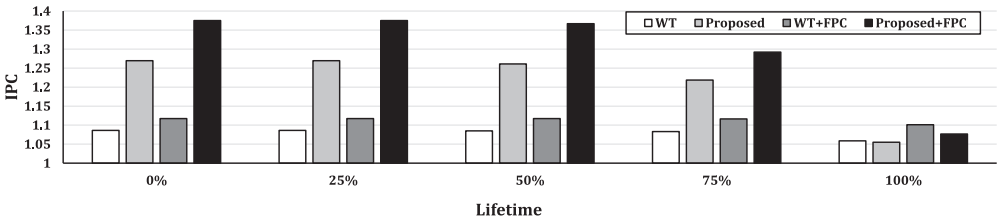


Fig. 19. IPC of implemented schemes at different periods of memory lifetime (normalized to baseline).

(e.g., SET), we weighted their wearout cost based on their energy consumption (as in [4]). This means that SET pulse has the lowest wearout cost and MQ pulse has the highest wearout cost of 2.32 × of SET's.

We first analyzed the behavior of each workload in order to determine the average cell values and energy consumption in each write operation. Then, for each workload, we followed the approach in [47] by creating a test set of 1,024 pages and applying writes to determine the lifetime of test pages. We also consider disabling the proposed scheme for faulty lines in lifetime analysis. Afterward, we scaled this lifetime to a 1GB memory bank. Figure 18 shows the average lifetime of baseline, WT, RWR, WT+FPC, and RWR+FPC memory systems for all workloads. Since in RWR write energy is reduced, lifetime is expected to improve as well. On the other hand, WT, WT+FPC, and RWR+FPC reduce the lifetime in accordance with the increased write energy. Our simulations show that, on average, RWR improves lifetime by 5.6%, while WT, WT+FPC, and RWR+FPC reduce the lifetime by 10.1%, 13.1%, and 3.4%, respectively.

Wearout degrades the efficiency of the proposed scheme (and also other schemes using the ES section) when memory ages and stuck-at faults begin to appear. We assume that the proposed scheme is disabled for lines with faulty cells to simplify the logic. Therefore, memory has less opportunity to use our proposed scheme as time passes and faulty lines appear. To analyze the effect of memory aging on the proposed scheme, we first set the COV factor to 20% (like the similar study in [39]). Figure 19 shows the performance degradation of WT, RWR, WT+FPC, and RWR+FPC schemes at different points of memory lifetime. Note that for compatibility of WT and WT+FPC with hard faults, we implemented the original proposal in [26] by sharing unutilized storage parts among ECP entries and ECC metadata. Note that WT and WT+FPC schemes' performance also degrades when the number of faults per line accumulates and there is not enough space for ECC or FPC metadata to fit in the ES section. As Figure 19 indicates, RWR and other implemented schemes show low performance degradation during the first half of memory lifetime. This is because according to motivation 3 in Section 3.1, less than 4% of memory lines are faulty in this period. The performance benefits of WT, RWR, WT+FPC, and RWR+FPC schemes remain unchanged during the first quarter of memory lifetime. When memory reaches its half lifetime,

the performance gain of WT, RWR, WT+FPC, and RWR+FPC schemes becomes 8.5%, 26.1%, 11.7%, and 36.7%, respectively. As the time passes, the number of faulty lines increases. When 75% of memory lifetime is passed, overall performance for WT, RWR, WT+FPC, and RWR+FPC schemes degrades to 8.3%, 21.8%, 11.6%, and 29.2%, respectively. The reason for the low degradation of WT and WT+FPC schemes at this point is that the ES section and compression could provide enough space for both ECP and ECC metadata to fit. Finally, when memory approaches its end of life, more than 85% of lines are faulty and the performance of RWR and RWR+FPC schemes drastically falls. At this point, WT and WT+FPC schemes outperform other schemes since they tolerate more faults (for the cost of having more complexity). Here, the IPC benefit for WT, RWR, WT+FPC, and RWR+FPC schemes falls to 5.8%, 5.5%, 10.1%, and 7.6%, respectively.

5 RELATED WORK

5.1 Relevant Works to Address Other Aspects of PCM

Despite access latency, PCM memory still suffers from several drawbacks including limited lifetime, soft errors, and high energy consumption. In order to mitigate such drawbacks, several techniques were proposed to make PCM a viable candidate for replacing DRAM.

Lifetime. Zhou et al. [63] introduced a set of techniques to improve PCM lifetime. Differential write is proposed to omit write operations for the cells whose stored value and new value are identical. Another scheme, called row shifting, periodically rotates the contents of the row by 1 byte to fairly spread wearout within cells of a row. Segment swapping swaps the segments of PCM memory to distribute writes among memory lines. In [42, 49], some wear-leveling techniques for lifetime improvement were also provided.

The authors in [46] suggested to use approximate storage since many applications do not need exact data. Approximation of PCM is done by relaxing the target resistance bands, which leads to a smaller number of P&V iterations. Fewer iterations correspond to saving time, energy, and lifetime. Relaxing the target bands exposes data to drift since margins between neighboring states are decreased; therefore, this method is suitable only for approximate applications.

Soft Error. To address the drift problem, authors in [57] suggest to dynamically increase reference resistances for the lines with an outdated last write. Another method called ReadDuo (proposed in [55]) tries to guarantee the reliability of MLC PCM by issuing an M-Metric read if the R-Metric read fails to retrieve data correctly. The authors in [24] highlighted the fact that in addition to the initial resistance and elapsed time, resistance drift strongly depends on temperature. Therefore, they provided a model to take into account temperature as well as other parameters in order to predict cell resistance after drift. This model allows the read circuit to dynamically adjust its reference resistance values, and hence soft error probability is decreased.

Energy. The authors in [30] suggested two novel schemes to make PCM a feasible solution as main memory. In memories, a wide row buffer is used to store contents of adjacent memory lines; hence, requests to the nearby lines are served via row buffer rather than sensing from PCM cells. Based on the fact that in PCM, sensing and buffering are performed by separate peripherals, it is possible to choose a set of narrower row buffers rather than a single row buffer. Narrower write buffers eliminate excess area and energy of sense peripherals, and multiple row buffers exploit locality to coalesce writes, and hence, performance is improved as well. Another scheme is to keep track of modified data at byte/line granularity in the cache level. Consequently, clean bytes/lines are skipped when a write to PCM memory is issued. This results in additional improvement in energy consumption and lifetime of PCM.

Wang et al. [54] introduced a dynamic coding technique to reduce the number of cells with intermediate states, since intermediate levels need more energy to be written. Some cache

management techniques [62], [56] try to organize LLC to decrease the number of writes to PCM memory, because the writes are more expensive in terms of energy, access time, and endurance.

5.2 Similar Challenges in Other Nonvolatile Technologies

It is worth noting that many concepts in PCM exist in most nonvolatile memory technologies. As an example, in flash memory, threshold voltages of flash transistors determine the cell value. For MLC NAND flash, recent work [11] has shown distortion in threshold distribution as a result of charge leakage (similar to soft errors in PCM). Thereby, dynamically changing sense voltage [11] or periodically rewriting data [13] (similar to scrubbing in PCM) might mitigate soft errors. Similar to MLC PCM, writing to MLC NAND flash requires multiple write-read iterations [15, 38]. But flash cell programming can lead to cell-to-cell program interference, which may create errors in neighboring flash cells [12, 14]. Another issue related to flash is read disturbance, that is, accidental writes to neighboring cells in read operations [7, 10, 18]. Hence, recent studies in [8] proposed novel schemes to mitigate program interference and read disturbance at write operations in MLC flash. Another limitation is the lifetime of flash cells (similar to hard failures in PCM) due to expansion of voltage distribution as cycles of program/erases accumulate [9, 34]. As a result, some flash memories (e.g., [52]) rely on dynamically changing sense voltages to a lower bit-error-rate and improve lifetime as wearout increases. In general, one common way to tolerate soft and hard errors in commercial flash devices is to use ECC [7]. It is very similar to reserved extra storage in PCM for error recovery purposes.

STT-RAM is another family of nonvolatile memories that stores data based on the magnetic orientation of its layers. Each STT-RAM cell consists of two layers. The magnetic orientation of constitutive layers defines cell resistance. Recent studies in [6, 61] showed the feasibility of storing MLC in STT-RAM. In recent years, several efforts have been made to use STT-RAM in cache [53] and main memory [29] levels. Like other types of nonvolatile memories, STT-RAM also suffers from soft and hard failures that need careful treatment before it is deployed in commercial products.

Due to common similarities in nonvolatile memories, it is possible to apply some methods of one technology to another. The ideas of this article (i.e., utilizing reserved storage for performance purposes) is general and could possibly be applied to other NVM technologies, which is left for future work.

6 CONCLUSIONS

We proposed the Relaxed Write/Read (RWR) method for improving the performance of MLC PCM. RWR marks the intermediate state cells by programming them to the intermediate state of Tri-Level rather than their exact value in MLC mode, and uses the idle extra storage (dedicated to hard error correction) to determine their exact resistance levels. In addition, RWR writes major amounts of lines in pure Tri-Level format, hence improving their read latency significantly. Our technique improved the read latency by 57.2% and write latency by 56.1%, while it decreased the write energy by 6.6% and improved the IPC by 26.9% over the 2-bit MLC PCM baseline. Also, RWR with a compression technique (FPC) improved the read latency by 75.2% and write latency by 67%, while increasing the write energy by 6.7% and improving the IPC by 37.4% over the 2-bit MLC PCM baseline.

APPENDIXES

A WRITE MODEL FOR MLC PCM

The write model characterizes the write operation. One property of the write model is to describe the percentage of the cells that finish their write after each P&V iteration. Here, we use the model

proposed in [28, 46]. Algorithm 1 describes the write model for programming a cell to intermediate states. As is shown, first a RESET pulse is applied to drive the cell to the amorphous state. Then, the current resistance of the cell is measured. If it is not in the desired target range, then the difference between the current and the target resistance is calculated. A new programming pulse—that is, the SET pulse—is applied to reduce the difference. The process of applying SET pulses continues until the cell resistance is placed within the desired resistance band. Note that by applying each SET pulse, the current resistance changes with a *Gaussian distribution* with the mean value of target resistance. The parameter P determines the average number of required P&V iterations. Higher P values imply that the cell resistance is changed at higher variances after each P&V iteration, and hence, it is less likely that the resistance is placed at an appropriate range after a P&V iteration, resulting in more write iterations for higher values of P.

ALGORITHM 1: MLC write model for intermediate states

Inputs: TB: Target band
R_t: Target resistance (center of TB)
R: Current value of cell resistance
Variables: N_{itr}: Current # of P&V iterations
size: Difference between R_t and R
Outputs: R, N_{itr}

```
1 N_itr = 1;
2 R = N(R_RESET, σ_R^2);
3 while |R_t − R| > TB do
4 |   N_itr + +;
5 |   size = R_t − R;
6 |   R+ = N(size, size.P);
7 end
8 return R, N_itr;
```

In order to fit the write model results to real data [36], the P parameter should be set in a way that results in an average iteration of 4.05 for "01" and "10" values, as reported in [36]. Therefore, we performed a Monte-Carlo simulation by creating a test set of 50 million cells for "01" and "10" values and applying Algorithm 1 for different P values. The amount of P that leads to a 4.05 average number of iterations for "01" and "10" was 0.6 and 0.51, respectively. Figure 20 shows the percentage of programmed cells as a function of P&V iterations in our model and the real prototype in [36]. As shown in the figure, our model tracks the real data with an acceptable error of less than 5%.

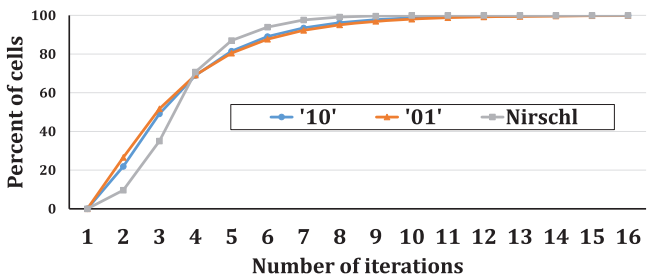


Fig. 20. Percentage of programmed cells versus the number of P&V iterations.

B CONVENTIONAL READ METRIC (R-METRIC)

In the conventional read mechanism (R-Metric), a fixed bias voltage is applied to the cell and the cell's current is sensed. However, the bias voltage should be relatively small to avoid threshold switching (i.e., accidental writes to the cell). It is called R-Metric because it measures the resistance characteristics of the PCM cell. According to the study in [48], the conventional R-Metric suffers from several drawbacks. One drawback is the low signal-to-noise ratio (SNR) of sensed current at high resistance levels, because at high resistance levels, the sensed current is too small and can be affected by small noises.

Another drawback is that R-Metric is crucially affected by drift. To be more specific, according to the study in [48], the I-V equation and resistance equation (for small V) of PCM devices are given as

$$I = 2q\frac{\pi r_{E_{eff}}^2}{\tau_0} N_T \Delta Z e^{\frac{-(E_c - E_f)}{kT}} Sinh\left[\frac{q\Delta ZV}{2KTu_{A_{eff}}}\right] \tag{5a}$$

$$R = \frac{KTu_{A_{eff}}\tau_0 e^{\frac{E_c - E_f}{kT}}}{q^2\pi r_{E_{eff}}^2 N_T \Delta Z^2}. \tag{5b}$$

The parameters used above are similar to those used and described in Section 2.2.1. As Equations (5a) and (5b) indicate, the resistance of PCM (and sensed current I) is an exponential function of activation energy ($E_c - E_f$). Thus, R-Metric is too vulnerable to temporal drift, which makes it inappropriate as a read mechanism [48].

C DATA ALIGNMENT WHEN RWR IS COMBINED WITH FPC

FPC divides data into 32-bit segments and tries to compress each segment individually by finding predefined patterns in it. In FPC, the seven most frequent patterns are defined; hence, if each segment matches with any of these patterns, it could be coded in a few bits. Therefore, 3 bits per segment, called prefix, are needed to indicate what pattern is used, or if the segment is incompressible. This implies that for a 512-bit line, 48 bits of metadata (stored in 24 cells) are needed to store prefixes. In RWR+FPC, a memory line can be in four states:

- **SLC form:** The line is stored in SLC form due to the efficiency of compression (compression ratio ≥ 2). In this state, SLC read operations are issued for subsequent reads of the line.
- **Fully Tri-Level:** The line is in Tri-Level form, since all Hard-to-Write (HTW) cells of the line are relaxed by RWR. Hence, it could be read as in Tri-Level read operation.
- **Partially Tri-Level:** The line is in MLC form, and some HTW cells are relaxed by RWR. Therefore, an MLC read operation is needed to reliably retrieve the data.
- **ES section is used for hard error correction:** The line contains faulty cells, and hence both RWR and FPC are disabled for the line and the ES section is used to cover hard failures. In this state, all reads/writes perform as in the MLC system.

This implies that in RWR+FPC, 2 bits per line are sufficient to track the state of each line (like RWR). Note that when the line does not contain faulty cells, we assume that FPC always applies to the line even if it fails to compress the line properly. This is to simplify the encoding/decoding logic.

Figure 21 shows an example data alignment when RWR is used along with FPC. In this arrangement, metadata is always aligned to the 24 leftmost cells among 256 + 32 = 288 cells of the line. The correction bits region is aligned to the rightmost cells, while compressed data bits are placed between prefix bits and correction bits. Here, correction bits are arranged in some specific manner. In this arrangement, the rightmost correction bit is mapped to the leftmost HTW cell in the line,

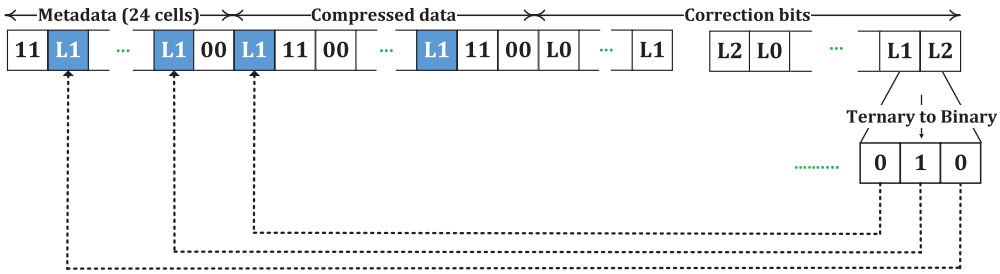


Fig. 21. FPC+RWR line cells' arrangement.

one after the rightmost bit in correction bits corresponds to one after the leftmost HTW cell in the line, and so on (see Figure 21). Such an alignment allows the RWR decoder to easily correct the metadata region. Since the size and placement of metadata cells are always fixed, RWR can detect HTW cells of metadata and their corresponding correction bits and correct metadata. After constructing metadata, the decoding process becomes like the process explained in Section 3.3.2.

Note that the only assumption here is that all HTW cells of the metadata section have one mapped bit in the correction bits section. This assumption is always valid. In the worst case, if all 24 cells of metadata are HTW, at least 16 unutilized cells are needed to store 24 correction bits of the metadata section in Tri-Level form. In the worst case of compression, when all segments fail to compress in FPC, metadata + data occupies 24 + 256 = 280 cells of the line, leaving only 288 − 280 = 8 cells available for RWR. But in this case, according to the FPC compression method [2], the prefixes of all segments are "111." This means in this case, metadata has no HTW cells at all. In other cases, at least one segment is compressed. Compression of each segment at least adds eight more unutilized cells in FPC compression. That is, in other cases with at least one segment being compressed, there are at least 8 + 8 = 16 unutilized cells in the line, sufficient to cover the whole metadata section. Therefore, each HTW cell of metadata always has 1 mapped bit in RWR.

REFERENCES

[1] N. Agarwal and T. F. Wenisch. 2017. Thermostat: Application-transparent page management for two-tiered main memory. In *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'17)*. ACM, New York, 631–644. DOI: http://dx.doi.org/10.1145/3037697.3037706

[2] A. R. Alameldeen and D. A. Wood. 2004. Adaptive cache compression for high-performance processors. In *Proceedings of the 31st Annual International Symposium on Computer Architecture, 2004.* 212–223. DOI: http://dx.doi.org/10.1109/ISCA.2004.1310776

[3] M. Arjomand, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das. 2016. Boosting access parallelism to PCM-based main memory. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA'16)*. IEEE Press, Piscataway, NJ, 695–706. DOI: http://dx.doi.org/10.1109/ISCA.2016.66

[4] M. Asadinia, M. Arjomand, and H. Sarbazi-Azad. 2015. Variable resistance spectrum assignment in phase change memory systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 11 (Nov. 2015), 2657–2670. DOI: http://dx.doi.org/10.1109/TVLSI.2014.2363102

[5] A. Athmanathan, M. Stanisavljevic, J. Cheon, S. Kang, C. Ahn, J. Yoon, M. Shin, T. Kim, N. Papandreou, H. Pozidis, and E. Eleftheriou. 2014. A 6-bit drift-resilient readout scheme for multi-level phase-change memory. In *Proceedings of the 2014 IEEE Asian Solid-State Circuits Conference (A-SSCC'14)*. 137–140. DOI: http://dx.doi.org/10.1109/ASSCC.2014.7008879

[6] X. Bi, M. Mao, D. Wang, and H. Li. 2013. Unleashing the potential of MLC STT-RAM caches. In *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'13)*. 429–436. DOI: http://dx.doi.org/10.1109/ICCAD.2013.6691153

[7] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu. 2017. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. *Proceedings of the IEEE* 105, 9 (Sept. 2017), 1666–1704. DOI: http://dx.doi.org/10.1109/JPROC.2017.2713127

[8] Y. Cai, S. Ghose, Y. Luo, K. Mai, O. Mutlu, and E. F. Haratsch. 2017. Vulnerabilities in MLC NAND flash memory programming: Experimental analysis, exploits, and mitigation techniques. In *Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA'17)*. 49–60. DOI: http://dx.doi.org/10.1109/HPCA.2017.61

[9] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai. 2013. Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling. In *Proceedings of the 2013 Design, Automation Test in Europe Conference Exhibition (DATE'13)*. 1285–1290. DOI: http://dx.doi.org/10.7873/DATE.2013.266

[10] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu. 2015. Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery. In *Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 438–449. DOI: http://dx.doi.org/10.1109/DSN.2015.49

[11] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu. 2015. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA'15)*. 551–563. DOI: http://dx.doi.org/10.1109/HPCA.2015.7056062

[12] Y. Cai, O. Mutlu, E. F. Haratsch, and K. Mai. 2013. Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation. In *Proceedings of the 2013 IEEE 31st International Conference on Computer Design (ICCD'13)*. 123–130. DOI: http://dx.doi.org/10.1109/ICCD.2013.6657034

[13] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S. Unsal, and K. Mai. 2012. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In *Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD'12)*. 94–101. DOI: http://dx.doi.org/10.1109/ICCD.2012.6378623

[14] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, O. Unsal, A. Cristal, and K. Mai. 2014. Neighbor-cell assisted error correction for MLC NAND flash memories. *SIGMETRICS Performance Evaluation Review* 42, 1 (June 2014), 491–504. DOI: http://dx.doi.org/10.1145/2637364.2591994

[15] R. A. Cernea, L. Pham, F. Moogat, S. Chan, B. Le, Y. Li, S. Tsao, T. Y. Tseng, K. Nguyen, J. Li, J. Hu, J. H. Yuh, C. Hsu, F. Zhang, T. Kamei, H. Nasu, P. Kliza, K. Htoo, J. Lutze, Y. Dong, M. Higashitani, J. Yang, H. S. Lin, V. Sakhamuri, A. Li, F. Pan, S. Yadala, S. Taigor, K. Pradhan, J. Lan, J. Chan, T. Abe, Y. Fukuda, H. Mukai, K. Kawakami, C. Liang, T. Ip, S. F. Chang, J. Lakshmipathi, S. Huynh, D. Pantelakis, M. Mofidi, and K. Quader. 2009. A 34 MB/s MLC write throughput 16 Gb NAND With all bit line architecture on 56 nm technology. *IEEE Journal of Solid-State Circuits* 44, 1 (Jan. 2009), 186–194. DOI: http://dx.doi.org/10.1109/JSSC.2008.2007152

[16] Y. Du, M. Zhou, B. R. Childers, D. Mossé, and R. Melhem. 2013. Bit mapping for balanced PCM cell programming. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*. ACM, New York, 428–439. DOI: http://dx.doi.org/10.1145/2485922.2485959

[17] J. Fan, S. Jiang, J. Shu, Y. Zhang, and W. Zhen. 2013. Aegis: Partitioning data block for efficient recovery of stuck-at-faults in phase change memory. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-46'13)*. ACM, New York, 433–444. DOI: http://dx.doi.org/10.1145/2540708.2540745

[18] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf. 2009. Characterizing flash memory: Anomalies, observations, and applications. In *Proceedings of the 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'09)*. 24–33. DOI: http://dx.doi.org/10.1145/1669112.1669118

[19] M. Hoseinzadeh, M. Arjomand, and H. Sarbazi-Azad. 2015. SPCM: The striped phase change memory. *ACM Transactions on Architecture and Code Optimization* 12, 4, Article 38 (Nov. 2015), 25 pages. DOI: http://dx.doi.org/10.1145/2829951

[20] Y. N. Hwang, C. Y. Um, J. H. Lee, C. G. Wei, H. R. Oh, G. T. Jeong, H. S. Jeong, C. H. Kim, and C. H. Chung. 2010. MLC PRAM with SLC write-speed and robust read scheme. In *Proceedings of the 2010 Symposium on VLSI Technology*. 201–202. DOI: http://dx.doi.org/10.1109/VLSIT.2010.5556227

[21] Y. N. Hwang, C. Y. Um, J. H. Lee, C. G. Wei, H. R. Oh, G. T. Jeong, H. S. Jeong, C. H. Kim, and C. H. Chung. 2010. MLC PRAM with SLC write-speed and robust read scheme. In *Proceedings of the 2010 Symposium on VLSI Technology*. 201–202. DOI: http://dx.doi.org/10.1109/VLSIT.2010.5556227

[22] D. Ielmini, A. L. Lacaita, and D. Mantegazza. 2007. Recovery and drift dynamics of resistance and threshold voltages in phase-change memories. *IEEE Transactions on Electron Devices* 54, 2 (Feb. 2007), 308–315. DOI: http://dx.doi.org/10.1109/TED.2006.888752

[23] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda. 2010. Dynamically replicated memory: Building reliable systems from nanoscale resistive memories. In *Proceedings of the 15th Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems (ASPLOS XV'10)*. ACM, New York, 3–14. DOI: http://dx.doi.org/10.1145/1736020.1736023

[24] M. Jalili, M. Arjomand, and H. S. Azad. 2014. A reliable 3D MLC PCM architecture with resistance drift predictor. In *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 204–215. DOI: http://dx.doi.org/10.1109/DSN.2014.31

[25] L. Jiang, B. Zhao, J. Yang, and Y. Zhang. 2014. A low power and reliable charge pump design for phase change memories. In *Proceedings of the 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA'14)*. 397–408. DOI: http://dx.doi.org/10.1109/ISCA.2014.6853194

[26] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers. 2012. Improving write operations in MLC phase change memory. In *Proceedings of the IEEE International Symposium on High-Performance Comp Architecture*. 1–10. DOI: http://dx.doi.org/10.1109/HPCA.2012.6169027

[27] D. H. Kang, J. H. Lee, J. H. Kong, D. Ha, J. Yu, C. Y. Um, J. H. Park, F. Yeung, J. H. Kim, W. I. Park, Y. J. Jeon, M. K. Lee, Y. J. Song, J. H. Oh, G. T. Jeong, and H. S. Jeong. 2008. Two-bit cell operation in diode-switch phase change memory cells with 90nm technology. In *Proceedings of the 2008 Symposium on VLSI Technology*. 98–99. DOI: http://dx.doi.org/10.1109/VLSIT.2008.4588577

[28] Y. Kim, S. Yoo, and S. Lee. 2016. Improving write performance by controlling target resistance distributions in MLC PRAM. *ACM Transactions on Design Automation of Electronic Systems* 21, 2, Article 23 (Jan. 2016), 27 pages. DOI: http://dx.doi.org/10.1145/2820610

[29] E. Kltrsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu. 2013. Evaluating STT-RAM as an energy-efficient main memory alternative. In *Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'13)*. 256–267. DOI: http://dx.doi.org/10.1109/ISPASS.2013.6557176

[30] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger. 2009. Architecting phase change memory as a scalable dram alternative. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*. ACM, New York, 2–13. DOI: http://dx.doi.org/10.1145/1555754.1555758

[31] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller. 2003. Energy management for commercial servers. *Computer* 36, 12 (Dec. 2003), 39–48. DOI: http://dx.doi.org/10.1109/MC.2003.1250880

[32] Y. Li, S. Ghose, J. Choi, J. Sun, H. Wang, and O. Mutlu. 2017. Utility-based hybrid memory management. In *Proceedings of the 2017 IEEE International Conference on Cluster Computing (CLUSTER'17)*. 152–165. DOI: http://dx.doi.org/10.1109/CLUSTER.2017.130

[33] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi. 2012. Scale-out processors. In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA'12)*. IEEE Computer Society, Washington, DC, 500–511. http://dl.acm.org/citation.cfm?id=2337159.2337217

[34] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu. 2016. Enabling accurate and practical online Flash channel modeling for modern MLC NAND flash memory. *IEEE Journal on Selected Areas in Communications* 34, 9 (Sept. 2016), 2294–2311. DOI: http://dx.doi.org/10.1109/JSAC.2016.2603608

[35] P. J. Nair, C. Chou, B. Rajendran, and M. K. Qureshi. 2015. Reducing read latency of phase change memory via early read and turbo read. In *Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA'15)*. 309–319. DOI: http://dx.doi.org/10.1109/HPCA.2015.7056042

[36] T. Nirschl, J. B. Philipp, T. D. Happ, G. W. Burr, B. Rajendran, M. H. Lee, A. Schrott, M. Yang, M. Breitwisch, C. F. Chen, E. Joseph, M. Lamorey, R. Cheek, S. H. Chen, S. Zaidi, S. Raoux, Y. C. Chen, Y. Zhu, R. Bergmann, H. L. Lung, and C. Lam. 2007. Write strategies for 2 and 4-bit multi-level phase-change memory. In *Proceedings of the 2007 IEEE International Electron Devices Meeting*. 461–464. DOI: http://dx.doi.org/10.1109/IEDM.2007.4418973

[37] N. Papandreou, A. Sebastian, A. Pantazi, M. Breitwisch, C. Lam, H. Pozidis, and E. Eleftheriou. 2011. Drift-resilient cell-state metric for multilevel phase-change memory. In *Proceedings of the 2011 IEEE International Electron Devices Meeting (IEDM'11)*. 3.5.1–3.5.4. DOI: http://dx.doi.org/10.1109/IEDM.2011.6131482

[38] K. T. Park, M. Kang, D. Kim, S. W. Hwang, B. Y. Choi, Y. T. Lee, C. Kim, and K. Kim. 2008. A zeroing cell-to-cell interference page architecture with temporary LSB storing and parallel MSB program scheme for MLC NAND Flash memories. *IEEE Journal of Solid-State Circuits* 43, 4 (April 2008), 919–928. DOI: http://dx.doi.org/10.1109/JSSC.2008.917558

[39] M. K. Qureshi. 2011. Pay-as-you-go: Low-overhead hard-error correction for phase change memories. In *Proceedings of the 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'11)*. 318–328.

[40] M. K. Qureshi, M. M. Franceschini, L. A. Lastras-Montaño, and J. P. Karidis. 2010. Morphable memory system: A robust architecture for exploiting multi-level phase change memories. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. ACM, New York, 153–162. DOI: http://dx.doi.org/10.1145/1815961.1815981

[41] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-Montao. 2010. Improving read performance of phase change memories via write cancellation and write pausing. In *Proceedings of the 16th International Symposium on High-Performance Computer Architecture (HPCA'10)*. 1–11. DOI: http://dx.doi.org/10.1109/HPCA.2010.5416645

[42] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali. 2009. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42'09)*. ACM, New York, 14–23. DOI: http://dx.doi.org/10.1145/1669112.1669117

[43] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. 2009. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*. ACM, New York, 24–33. DOI: http://dx.doi.org/10.1145/1555754.1555760

[44] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y. C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S. H. Chen, H. L. Lung, and C. H. Lam. 2008. Phase-change random access memory: A scalable technology. *IBM Journal of Research and Development* 52, 4.5 (July 2008), 465–479. DOI: http://dx.doi.org/10.1147/rd.524.0465

[45] M. Hossein Samavatian, M. Arjomand, R. Bashizade, and H. Sarbazi-Azad. 2015. Architecting the last-level cache for GPUs using STT-RAM technology. *ACM Transactions on Design Automation of Electronic Systems* 20, 4, Article 55 (Sept. 2015), 24 pages. DOI: http://dx.doi.org/10.1145/2764905

[46] A. Sampson, J. Nelson, K. Strauss, and L. Ceze. 2013. Approximate storage in solid-state memories. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-46'13)*. ACM, New York, 25–36. DOI: http://dx.doi.org/10.1145/2540708.2540712

[47] S. Schechter, G. H. Loh, K. Strauss, and D. Burger. 2010. Use ECP, Not ECC, for hard failures in resistive memories. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. ACM, New York, 141–152. DOI: http://dx.doi.org/10.1145/1815961.1815980

[48] A. Sebastian, N. Papandreou, A. Pantazi, H. Pozidis, and E. Eleftheriou. 2011. Non-resistance-based cell-state metric for phase-change memory. *Journal of Applied Physics* 110, 8 (2011), 084505. DOI: http://dx.doi.org/10.1063/1.3653279

[49] N. H. Seong, D. H. Woo, and H.-H. S. Lee. 2010. Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. ACM, New York, 383–394. DOI: http://dx.doi.org/10.1145/1815961.1816014

[50] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee. 2010. SAFER: Stuck-at-fault error recovery for memories. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-43'10)*. IEEE Computer Society, Washington, DC, 115–124. DOI: http://dx.doi.org/10.1109/MICRO.2010.46

[51] N. H. Seong, S. Yeo, and H.-H. S. Lee. 2013. Tri-level-cell phase change memory: Toward an efficient and reliable memory system. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*. ACM, New York, 440–451. DOI: http://dx.doi.org/10.1145/2485922.2485960

[52] H. Shim, S. S. Lee, B. Kim, N. Lee, D. Kim, H. Kim, B. Ahn, Y. Hwang, H. Lee, J. Kim, Y. Lee, H. Lee, J. Lee, S. Chang, J. Yang, S. Park, S. Aritome, S. Lee, K. O. Ahn, G. Bae, and Y. Yang. 2011. Highly reliable 26nm 64Gb MLC E2NAND (Embedded-ECC amp; Enhanced-efficiency) flash memory with MSP (Memory signal processing) controller. In *Proceedings of the 2011 Symposium on VLSI Technology - Digest of Technical Papers*. 216–217.

[53] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*. 50–61. DOI: http://dx.doi.org/10.1109/HPCA.2011.5749716

[54] J. Wang, X. Dong, G. Sun, D. Niu, and Y. Xie. 2011. Energy-efficient multi-level cell phase-change memory system with data encoding. In *Proceedings of the 2011 IEEE 29th International Conference on Computer Design (ICCD'11)*. IEEE Computer Society, Washington, DC, 175–182. DOI: http://dx.doi.org/10.1109/ICCD.2011.6081394

[55] R. Wang, Y. Zhang, and J. Yang. 2016. 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'16)*. 203–214. DOI: http://dx.doi.org/10.1109/DSN.2016.27

[56] Z. Wang, S. Shan, T. Cao, J. Gu, Y. Xu, S. Mu, Y. Xie, and D. A. Jiménez. 2013. WADE: Writeback-aware dynamic cache management for NVM-based main memory system. *ACM Transactions on Architecture and Code Optimization* 10, 4, Article 51 (Dec. 2013), 21 pages. DOI: http://dx.doi.org/10.1145/2555289.2555307

[57] W. Xu and T. Zhang. 2010. Using time-aware memory sensing to address resistance drift issue in multi-level phase change memory. In *Proceedings of the 2010 11th International Symposium on Quality Electronic Design (ISQED'10)*. 356–361. DOI: http://dx.doi.org/10.1109/ISQED.2010.5450549

[58] B. D. Yang, J. E. Lee, J. S. Kim, J. Cho, S. Y. Lee, and B. G. Yu. 2007. A low power phase-change random access memory using a data-comparison write scheme. In *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems*. 3014–3017. DOI: http://dx.doi.org/10.1109/ISCAS.2007.377981

[59] H. Yoon, J. Meza, N. Muralimanohar, N. P. Jouppi, and O. Mutlu. 2014. Efficient data mapping and buffering techniques for multilevel cell phase-change memories. *ACM Transactions on Architecture and Code Optimization* 11, 4, Article 40 (Dec. 2014), 25 pages. DOI: http://dx.doi.org/10.1145/2669365

[60] L. Zhang, B. Neely, D. Franklin, D. Strukov, Y. Xie, and F. T. Chong. 2016. Mellow writes: Extending lifetime in resistive memories through selective slow write backs. In *Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA'16)*. 519–531. DOI: http://dx.doi.org/10.1109/ISCA.2016.52

[61] Y. Zhang, L. Zhang, W. Wen, G. Sun, and Y. Chen. 2012. Multi-level cell STT-RAM: Is it realistic or just a dream? In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'12)*. 526–532.

[62] M. Zhou, Y. Du, B. Childers, R. Melhem, and D. Mossé. 2012. Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems. *ACM Transactions on Architecture and Code Optimization* 8, 4, Article 53 (Jan. 2012), 21 pages. DOI: http://dx.doi.org/10.1145/2086696.2086732

[63] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. 2009. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*. ACM, New York, 14–23. DOI: http://dx.doi.org/10.1145/1555754.1555759

Received May 2017; revised November 2017; accepted December 2017