ReadDuo: Constructing Reliable MLC Phase Change Memory through Fast and Robust Readout

Rujia Wang† Youtao Zhang§ Jun Yang†
† *Electrical and Computer Engineering Department* § *Computer Science Department*
University of Pittsburgh
{*ruw16,youtao,juy9*}*@pitt.edu*

***Abstract*—Phase change memory (PCM) has emerged as a promising non-volatile memory technology. Multi-level cell (MLC) PCM, while effectively reducing per bit fabrication cost, suffers from resistance drift based soft errors. It is challenging to construct reliable MLC chips that achieve high performance, high storage density, and low energy consumption simultaneously.**

In this paper, we propose ReadDuo, a fast and robust readout solution to address resistance drift in MLC PCM. We first integrate fast current sensing and resistance drift resilient voltage sensing, which exposes performance optimization opportunities without sacrificing reliability. We then devise last writes tracking and selective different write schemes to minimize performance and energy consumption overhead in scrubbing. Our experimental results show that ReadDuo achieves 37% improvement on average over existing solutions when considering performance, energy consumption, and storage density all together.

***Keywords*-Phase Change Memories; Multi-level Cell; Resistance Drift**

I. Introduction

Phase Change Memory (PCM) [20] is an emerging memory technology that takes advantage of the stable resistance states of phase change material (e.g., GST) to record data. A PCM cell, when having currents injected through the cell, can be programmed to either fully crystalline state or fully amorphous state, ranging from several kilo Ωs to several million Ωs [21]. Given the large resistance difference between these two states, MLC (multi-level cell) PCM has been devised to utilize the middle resistance states to store multiple bits in one cell. As a comparison, SLC (single-level cell) PCM uses only full crystalline state and full amorphous state, and stores only one bit in each cell.

The resistance of a PCM cell increases after write, which is referred to as *resistance drift*. While both SLC PCM and MLC PCM have resistance drift, the latter is more vulnerable because its middle resistance states are not as stable as the fully crystalline state and the resistance range of each MLC state is often very tight. Resistance drift has been identified as the main source of soft errors in MLC PCM and a major obstacle that restricts MLC PCM from wide adoption.

Designs have been proposed to mitigate the resistance drift in MLC PCM. Efficient scrubbing [2] periodically scans MLC lines for resistance drift errors and, if found, rewrites all the cells in a drifted line after correction. This scheme tends to consume large memory bandwidth and write energy if we want to match the reliability of MLC PCM chips to that of today's DRAM chips. Tri-Level-Cell (TLC) [26] removes the most drift-prone state in four-level MLC design, making a trade off between density and reliability. Helmet [34] was proposed to adopt more precise MLC write control and data encoding to mitigate resistance drift. M-metric [23] is a recently proposed readout metric that conducts drift resilient voltage sensing (rather than traditional current sensing, referred to as R-metric). A M-metric based readout scheme requires much longer latency to differentiate the stored data. In summary, it remains challenging to design a drift resilient scheme that achieves high performance and low energy consumption while maintaining the same density and endurance as those in drift-free MLC PCM.

In this paper, we address the challenge using a hybrid readout solution that integrates voltage sensing and current sensing. We strive to achieve the best tradeoff among performance, energy consumption, storage density and chip lifetime. We summarize our contributions as follows.

- We propose ReadDuo, a hybrid readout solution that integrates fast current sensing and resistance drift resilient voltage sensing. In the *Hybrid* design, R-metric sensing helps to improve read performance while M-metric sensing helps to meet DRAM reliability with low overhead scrubbing.
- We propose *Last Writes Tracking* and *Selective Rewrite* schemes to minimize performance and energy consumption overhead in scrubbing. *Last Write Tracking* enables relaxed M-metric scrubbing without sacrificing R-sensing reliability. *Selective Rewrite* reduces the average number of cell writes in most write operations and save energy consumption.
- We evaluate the proposed schemes with comparison to the state-of-the-art. A new metric EDAP (Energy Delay Area Product) is used to evaluate the trade off among energy consumption, performance and storage density. Our results show that, on average, ReadDuo achieves 37% improvements over existing schemes.

In the rest of the paper, we briefly review the background in Section II. We elaborate the design details in Section III. The experiment methodology and results are discussed in Section IV and Section V, respectively. We discuss more related work in Section VI and conclude the paper in Section VII.

II. Background

A. Multi-level Phase Change Memory

PCM takes advantage of the stable states of chalcogenide material such as GST [35], [12] to record data. MLC PCM [32], e.g., a 2-bit MLC PCM in Figure 1, partitions the resistance range between full crystalline state (several kilo Ωs) and full amorphous state (several million Ωs) to four resistance subranges, i.e., states.

Reading MLC PCM needs to compare a cell's resistance to three reference cells (denoted as $\texttt{Ref}_{1/2/3}$) in two rounds — the resistance is first compared to $\texttt{Ref}_2$ and then, based on the comparison results, compared to either $\texttt{Ref}_1$ or $\texttt{Ref}_3$. The stored data is determined based on the comparison results at two steps.

Writing MLC PCM often adopts iterative program and verify (P&V) strategy. A to-be-written cell is always RESET to amorphous state '00' and undertaken a series of SET pulse to program the cell to the target resistance range. Writing a MLC PCM often needs to write its resistance to a narrower range between two reference cells, leaving a small resistance interval as guard band.

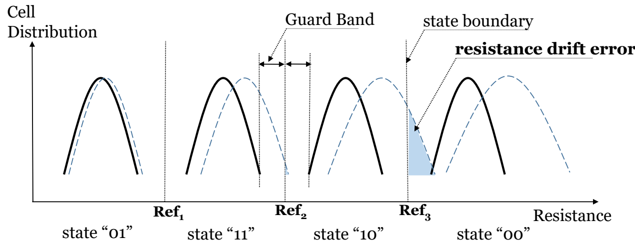


Figure 1: MLC PCM and resistance drift errors (Solid distribution line denotes resistance distribution at t0; Dashed distribution line denotes resistance distribution at t (t>t0)).

B. PCM I-V Characteristics and Readout Metrics

Figure 2(a) shows a typical PCM I-V curve. SET and RESET operations require relatively large current and voltage, as shown in the figure. Read operation usually falls in the left below section, i.e, low field. Low field electrical resistance is typically used to quantify the programmed cell state. A read voltage cannot exceed the V_{th} as otherwise, the resistance of the cell may drop significantly, resulting threshold switching phenomenon and may disturb cell state [23].

Figure 2(b) plots the low field I-V characteristic of MLC PCM cells when they have different amount of amorphous material (U_A represents material thickness). The larger the U_A is, the higher resistance the cell has.

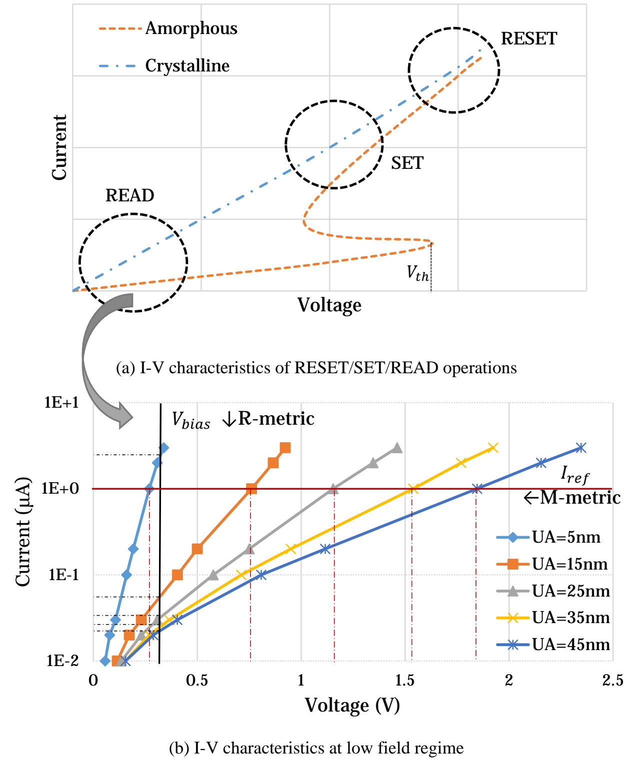


Figure 2: The I-V curve and PCM operations.

1) R-metric: A Current based Readout Metric: To readout the data stored in a MLC cell, we conventionally use current sensing to compare the resistance of the cell to the reference cell. By applying a small voltage V_{bias} to MLC cells, the read circuit compares the sensed currents with reference currents and determine the stored values based on the comparison results [16]. This is referred to as R-metric based sensing, or R-sensing, as it tests the resistance characteristic of a cell.

Recent studies [23], [7] showed that R-metric is a strong function of the activation energy, which is the fundamental material characteristic that change over time, and leads to temporal *resistance drift* [15], i.e., the resistance of a MLC cell, in particular if the cell was programmed to a middle state, increases over time. For the example in Figure 1, if the resistance of a cell in '01' state drifts above the resistance of $\texttt{Ref}_3$, it is readout as '00', leading to drift errors.

It is also evident that R-metric has a very low signal-to-noise ratio at high resistance level, making the high resistance cell difficult to be sensed. As shown in Figure 2(b), the current differences under V_{bias} is difficult to be sensed.

R-metric resistance drift can be modeled by Equation 1 [6], [8]:

$$R(t) = R_0(t/t_0)^{\alpha_R} \tag{1}$$

where R_0 denotes the initial resistance at time t_0, $R(t)$ denotes the cell resistance at time t,and α is the drift

coefficient. Due to process variation, R_0 for a given state is subject to normal distribution of $N(\mu_R, \sigma_R^2)$. Ideally, a desired programmed resistance range is within $10^{\mu_R \pm 2.75\sigma_R}\Omega$, while the boundary of a given state are $10^{\mu_R \pm 3\sigma_R}\Omega$. That is, each state has $10^{\mu_R \pm 0.25\sigma_R}\Omega$ guard band towards its left and right states, respectively. The value of drift coefficient is also subject to a normal distribution of $N(\mu_\alpha, {\sigma_\alpha}^2)$. Table I summarizes the resistance distribution and drift parameters [2] and [26] that we use in the paper.

Table I: R-metric Configuration of four level MLCs when $t_0 = 1s$

Storage Level	Data	$log_{10}R$		α_R	
		μ_R	σ_R	μ_α	σ_α
0	01	3	1/6	0.001	$0.4 \times \mu_\alpha$
1	11	4		0.02	
2	10	5		0.06	
3	00	6		0.10	

Writing MLC cells into narrower resistance subranges enlarges inter-state guardbands such that it takes longer time to drift into errors. This mitigation approach demands precise writing control that takes more write iterations and longer write latency to finish write operations [34]. It is an orthogonal approach to the schemes that are designed and compared in this paper.

2) M-metric: A Drift-Tolerant Voltage based Readout Metric: Alternatively, M-metric [23] was proposed for MLC PCM cells. By applying a bias current to MLC cells, M-metric based sensing, or M-sensing tests the voltage difference of cells storing different data.

Being a weak function of the activation energy, voltage sensing shows significant tolerance to resistance drift — the drift coefficient for M-metric is 6-8x lower than R-metric [23], [1]. Also, the signal range is higher as shown in Figure 2(b). Cells at different resistance states produces easy-to-differentiate voltage values.

Ideally, the readout operation for M-metric is performed by current biasing and voltage sensing. By applying a bias current I_{ref} lower than the threshold current, the voltage difference of different cells are sensed [23]. A major drawback of voltage sensing is its long sensing latency. A naive implementation often needs more than 1000ns to finish read operation [1]. Therefore, recently, [16], [1], [14] optimized the sensing circuit, which improves the read latency in the range of 450ns and makes M-metric sensing practical. In this paper, we use *M-sensing* to refer to this optimized implementation of voltage sensing.

M-metric and R-metric are both cell metrics that can be derived from I-V curve, and M-metric is more resistance drift tolerant. The drift behavior of M-metric was approximated as similar empirical model as R-metric [23]. Equation 2 and Table II shows the M-metric drift model and configuration adopted in this paper. We assume the initial distribution of M-metric is similar to R-metric, $log_{10}M$ is subject to normal distribution, with $\mu_M = \mu_R - 4$, meaning the value of M-metric is 4 orders smaller than R-metric [23]. The drift-coefficient is set to 1/7 of the R-metric as suggested in [1].

$$M(t) = M_0(t/t_0)^{\alpha_M} \quad (2)$$

Table II: M-metric Configuration of four level MLCs when $t_0 = 1s$

Storage Level	Data	$log_{10}M$		α_M	
		μ_M	σ_M	μ_α	σ_α
0	01	-1	1/6	0.0001	$0.4 \times \mu_\alpha$
1	11	0		0.003	
2	10	1		0.010	
3	00	2		0.014	

III. The Design Details

In this section, we first motivate our design by studying the state-of-the-art mitigation schemes. We then elaborate the design details and evaluate the architectural overhead.

A. Motivation

The soft error rate (SER) is an important metric to evaluate the reliability of the memory system in modern computers. The soft errors of DRAM come mainly from particle strikes. Recent studies on DRAM SER shows a wide range from 25∼50 FIT [29] to 25,000∼75,000 FIT [28] (failures in time per billion hours) per Mbit, depending on the settings of the systems. In this paper, we choose a small FIT value, i.e., 25 FIT per Mbit. The smaller value the FIT is, the higher reliability the memory system has. Given that the soft errors of MLC PCM come mainly from drift errors [33], the design goal of this work is to mitigate resistance drift such that, at any given time, the reliability of MLC PCM chip can match that of DRAM chip, i.e., 25FIT for DRAM reliability SER. For a 64B MLC PCM line that contains 512 bits, this SER is translated to line error rate (LER), 3.56E-15 per line-second and 1.28E-11 per line-hour, respectively.

Our design is based on *efficient scrubbing* [2]. More formally,

Definition An (`E, S, W`) *efficient scrubbing* is a scheme that attaches a BCH-`E` code to each MLC PCM line to correct all `E` or fewer errors. It scrubs each memory line in every `S` seconds, and rewrites all cells of the line if detecting `W` or more drifted errors.

In order to meet the reliability of DRAM in an efficient scrubbing design, we need to choose (`E, S, W`) such that:

(i) the probability of a memory line accumulating more than `E` drift errors in the first `S`-second interval after its write, is smaller than LER_{DRAM};

Table III: The line error rate (LER) under different ECC code and scrub interval (**using R-metric sensing**).

Time S (seconds)	Scan in every S seconds and eliminate E or fewer errors								LER_{DRAM} (Target)
	E=0 (No protection)	E=1	E=7	E=8	E=9	E=16	E=17	E=18	
2^2	1.23E-02	9.34E-05	too small	too small	too small	too small	too small	too small	1.42E-14
2^3	7.09E-02	2.56E-03	1.81E-14	1.78E-14	too small	too small	too small	too small	2.84E-14
2^4	1.63E-01	1.43E-02	2.09E-11	4.07E-13	9.55E-15	too small	too small	too small	5.69E-14
2^5	2.81E-01	4.44E-02	2.51E-09	8.98E-11	2.88E-12	too small	too small	too small	1.14E-13
2^6	4.20E-01	1.03E-01	1.06E-07	6.17E-09	3.23E-10	too small	too small	too small	2.28E-13
2^7	5.65E-01	2.03E-01	2.52E-06	2.25E-07	1.80E-08	too small	too small	too small	4.55E-13
2^8	7.02E-01	3.43E-01	3.73E-05	4.84E-06	5.63E-07	9.10E-15	too small	too small	9.10E-13
2^9	8.18E-01	5.11E-01	3.78E-04	6.86E-05	1.12E-05	3.33E-12	2.92E-13	1.06E-14	1.82E-12
640	8.50E-01	5.65E-01	7.21E-04	1.44E-04	2.60E-05	1.55E-11	1.51E-12	1.32E-13	2.28E-12
2^{10}	9.03E-01	6.79E-01	2.68E-03	6.59E-04	1.46E-04	3.80E-10	4.61E-11	4.42E-12	3.64E-12

Table IV: The line error rate (LER) under different ECC code and scrub interval (**using M-metric sensing**)

Time S (seconds)	Scan in every S seconds and eliminate E or fewer errors								LER_{DRAM} (Target)
	E=0 (No protection)	E=1	E=2	E=3	E=4	E=5	E=6	E>=7	
$2^1 - 2^6$	too small	too small	too small	too small	too small	too small	too small	too small	2.28E-13
2^7	6.40E-06	2.04E-11	too small	too small	too small	too small	too small	too small	4.55E-13
2^8	3.84E-05	7.34E-10	3.33E-15	too small	too small	too small	too small	too small	9.10E-13
2^9	2.69E-04	3.60E-08	3.18E-12	too small	too small	too small	too small	too small	1.82E-12
2^{10}	9.85E-04	4.83E-07	1.58E-10	4.54E-14	7.11E-15	too small	too small	too small	3.64E-12
2^{11}	2.42E-03	2.91E-06	2.33E-09	1.38E-12	7.99E-15	too small	too small	too small	7.28E-12
2^{12}	4.78E-03	1.14E-05	1.80E-08	2.13E-11	2.99E-14	too small	too small	too small	1.46E-11
2^{13}	8.14E-03	3.31E-05	8.94E-08	1.80E-10	3.01E-13	too small	too small	too small	2.91E-11
2^{14}	1.26E-02	7.91E-05	3.31E-07	1.03E-09	2.58E-12	6.88E-15	1.67E-15	too small	5.83E-11

Table V: The LER when choosing different W=1

R(...): R-sensing M(...): M-sensing	Probability of (ii)		Probability of (iii)	
	W=1	LER_{DRAM}	W=1	LER_{DRAM}
R(BCH=8,S=8)	3.59E-13	5.69E-14	9.1E-12	8.54E-14
R(BCH=10,S=8)	4.83E-14	5.69E-14	1.7E-14	8.54E-14
M(BCH=8,S=640)	too small	4.56E-12	too small	6.84E-12

(ii) the probability of a memory line accumulating less than W errors in the first S-second interval while accumulating more than (E-W) errors in the second S-second interval is smaller than LER_{DRAM};

(iii) the probability of a memory line accumulating less than W errors in the first two S-second intervals while accumulating more than E−W errors in the following S-second interval is smaller than LER_{DRAM};

We next check these probabilities to set up the parameter values for the baseline designs. Table III and Table IV summarize the probabilities of (i) when adopting R-sensing and M-sensing, respectively. From Table III, a R-sensing scheme that adopts (BCH=8,S=8) can meet LER_{DRAM}. In this scheme, each memory line is attached with a BCH-8 code and is scrubbed every 8s. This matches the observation in [26], [2] that adopts BCH-8 code. M-sensing can meet LER_{DRAM} using (BCH=8,S=640). While the scrubbing interval for M-sensing can be relaxed to 2^{14} (16,384) seconds, we choose S=640 for reasons that we will elaborate in the next section.

We then check the probabilities of (ii) and (iii) and summarize the results for three selected E and S combinations in Table V. Our study shows that W needs to be small for R-sensing — a bigger W requires stronger BCH code than those listed. From Table V, we find that (BCH=8, S=8) fails to match LER_{DRAM} even using W=1. The implication is, a scrubbing scheme that adopts (BCH=8, S=8) needs to rewrite every line at scrubbing time no matter if the line has a drift error. A scheme that adopts (BCH=10, S=8) can relax it to skip rewriting if no error is found. A M-sensing scheme is much safer, e.g., using (BCH=8,S=640,W=1) is sufficient to meet LER_{DRAM}.

In this paper, we choose (BCH=8, S=8, W=1) for R-metric based scrubbing and (BCH=8, S=640,W=1) for M-metric based scrubbing. From above analysis, a reliable R-sensing scheme needs either using W=0 (i.e., rewriting at scrubbing time) or using (BCH=10,W=1). Both of which tend to introduce larger overhead. We choose (BCH=8, S=8, W=1) only for comparison purpose. For M-metric based scrubbing, it is possible to relax W or S to further reduce overhead. Given the large interval, we observe that the overhead is already low.

Comparing different mitigation schemes. Figure 3 studies the state-of-the-art drift mitigation schemes. `Scrubbing` indicates the scheme that adopts R-sensing [2] with (BCH=8,S=8,W=1). `M-metric` indicates the scheme that adopts M-sensing with (BCH=8,S=640,W=1). `TLC` indicates the scheme that adopts the TLC scheme [26].

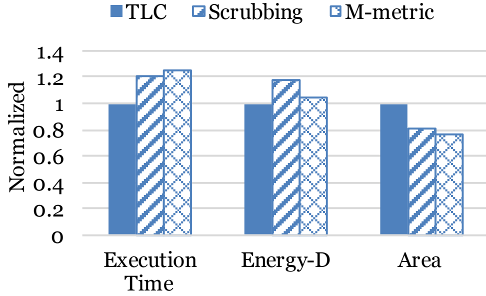


Figure 3: Comparing existing drift mitigation schemes.

As shown in the figure, `Scrubbing` and `M-metric` introduce large performance degradation — `scrubbing` wastes memory bandwidth on scrubbing and reduces memory availability; `M-metric` slows down each memory access. `TLC`, while showing no performance degradation, show large density penalty. In summary, mitigating resistance drift in MLC PCM chips remains a challenging research topic.

In this paper, our design goal is to devise a drift resilient solution that matches the reliability of MLC PCM with that of DRAM, and achieve high performance, low energy consumption, good memory density, and good chip lifetime simultaneously, as shown in Table VI.

Table VI: The design goal of our proposed ReadDuo scheme

	Performance	Energy	Density	Endurance
Scrubbing	-	-	+	-
TLC	+	+	-	+
M-metric	-	-	+	+
ReadDuo	**+**	**+**	**+**	**+**

B. ReadDuo-Hybrid: a Simple Hybrid Integration of R-Sensing and M-Sensing

Given that (i) R-sensing is faster than M-sensing but requires a short 8s scrubbing interval, and (ii) M-sensing can meet $\text{LER}_{\texttt{DRAM}}$ with a large 640s scrubbing interval, a simple integration of both sensing approaches is to conduct R-sensing first and, if R-sensing fails due to resistance drift, conducts M-sensing. This is referred to as *simple hybrid sensing*, or *ReadDuo-Hybrid*. Intuitively, if most read operations finish with R-sensing, ReadDuo-Hybrid can achieve performance close to that in R-metric only sensing, and, if there are drift errors, M-sensing with (BCH=8,S=640,W=1) helps to provide reliability guarantee.

Unfortunately, R-sensing is less reliable. As an example, the data returned from R-sensing may contain more than 20 errors while, if sensing the *same* memory line at the *same* time with M-metric, the returned data has only five errors. In ReadDuo-Hybrid, we need to ensure the undetected errors, i.e., those that cannot be detected by BCH-8, during R-sensing can meet the reliability target $\text{LER}_{\texttt{DRAM}}$.

We next elaborate the design details of ReadDuo-Hybrid. In particular, we need to enhance W=1 to W=0 in order to meet DRAM reliability, i.e., ReadDuo-Hybrid is a hybrid sensing that adopts (BCH=8,S=640,W=0) for M-metric based scrubbing — each line is refreshed every 640s.

Decoupling error detection and error correction. BCH code is an ECC code with Hamming distance `d`, which can detect up to `d-1` errors and correct up to `d/2-1` errors [22]. That is, the BCH-8 code adopted in hybrid sensing can correct up to 8 errors and detect up to $8 \times 2 + 1 = 17$ errors. However, when adopting BCH-8 in memory systems, we often do not care if 9 or more errors may be detected as the line shall be left in erroneous state anyway. In this paper, we decouple error detection and error correction, and take advantage of the full error detection capability of BCH-8 code.

In particular, to service a read request, ReadDuo-Hybrid first conducts R-sensing and determines the number of drift errors under R-metric.

- If there are 8 or fewer errors, ReadDuo-Hybrid corrects the errors using the BCH-8 attached to the memory line and returns the corrected data to the processor.
- If there are 9 to 17 errors, ReadDuo-Hybrid re-issues the memory request to conduct M-metric sensing. The second try returns the correct data to the processor.
- If there are more than 17 errors, ReadDuo-Hybrid cannot differentiate erroneous data from correct data, and thus return the sensed data to the processor with no correction.

We then check Table III and find that the probability of a memory line having 17 or more errors is lower than $\text{LER}_{\texttt{dram}}$ for up to 640s. Given that our M-metric scrubbing interval is also 640s, we next discuss how to ensure the reliability of ReadDuo-Hybrid.

Enhancing W=1 based scrubbing to W=0. When a hybrid sensing uses setting (BCH=8,S=640s,W=1), it shall perform M-metric based scrubbing on all lines in every 640s. Since M-metric is drift-resistant, scrubbing will skip re-writing most memory lines as there are no errors. Given that R-sensing can reliably sense a memory line only within 640s after its last write, such a skip jeopardizes R-sensing as it may be issued beyond 640s from its last write.

In this section, ReadDuo-Hybrid addresses the issue with a simple enhancement that adopts W=0 instead of W=1. We leave more advanced designs to the following sections. With setting (BCH=8,S=640s,W=0), ReadDuo-Hybrid re-writes each memory line at its scrub time, no matter if an error was found. In this way, R-sensing is always within 640s after a line write, which ensures the reliability in ReadDuo-Hybrid.

Read modes. Figure 4 compares ReadDuo-Hybrid to the two schemes that adopt R-metric sensing and and M-metric sensing individually. To simplify discussion, we differentiate the following three types of read operations.

- *R-read.* The R-metric based sensing finishes in 150ns. Scrubbing the memory using (BCH=8,S=8,W=1) may roughly ensure all read operations being safely serviced with R-read.
- *M-read.* The M-metric based sensing finishes in 450ns. Scrubbing the memory using (BCH=8,S=640,W=1) can ensure all read operations being safely serviced with M-read.
- *R-M-read.* ReadDuo-Hybrid, when encountering 9 to 17 drift errors in R-sensing step, needs to conduct M-sensing and thus finishes in 600ns (=150ns+450ns). This is referred to as *R-M-read*.

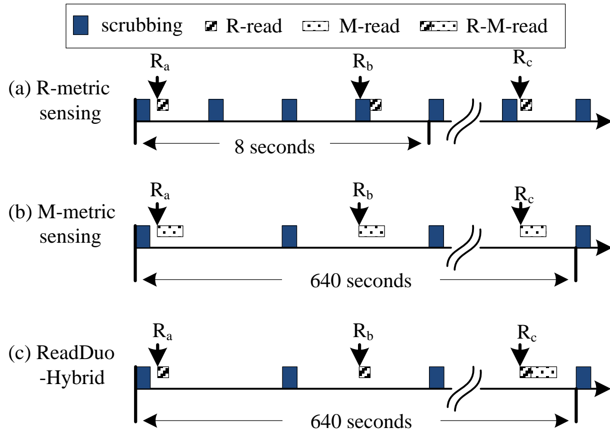


Figure 4: Hybrid sensing speeds up read operations.

As shown in Figure 4(a) that adopts R-metric only sensing, read requests are all serviced by fast R-read operations. An individual request, e.g., R_b, could be delayed by frequent scrubbing operations. In Figure 4(b) that adopts M-metric only sensing, scrubbing is much less frequent and the bank is also less busy. However, read requests are serviced by slow M-read operations. In Figure 4(c) that adopts hybrid sensing, a read request can be serviced by either R-read or R-M-read. Due to low error rate under BCH-8 protection, most read requests are serviced by R-read.

C. ReadDuo-LWT: Tracking Last Write in Hybrid Sensing

ReadDuo-Hybrid chooses W=0 in M-metric scrubbing to ensure read reliability. Comparing to a scheme that chooses W=1, ReadDuo-Hybrid introduces large overhead as it re-writes each memory lines in every 640s; the W=1 scheme re-writes a line only if there are errors. Given the low error rate in M-sensing, the W=1 scheme generates negligible re-write operations in each 640s scrubbing interval.

To reduce the number of write operations in ReadDuo-Hybrid, we propose a different optimization over the W=1 baseline. The scheme, referred to as *ReadDuo-LWT*, tracks the last write to each memory line and switches to conduct M-sensing if a read occurs beyond 640s from its last write.

Tracking the last write. To track the last write, a ReadDuo-LWT-k scheme partitions one scrubbing interval into k sub-intervals and attaches two flags (with k bits and $\log_2$k bits, respectively) to each memory line.

- We label the sub-intervals from 0 to k-1, as shown in Figure 5. One memory line keeps a k-bit vector-flag to indicate if the memory line was written in the preceding sub-intervals — bit-x=1 indicates there was a write in the current or closest sub-interval labeled with x. It also saves a $\log_2$k-bit index-flag ind to indicate the sub-interval in which the last write occurs.
  In the example shown in Figure 5, write W_1 sets bit 2 of the vector-flag and modifies the index-flag to 2 because it locates in sub-interval#2.
- When scrubbing a memory line, we first clear the vector-flag bits before the last write, i.e., all bits in the range of [0, ind-1]. If ind=0, then all bits in the vector-flag are cleared. We then set bit 0 of the vector flag to 1 if the scrub operation re-writes the line, and to 0 otherwise.
  In the example shown in Figure 5, scrub1 clears bit 1 and bit 0 in the vector-flag while scrub3 clear all bits.
- When there is a write operation, we identify the sub-interval s that the write belongs to, and the current index flag ind. If s >1, we clear the vector flag bits that correspond to sub-interval range [ind+1,s). We set the vector flag bit for sub-interval s.

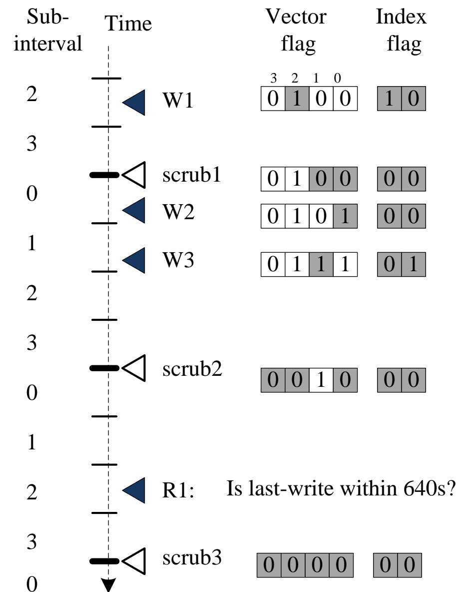


Figure 5: The partition of scrub interval for one memory line. (The three scrub operations are all for this memory line; none actually re-writes the line. The shaded flag bits are changed bits at each step).

Intuitively, the vector-flag is a sliding window that tracks and retires writes in sub-intervals. The index-flag records sub-interval of the last write or the starting of a new scrubbing interval. The location of the last write is determined by both of them, as discussed in the following.

Enhanced readout control. To service a read operation, ReadDuo-LWT reads data/ECC bits using R-sensing and simultaneously reads the flag bits. The flag bits are stored as single-level cell (SLC) and thus do not suffer from resistance drift [2]. ReadDuo-LWT also identifies the sub-interval s in which the read belongs to.

There are three cases — (i) If both vector-flag and index-flag are non-zero, indicating there is a write within 640s, ReadDuo-LWT can continue R-sensing as it is reliable. (ii) If the vector-flag is zero, indicating there is no write in the past 640s, ReadDuo-LWT terminates R-sensing and switches to M-sensing. (iii) Otherwise, i.e., the vector-flag is non-zero while the index-flag is zero. ReadDuo-LWT first discards the vector-flag bits in [1,s] as these bits indicate writes beyond 640s. If the vector flag is still non-zero, ReadDuo-LWT continues R-sensing. Otherwise, ReadDuo-LWT switches to M-sensing.

In Figure 5, read R1 locates in sub-interval 2. The vector-flag, while being non-zero initially, becomes zero after discarding bit 1 and bit 2 (i.e., in range [1, s]). We therefore switch to M-sensing to reliably read the memory line.

R-M-Read conversion. By tracking last writes to memory lines, ReadDuo-LWT can choose (BCH=8,S=640,W=1) for M-metric scrubbing without impacting reliability. This greatly reduces memory re-write operations. However, when many read operations are beyond 640s of their last write operations, the memory performance can become worse than the scheme that only adopts M-sensing — R-M-read is slower than M-read because the latter does not need to test flag bits. This could become a big concern if, as an example, an in-memory database application first creates the database and then performs read-intensive query operations. In this case, the read operations need to access data that were written to memory a long time ago.

To mitigate this concern, we propose to conduct redundant write operation after R-M-read. That is, After servicing a read request with R-M-read, ReadDuo-LWT re-writes the same data back to the MLC PCM. ReadDuo-LWT then tracks this write and enables fast R-sensing in the next 640s interval.

However, blindly converting all R-M-read operations could greatly degrade chip lifetime due to additional writes introduced to PCM chips. For this reason, we dynamically monitor $P\%$ — the percentage of reads falling to un-tracked memory lines. We convert $T\%$ of R-M-read and adjust T between [0,100] at step 10. We increase T if an increment gives 2 times percentage increase on P and decrease, and decrease T if P is greater than 85%. Otherwise, we keep the same T.

D. ReadDuo-Select: Selectively Rewriting MLC Cells

Reducing the number of cell writes is an effective approach to address the well-known write endurance problem in PCM. For example, based on the observation that a write operation typically changes around 20% data bits to its memory line, *differential write* only writes modified bits to PCM cells, which greatly improve PCM chip lifetime [35]. Unfortunately, due to resistance drift, the MLC write operations, including both writes from processors and writes from scrubbing, need *full-line* write i.e., writing all cells in the line.

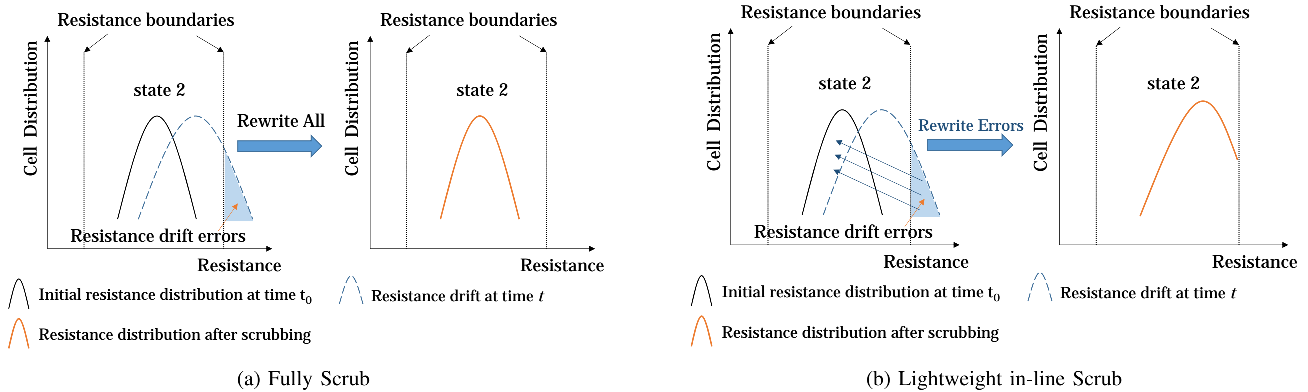
Figure 6 elaborates the details. To simplify the discussion, we assume all cells were programmed to state '01', with their resistances forming a normal distribution within the resistance range of state '01'. Figure 6a shows that, due to resistance drift, a small number of cells drift across the state boundary between state '01' and state '00', leading to drift errors. If we only write the modified cell or the drifted cells, as shown in Figure 6b, the resistances of the cells in this MLC line do not follow normal distribution. A large number of cells are now close to the boundary such that, in the next scrubbing interval, this line may accumulate more drift errors than what its ECC can handle. Therefore, writing a memory line needs to write all cells such that their resistances still follow normal distribution.

Full-line write not only wastes write energy but also degrade MLC chip lifetime. We therefore propose ReadDuo-Select, a selective differential write (SDW) design that safely reduces cell writes in MLC PCM. A ReadDuo-Select-(k:s) scheme enhances the last-write tracking policy used in ReadDuo-LWT-k as follows.

- Intuitively, a ReadDuo-Select-(k:s) scheme performs only one full-line write in s consecutive sub-intervals. The index-flag indicates the sub-interval in which the last full-line write was performed.
- When performing a write operation to a memory line, we determine the sub-interval that the write operation belongs to, and the last full-line write (saved in the index-flag). If they are within s sub-intervals, we only write modified cells. Otherwise, a full-line write is performed.

  When we convert a read operation to a write operation in ReadDuo-LWT, we need to write all cells as this write is the only write in the past s sub-intervals.
- When performing a differential write in MLC PCM, ReadDuo-Select does not update the index-flag. As such, when determining if a read can perform R-sensing, ReadDuo-Select conservatively measures the distance to the last full-line write, which ensure its read reliability.

When s=1, SDW performs a full-line write only for the first write operation in each sub-interval and converts following writes from the same sub-interval to differential



(a) Fully Scrub

(b) Lightweight in-line Scrub

Figure 6: Reliable scrubbing needs to rewrite all cells.

writes. When s >1, SDW converts more full-line writes to differential writes, but relaxes tracking the last write operation, which may slow down some ready operations. In the experimental sections, we will study different s values and their impacts on performance and chip lifetime.

E. Architectural Enhancement

Figure 7 presents an overview of the proposed ReadDuo architecture. One PCM rank consists of eight data chips, one ECC chip, and a bridge chip. The ECC chip is to mitigate soft errors triggered by resistance drift, the same as that in [2]. To defend hard errors, we may increase the error correction capability of the current ECC chip, or even add a new ECC chip if necessary. The design of hard error mitigation schemes is orthogonal to our work. The bridge chip is responsible for fine-grained PCM timing control and device specific management, which helps to mitigate the non-determinism in MLC PCM [9].

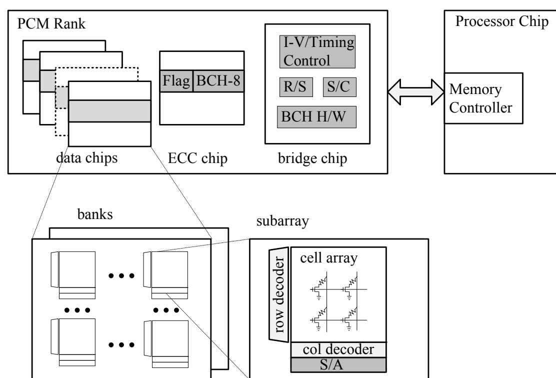
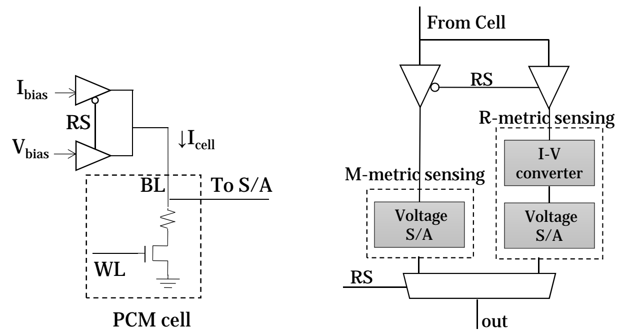


Figure 7: An overview of ReadDuo architecture (dark boxes show enhanced components).

In figure 7, the dark boxes mark the ReadDuo enhanced architecture components that include (i) readout selection (R/S) to switch between R-sensing and M-sensing; (ii) scrubbing control (S/C) to scrub memory lines periodically; (iii) BCH hardware to encode and decode the BCH-8 code attached to each memory line; (iv) the enhanced I/V and timing control; (v) a hybrid sense amplifier (S/A) that supports both current and voltage sensing.



(a) Applying read voltage/current

(b) Sensed by current/voltage S/A

Figure 8: The details of hybrid sensing.

Within each subarray, ReadDuo integrates both the traditional R-metric based current-mode sensing logic and the new M-metric based voltage-mode sensing logic. The details are shown in Figure 8. Current-mode sensing [4], [5] applies a fixed bias voltage V_{bias} to the cell and compares the current flow through the bitline I_{cell} to the reference current generated by reference cells. It requires large die area because current signals are converted to voltage signals and then sensed. In contrast, M-metric reading only requires voltage sensing sense amplifier but not the I-V converter. By applying a bias current I_{bias} to the cell, we sense the voltage on the bitline and then compare it to predefined reference voltage, and finally output the readout value. Other peripheral circuits are shared, such as row decoder, column decoder, precharger etc.

We revised NVSim [5] to model both types of sense amplifiers and estimate the area overhead. Each 2GB memory bank consists of 32 mats while each mat contains 16 subarrays. At the subarray level, the area occupancy of control logic and data array are shown in Table VII. The overall area increment is 0.27%.

Table VII: Subarray level area analysis

Component	Baseline[mm^2] (current sensing)	R+M-metric[mm^2] (hybrid sensing)
Cell Array	2.466	2.466
Row Decoder	0.254	0.254
Precharger	0.026	0.026
I-V converter	0.208	0.208
Bitline Mux & Dec	0.015	0.015
S/A Mux & Dec	0.002	0.002
S/A	0.003	0.006
Mode Control	X	0.005
Total	2.974	2.982

Table VIII: Baseline Configuration

CPU	4-core single issue in-order CMP, 4GHz
L1	private, I/D separate, 32KB per core, 64B line
L2	private, 256KB per core, 4-way LRU, 64B line write back
L3	DRAM cache, private, 8MB per core, 8-way LRU 64B line, 50ns (200-cycle) hit
Main Memory	16GB, 8 banks, 32-entry write queue per bank R-Read: 150ns [3] ,M-Read: 450ns [1] Write: 1000ns

ReadDuo maintains a scrub register to indicate the next memory line to be scrubbed. It increments after each scrub, i.e., increments every (S / number_of_memory_lines) seconds. In addition, each memory line keeps (k+$\log_2 k$) bits to track its last write. These bits are stored as SLC in the ECC chip, which do not suffer from resistance drift [2]. Accessing the flag bits is off the critical path and incurs negligible performance overhead.

IV. Experimental Methodology

To evaluate the effectiveness of ReadDuo, we simulated a system using 4-core in-order CPU and MLC PCM based main memory. The baseline architectural configuration follows [26]. We generated memory accesses trace through Pintool [13], and fed it to our in-house memory system simulator, which models the entire memory hierarchy, the memory controller and PCM based main memory. We also considered timing constraints including cache-memory related bus contention, memory bank conflicts and DDR scheduling constraints. Write cancellation [18] was also implemented so that read operations are given highest priority. For the memory systems, the read latency is 150ns [3] for fast R-metric sensing, and 450ns [1] for slow M-metric sensing. MLC write latency is set as 1000ns [2] for the iterative based write strategy. Scrubbing needs to read a PCM line first, scan for drift errors, and re-write a line if errors were found. The detailed system configuration is in Table VIII. Table IX lists the energy consumed to read and write MLC cells [31].

In our experiment, we simulated 14 workloads from SPEC2006 benchmarks. We listed their RPKI/WPKI (read/write operations per thousand instructions) in Table X.

Table IX: Energy Model of PCM per bit access

R-Read	M-Read	Write 01	Write 11	Write 10	Write 00
$10pJ$	30 pJ	$50pJ$	$100pJ$	$400pJ$	$1600pJ$

Table X: Simulated Workloads

	RPKI	WPKI		RPKI	WPKI
astar	14.12	11.35	bwaves	17.81	9.14
bzip2	2.83	2.83	gemsfdtd	9.67	9.08
gromacs	0.49	0.49	lbm	17.08	11.75
leslie3d	5.21	5.14	libquantum	13.53	7.55
mcf	23.23	21.15	milc	21.22	13.08
sjeng	0.39	0.39	sphinx	3.97	3.36
wrf	0.90	0.90	zeusmp	4.23	4.08

V. Results

We implemented and compared the following schemes in the section.

- `Ideal`. This is the setting that assumes no resistance drift in MLC PCM. A MLC line can be readout using R-metric sensing in 150ns.
- `Scrubbing`. This scheme adopts *efficient scrubbing* with R-metric sensing. It uses (BCH=8, S=8s, W=1). In practice, it needs to be enhanced with W=0 to meet LER_{DRAM}.
- `M-metric`. This scheme adopts M-metric sensing only. It uses (BCH=8, S=640s, W=1).
- `Hybrid`. This scheme implements ReadDuo-Hybrid. It uses (BCH=8, S=640s, W=0).
- `LWT-`k. This scheme is built on top of `Hybrid` and implements ReadDuo-Hybrid-k, which partitions 640s to k sub-intervals, and convert R-M-Read to write for untracked memory lines.
- `Select-(`k`:`s`)`. This scheme, in addition to `LWT-`k, implements ReadDuo-Select-(k:s).

A. Performance Comparison

Figure 9 reports the normalized execution time using different schemes. We normalized the results to `Ideal`. `Scrubbing` and `M-metric` introduce 21% and 25% performance degradation.

The overhead of `Scrubbing` comes mainly from busy memory banks. Note here we use W=1 setting, which cannot meet our target reliability requirement. If W=0 setting is used, the memory suffers from more scrubbing operations, and the execution time for that case is much longer, about 2-3x over ideal.

The overhead of `M-metric` comes mainly from slow memory read operations. Since read operation is on the critical path, the execution time is prolonged significantly.

`Hybrid` reduces the scrubbing frequency and also enables that most read operations can be fulfilled by fast R-read. The 5.8% extra execution time compared with `Ideal`

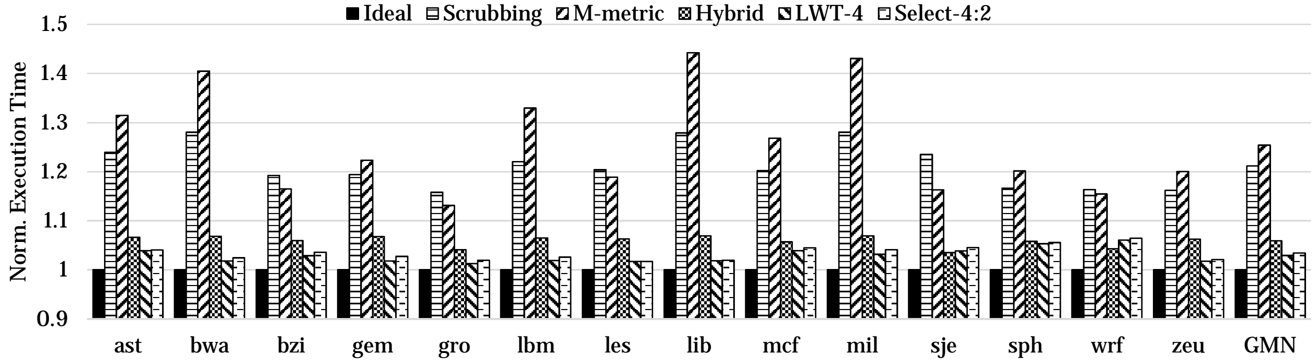


Figure 9: Comparing execution time.

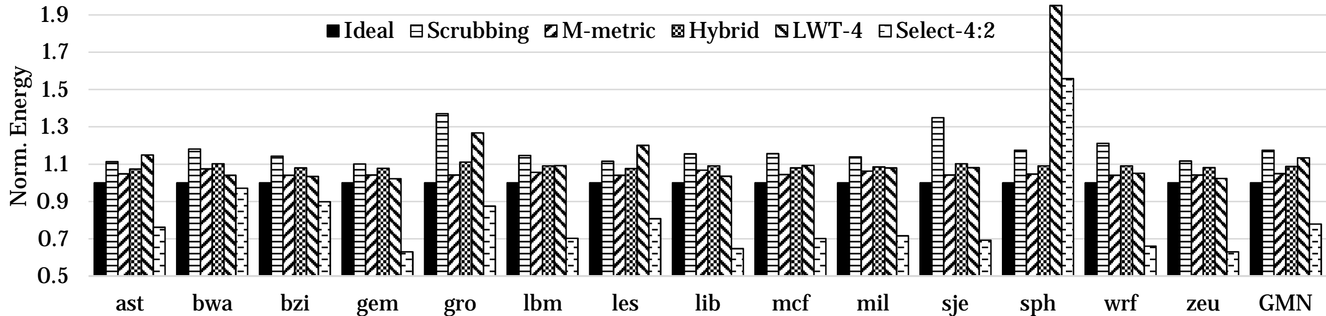


Figure 10: Comparing the energy consumption.

comes mainly from the scrubbing overhead with W=0 setting. The energy overhead comes from W=0 setting is even more significant.

`LWT-4` implements the last write tracking on a granularity of 1/4 of 640s, i.e, 160s sub-interval tracking. With W=1 scrubbing setting, the scrubbing overhead is reduced. By implementing the R-M-read conversion for untracked memory lines, we also lower the probability of long latency R-M-read. We further analyze the impact of each enhancement in next section. The `LWT-4` design can lower the extra execution time to 2.9% over ideal.

At last, the energy optimization scheme `Select-4:2` will slightly increase the performance overhead, because the probability of R-M-read will slightly increase. The extra execution time is 3.4% over ideal.

B. Dynamic Energy Comparison

Figure 10 reports the dynamic energy consumption under different schemes. We normalized the results to `Ideal`. Due to frequent scrubbing, `Scrubbing` consumes 17% more energy than `Ideal` with W=1 setting. `M-metric` consume 5% extra dynamic energy due to long read latency. The scrubbing energy consumption is small because the frequency to issue a scrub operation is lower. `Hybrid` adopts W=0 scrubbing, meaning that more scrub operations has to be issued. Therefore, the extra energy overhead is 8.7%.

Our proposed `LWT-4` reduces energy consumption for some benchmarks, like bwaves, bzip2, etc. The reduction comes from two parts, one is because the read latency is now shortened compared with `M-metric`. Also, by using W=1 scrubbing mode, our scrubbing energy is also saved compared with `Hybrid`. However, it is noticeable that for some benchmarks, such as sphinx, the energy consumption is increased significantly. This is because for such benchmarks, we convert many R-M read to write operations, thus the energy consumption is higher. Overall, `LWT-4` has an energy overhead of 1.33%.

The `Select-4:2` reduces the write energy significantly, which contributes the major part of the overall dynamic energy. Since we only need to modify changed bits, the energy is only 77.8% of the `Ideal`, which writes all line with full writes.

C. Energy Delay Area Product

We next examined different schemes by taking subarray area into consideration. We start with comparison of the effective cell array sizes when storing same amount of information. Take 64B data as example, the TLC design with (72,64) SECDED code requires 192 tri-level cells to store data. Scrubbing with BCH-8 and parity check per 32bits, resulting to use 155 cells to store data. Our proposed schemes, also requires BCH-8 attached per line. With LWT-4, another 153 cells per line is required, thus storing 64B

requires cells. We normalize all the required cells to store a line to TLC and show the result in Figure 11.

To evaluate the schemes on performance, energy consumption, and area, we used EDAP (Energy Delay Area product) metric and reported the comparison results in Figure 11. In this figure, lower bars mean better results. `Product-D` and `Product-S` are the results when considering dynamic energy and system energy, respectively. When considering dynamic energy only, on average, `LWT-4` and `Select-4:2` achieves 7.5% and 37% over `TLC`.

When considering system energy, they achieves 11% and 23% over `TLC` design.

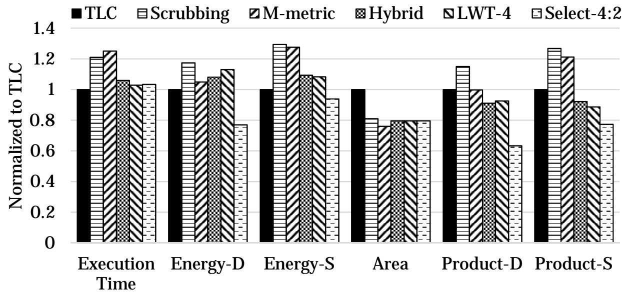


Figure 11: Comparing EDAP (Energy Delay Area Product).

D. Sensitivity Study

In this section, we varied the choices of parameters and evaluated their impacts on the system.

1) Impact of Subinterval Number(k): The value of k determines how many subintervals we have in one 640s scrubbing interval. With bigger k, we are able to track longer time. Longer tracking time will give some benchmarks higher opportunity to enable fast R-read. At the same time, we need to add more flag bits to enable finer granularity tracking. We report several benchmarks that shows significantly better performance with more subintervals. Figure 12 shows that the performance improvement from $k=2$ to $k=4$ is 0.7% for all benchmarks, and 2.3% for memory intensive benchmark *mcf*.

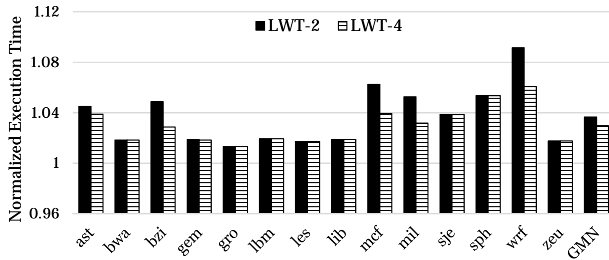


Figure 12: Impact of Subinterval Number(k)

2) Impact of Select Rewrite Interval Number(s): As stated in Section III-D, the choice of s determines how frequently we can convert a full write to a selective write. We compare Select-4:1 and Select-4:2 two settings and compare the energy reduction results. Figure 13 shows that the energy saving for $s=2$ over $s=1$ is 1.2%.

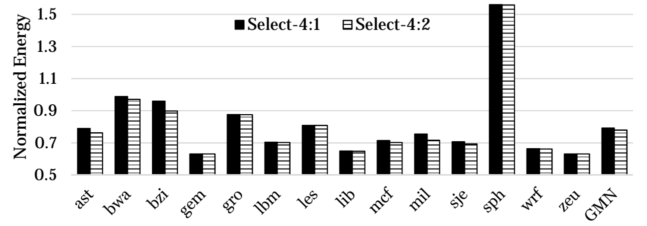


Figure 13: Impact of Select Rewrite Interval Number(s)

3) R-M-read conversion in LWT-k: Figure 14 shows the benefit of R-M read conversion enhancement in LWT-4. For several benchmarks, for example, sphinx, enabling R-M Read conversion will bring much better performance. The performance improvement for sphinx by enabling R-M Read conversion is 22%. Meanwhile, as we analyze in Section V-B, the energy consumption is also increased because of the conversion. Overall, R-M-read conversion with LWT-4 will gain 2.9% improvement on performance.

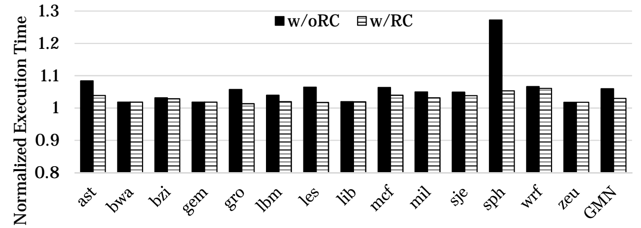


Figure 14: Impact of Read Conversion

E. Lifetime Impact

For PCM, the memory lifetime is determined by numbers of write operations. We examine different schemes impact on memory lifetime and show the result in Figure 15. `Scrubbing` will shorten 12.4% lifetime. `M-metric` with W=1 scrubbing will have negligible lifetime impact on memory. `Hybrid`, `LWT-4` will reduce lifetime by 6% and 10%. By enabling the `Select-4:2`, the memory lifetime is increased by 42%.

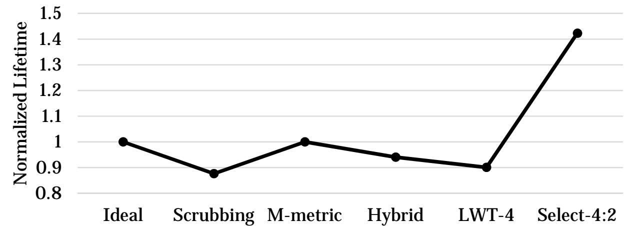


Figure 15: Impact on PCM Lifetime

VI. Related Work

Phase Change memory is one of promising non-volatile memory technologies for future memory systems. In addition to resistance drift, many issues have been studied in the literature. ECP [27], PAYG [17], Free-p [33], Safer [25] focus on hard errors in PCM. Write disturbance was addressed in SD-PCM [30] and DIN [10]. Wear-leveling techniques were proposed in Security-refresh [24] and Start-gap [19]. MLC PCM also suffers from long write latency and leading performance degradation. Write cancellation [18], write truncation[11] addressed the long write latency in MLC PCM.

VII. Conclusions

In this paper, we propose ReadDuo, a resistance drift resilient readout solution for MLC PCM system. ReadDuo combines fast R-metric sensing and drift resilient M-metric sensing. It converts the reliability design to a performance optimization problem. By selectively scrubbing MLC lines for drift errors and selectively rewriting drifted cells only, ReadDuo achieves high performance, low energy consumption, and good storage density simultaneously.

Acknowledgment

We thank all anonymous reviewers for their valuable comments and suggestions. This research is supported in part by NSF #1535755, NSF #1422331, and NSF #1012070.

References

[1] A. Athmanathan, M. Stanisavljevic, J. Cheon, *et al.*, "A 6-bit drift-resilient readout scheme for multi-level phase-change memory," in *IEEE A-SSCC*, 2014.

[2] M. Awasthi, M. Shevgoor, K. Sudan, *et al.*, "Efficient scrub mechanisms for error-prone emerging memories," in *HPCA*, pp. 1–12, 2012.

[3] Y. Choi, I. Song, M.-H. Park, *et al.*, "A 20nm 1.8 v 8gb pram with 40mb/s program bandwidth," in *ISSCC*, pp. 46–48, 2012.

[4] X. Dong, N. P. Jouppi, and Y. Xie, "Pcramsim: System-level performance, energy, and area modeling for phase-change ram," in *ICCAD*, pp. 269–275, 2009.

[5] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE TCAD*, 31(7):994–1007, 2012.

[6] D. Ielmini, A. L. Lacaita, and D. Mantegazza, "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories," *IEEE Trans. on Electron Devices*, 54(2):308–315, 2007.

[7] D. Ielmini, D. Sharma, S. Lavizzari, *et al.*, "Reliability impact of chalcogenide-structure relaxation in phase-change memory (pcm) cellspart i: Experimental study," *IEEE Trans. on Electron Devices*, 56(5):1070–1077, 2009.

[8] D. Ielmini, S. Lavizzari, D. Sharma, *et al.*, "Physical interpretation, modeling and impact on phase change memory (pcm) reliability of resistance drift due to chalcogenide structural relaxation," in *IEDM*, pp. 939–942, 2007.

[9] L. Jiang, Y. Zhang, B. R. Childers, and J. Yang, "FPB: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory," in *MICRO*, pp. 1–12, 2012.

[10] L. Jiang, Y. Zhang, and J. Yang, "Mitigating write disturbance in super-dense phase change memories," in *DSN*, pp. 216–227, 2014.

[11] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in mlc phase change memory," in *HPCA*, pp. 1–10, 2012.

[12] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *ISCA*, pp. 2–13, 2009.

[13] C.-K. Luk, R. Cohn, R. Muth, *et al.*, "Pin: building customized program analysis tools with dynamic instrumentation," in *PLDI*, pp. 190–200, 2005.

[14] N. Papandreou, H. Pozidis, T. Mittelholzer, *et al.*, "Drift-tolerant multilevel phase-change memory," in *IEEE International Memory Workshop (IMW)*, 2011.

[15] A. Pirovano, A. L. Lacaita, F. Pellizzer, *et al.*, "Low-field amorphous state resistance and threshold voltage drift in chalcogenide materials," *Electron Devices, IEEE Transactions on*, 51(5):714–719, 2004.

[16] H. Pozidis, N. Papandreou, A. Sebastian, *et al.*, "Enabling technologies for multilevel phase-change memory," in *European Phase Change & Ovonics Symposium*, 2011.

[17] M. K. Qureshi, "Pay-as-you-go: low-overhead hard-error correction for phase change memories," in *MICRO*, pp. 318–328, 2011.

[18] M. K. Qureshi, M. M. Franceschini, L. Lastras-Monta, *et al.*, "Improving read performance of phase change memories via write cancellation and write pausing," in *HPCA*, pp. 1–11, 2010.

[19] M. K. Qureshi, J. Karidis, M. Franceschini, *et al.*, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *MICRO*, pp. 14–23, 2009.

[20] S. Raoux, G. W. Burr, M. J. Breitwisch, *et al.*, "Phase-change random access memory: A scalable technology," *IBM Journal of Research and Development*, 52(4):465–479, 2008.

[21] A. Redaelli, A. Pirovano, F. Pellizzer, *et al.*, "Electronic switching effect and phase-change transition in chalcogenide materials," *IEEE Electron Device Letters*, 25(10):684–686, 2004.

[22] K. H. Rosen, *Discrete Mathematics and Its Applications,Chapter 5*. McGraw-Hill Higher Education, 5th ed., 2002.

[23] A. Sebastian, N. Papandreou, A. Pantazi, *et al.*, "Non-resistance-based cell-state metric for phase-change memory," *Journal of Applied Physics*, 110(8), 2011.

[24] N. H. Seong, D. H. Woo, and H.-H. S. Lee, "Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping," in *ISCA*, pp. 383–394, 2010.

[25] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "Safer: Stuck-at-fault error recovery for memories," in *MICRO*, pp. 115–124, 2010.

[26] N. H. Seong, S. Yeo, and H.-H. S. Lee, "Tri-level-cell phase change memory: Toward an efficient and reliable memory system," in *ISCA*, 2013.

[27] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ecp, not ecc, for hard failures in resistive memories," in *ISCA*, pp. 141–152, 2010.

[28] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: a large-scale field study," in *ACM SIGMETRICS*, pp. 193–204, 2009.

[29] V. Sridharan, J. Stearley, N. DeBardeleben, *et al.*, "Feng shui of supercomputer memory positional effects in dram and sram faults," in *SC*, pp. 1–11, 2013.

[30] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "Sd-pcm: Constructing reliable super dense phase change memory under write disturbance," in *ASPLOS*, pp. 19–31, 2015.

[31] W. Xu, J. Liu, and T. Zhang, "Data manipulation techniques to reduce phase change memory write energy," in *ISLPED*, pp. 237–242, 2009.

[32] H. Yoon, J. Meza, N. Muralimanohar, N. P. Jouppi, and O. Mutlu, "Efficient data mapping and buffering techniques for multilevel cell phase-change memories," *ACM TACO*, Dec. 2014.

[33] D. H. Yoon, N. Muralimanohar, J. Chang, *et al.*, "Free-p: Protecting non-volatile memory against both hard and soft errors," in *HPCA*, pp. 466–477, 2011.

[34] W. Zhang and T. Li, "Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system," in *DSN*, pp. 197–208, 2011.

[35] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *ISCA*, pp. 14–23, 2009.