

# Effective Write Disturbance Mitigation Encoding Scheme for High-density PCM

Muhammad Imran\*, Taehyun Kwon<sup>†,‡</sup> and Joon-Sung Yang<sup>§</sup>

\*Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, Korea (imran@skku.edu)

<sup>†</sup>Department of Semiconductor and Display Engineering, Sungkyunkwan University, Suwon, Korea (th.kwon@skku.edu)

<sup>‡</sup>System LSI Division, Samsung Electronics, Korea

<sup>§</sup>Department of Systems Semiconductor Engineering, Yonsei University, Seoul, Korea (js.yang@yonsei.ac.kr)

**Abstract**—Write Disturbance (WD) is a crucial reliability concern in a high-density PCM with below 20nm scaling. WD occurs because of the inter-cell heat transfer during a RESET operation. Being dependent on the type of programming pulse and the state of the vulnerable cell, WD is significantly impacted by the data patterns. Existing encoding techniques to mitigate WD reduce the percentage of a single WD-vulnerable pattern in the data. However, it is observed that reducing the frequency of a single bit pattern may not be effective to mitigate WD for certain data patterns. This work proposes a significantly more effective encoding method which minimizes the number of vulnerable cells instead of a single bit pattern. The proposed method mitigates WD both within a word-line and across the bit-lines. In addition to WD-mitigation, the proposed method encodes the data to minimize the bit flips, thus improving the memory lifetime compared to the conventional WD-mitigation techniques. Our evaluation using SPEC CPU2006 benchmarks shows that the proposed method can reduce the aggregate (word-line+bit-line) WD errors by 42% compared to the existing state-of-the-art (SD-PCM). Compared to the state-of-the-art SD-PCM method, the proposed method improves the average write time, instructions-per-cycle (IPC) and write energy by 12%, 12% and 9%, respectively, by reducing the frequency of verify and correct operations to address WD errors. With reduction in bit flips, memory lifetime is also improved by 18% to 37% compared to SD-PCM, given an asymmetric cost of the bit flips. By integrating with the orthogonal techniques of SD-PCM, the proposed method can further enhance the performance and energy efficiency.

**Index Terms**—Error Correction, Phase Change Memory (PCM), Write Disturbance (WD)

## I. INTRODUCTION

Phase Change Memory (PCM), with its better scaling potential, is a promising technology for the future memory systems [1]–[4]. Given its unique features, it may be used at various levels in the memory hierarchy [1].

PCM stores data by programming a phase-changing chalcogenide material to a low-resistance crystalline or a high-resistance amorphous state. While a single-level cell (SLC) PCM stores one bit per cell, a multi-level cell (MLC) PCM can store multiple bits per cell by partitioning the resistance range into multiple storage levels. Major reliability challenges in PCM include resistance drift in MLC PCM [1] and Write

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea by the Ministry of Education under Grant NRF-2018R1D1A1B07049842, in part by the KIAIT (Korea Institute for Advancement of Technology) grant funded by the Korea Government (MOTIE : Ministry of Trade Industry and Energy). (No. N0001883, HRD Program for Intelligent semiconductor Industry), in part by the MOTIE (Ministry of Trade, Industry Energy (10080594) and KSRC (Korea Semiconductor Research Consortium) support program for the development of the future semiconductor device.

Disturbance (WD) in a high-density (SLC and MLC) PCM with scaling below 20nm [2]–[5].

Programming to the high-resistance amorphous state (RESET) requires melting (using a high-current pulse) and cooling the chalcogenide material. The high temperature during RESET can disturb the contents of the adjacent cells [2]. The cells programmed together are not vulnerable to WD. However, in PCM, because of a higher programming current and to improve the memory lifetime, not all the cells are written together. A data-comparison write (DCW) [6] or a flip-and-write (FnW) [7] technique is used to reduce the number of bit flips (written cells). With such techniques, the cells which are not written can be vulnerable to WD. Only a cell in the amorphous state is vulnerable to WD because the impacting temperature is enough to crystallize it yet not enough to melt a crystalline cell [2]. This makes WD a data-pattern-dependent problem. Encoding techniques [2], [3] have been proposed to reduce the frequency of a WD-vulnerable pattern. By mitigating WD, the frequency of a verify and correct (VnC) operation can be reduced. VnC is necessary to ensure correct operation even when a single cell fails due to WD. Less frequent VnC operations can improve performance, energy efficiency and memory lifetime.

While the existing encoding methods reduce the frequency of a single WD-vulnerable pattern in the data, they ignore the impact of other patterns which can also cause WD. Moreover, the existing state-of-the-art methods rely on data compression which limits their effectiveness to highly compressible data. Based on these observations, we propose a novel encoding method which minimizes the number of vulnerable cells instead of a certain data pattern. The proposed method mitigates WD both within the word-lines and across the bit-lines and is more generally applicable, regardless of the data type. The existing encoding methods generally lead to higher bit flips due to a reduced effectiveness of the data-comparison write. Keeping this in view, the proposed method mitigates WD while also minimizing the bit flips. With these two features, the proposed encoding method is more effective than the existing techniques in mitigating WD and also improving the memory lifetime.

Rest of the paper is organized as follows. Background, related works and motivation are summarized in Section II. The proposed method and its implementation are described in Section III. Section IV evaluates the effectiveness as well as the overhead of the proposed method. Finally, a conclusion is given in Section V.

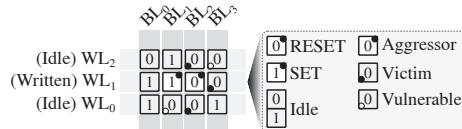


Fig. 1. Write Disturbance in PCM

## II. BACKGROUND AND MOTIVATION

### A. Write Disturbance in PCM

In PCM programming, high temperature from a RESET operation may disturb a neighboring cell in the amorphous state. With a minimal inter-cell spacing, WD may occur both within a word-line and across the bit-lines. Due to a slower heat decay across the bit-lines as compared to the word-lines [5], bit-line WD errors are more frequent. Fig. 1 illustrates WD in PCM. As shown in Fig. 1, word-line WL<sub>1</sub> is being written while the adjacent word-lines WL<sub>0</sub> and WL<sub>2</sub> are idle (not accessed). Cell being RESET is termed as *aggressor* (WD-causing) while an idle cell in the amorphous state immediately next to it is termed as *victim*. A RESET operation to restore the value of a disturbed cell can lead to more *victims*. Such cells that can be *victims* in a domino effect are termed as *vulnerable* in Fig. 1. This example illustrates how a single aggressor cell can cause multiple cells to be WD-vulnerable.

Not all the *victim* cells actually fail [2] and the number of failing cells follows a Weibull distribution [2]. A model to estimate WD error rate has been introduced in [2]. Based on this model, in SLC PCM, the failure rate is 9.9% (of the *victims*) within a word-line and 11.5% across the bit-lines [5].

### B. Existing State-of-the-art in PCM WD-mitigation

The general approach for WD-mitigation has been to encode the data to reduce WD within a word-line and use architectural enhancements to address WD in the bit-lines. Fewer word-line errors result in fewer aggressor cells for causing WD across the bit-lines. Ultimately, a verify and correct (VnC) method ensures a reliable operation. With fewer WD errors, performance can be improved by reducing the VnC frequency.

Encoding techniques to mitigate WD within a word-line include Data Insulation (DIN) [2] and Data Modification with Partitioning (DMPart) [3]. Both techniques reduce a WD-vulnerable data pattern. DIN performs compression and encodes the compressed data by selecting less WD-vulnerable patterns. DMPart reduces a (2-bit) WD-vulnerable pattern by replacing it with the least frequent pattern in the original data. Unlike DIN which uses compression for the redundant bits, DMPart relies on additional cells to store the auxiliary bits.

SD-PCM [5] uses DIN to mitigate WD within the word-lines and three additional techniques to address the bit-line WD errors. Firstly, it uses the available Error Correcting Pointers (ECP) to record the locations of the failed cells. VnC can be skipped if ECP can handle a small number of bit-line WD errors. ECP has been proposed by [8] to address stuck-at faults in PCM. Additionally, [5] has proposed early read for the adjacent rows when the word-line to be written is still in the write queue and to disable certain rows to avoid a VnC operation for those rows.

Old Data	<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> </table>	1	1	1	0	0	1	0	1	1	0	1	1	0
1	1	1	0	0	1	0	1	1	0	1	1	0		
New Data	<table border="1"> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr> </table>	0	0	1	0	1	1	0	1	1	1	0	0	1
0	0	1	0	1	1	0	1	1	1	0	0	1		
DCW	<table border="1"> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> </table>	0	1	1	0	1	1	0	1	1	0	1	1	0
0	1	1	0	1	1	0	1	1	0	1	1	0		
DMPart	<table border="1"> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> </table>	0	1	1	1	1	0	0	1	0	0	1	1	0
0	1	1	1	1	0	0	1	0	0	1	1	0		
	Aggressors = 4, Victims = 5, Bit Flips = 7, Min. Pattern: 01 Aggressors = 5, Victims = 2, Bit Flips = 9, Min. Pattern: 00													

Fig. 2. A case where minimizing the 00 pattern worsens the WD

ADAM [4] uses compression and an alternate left/right storage alignment for even/odd rows to reduce the overlap of useful data. This is effective in reducing the bit-line errors. [4] has also proposed to skip VnC for the cached data.

We compare the proposed encoding method with DIN, SD-PCM (use of ECP), DMPart and ADAM (alternate storage alignment) for both the word-line and the bit-line errors. The orthogonal techniques of SD-PCM (early read and disabling rows) and ADAM (skipping VnC for the cached data) are not considered in the comparison.

### C. Motivation

With writing a 0 to represent a RESET operation in PCM, the conventional encoding techniques such as DIN and DMPart reduce the frequency of a 2-bit 00 pattern in the data. However, the patterns 10 and 01 together can also lead to a WD-vulnerable data pattern. With a reduction in the frequency of a 00 pattern in the data, 10 and 01 patterns become more frequent. Thus, depending on the data, DIN and DMPart may or may not be effective in reducing the number of *victim* cells. This is demonstrated in Fig. 2 where a simple data-comparison write (DCW) is better than encoding such as DMPart to minimize the frequency of 00 pattern. In this case, DMPart increases the number of *victims*, bit flips and the aggressor cells. Therefore, the conventional encoding techniques perform poorer than DCW for certain workloads, as confirmed by the evaluation in Section IV. Even if these techniques successfully reduce WD within a word-line, they do not guarantee to reduce the bit-line errors. Therefore, these methods require extra inter-cell spacing across the bit-lines which may reduce the storage density. Moreover, DIN, SD-PCM and ADAM rely on data compression and, therefore, would not be a general solution. With a varying size of the compressed data, bit flips are also increased. Considering these drawbacks of the conventional methods, we propose ‘MinWD’ encoding which minimizes WD both within the word-lines and the bit-lines by minimizing the number of *victim* cells, regardless of the frequency of various data patterns and the compressibility of data. By considering the actual number of WD-vulnerable cells, the proposed encoding effectively mitigates WD. Additionally, MinWD encodes the data to minimize the bit flips as well.

## III. MINWD ENCODING

The proposed MinWD encoding uses *level shifting* to minimize the number of *victims* and the bit flips. *level shifting* is a simple encoding concept in which every 2-bit pattern is shifted to the next 2-bit binary sequence. For example a one-level shift in ‘00 01 10 11’ leads to ‘01 10 11 00’ while a two-level shift leads to ‘10 11 00 01’. MinWD applies the four possible level shift encodings to the new data. From the four shift encodings of the new data, the one which leads to

### Algorithm 1 MinWD Encoding

```

1: Write(NewData, Address)
2: {OldData, AdjLineA, AdjLineB} = Read(Address);
3: Shifted0 = NewData;
4: Victims0 = CountVictims(NewData, OldData, AdjLineA, AdjLineB);
5: BitFlips0 = CountBitFlips(NewData, OldData);
6: for i = 1 to 3 do
7:   Shiftedi = Shift(NewData, i levels);
8:   Victimsi = CountVictims(Shiftedi, OldData, AdjLineA, AdjLineB);
9:   BitFlipsi = CountBitFlips(Shiftedi, OldData);
10: end for
11: MinVictims = Min(Victims0, Victims1, Victims2, Victims3);
12: Choices = {i | Victimsi == MinVictims};
13: MinBitFlips = Min{BitFlipsi | i ∈ Choices};
14: EncodedData = Shiftedi | BitFlipsi == MinBitFlips, i ∈ Choices;

```

the minimum number of *victims* (in the accessed word-line and the two adjacent word-lines) and the bit flips is chosen as encoded data. Algorithm 1 shows an algorithm for the proposed encoding.

As shown in Algorithm 1, prior to a write, the old data from the accessed word-line as well as the adjacent word-lines (AdjLine<sub>A</sub> & AdjLine<sub>B</sub>) is read (line 2). Note that this read operation is common in all WD-mitigation techniques which address WD both within the word-lines and the bit-lines. *level shifting* leads to four possible encodings (*Shifted*<sub>0</sub> - *Shifted*<sub>3</sub>) of the new data (Line 3 and 7). The number of *victims* (in the accessed as well as the adjacent word-lines) and the bit flips are counted for each of the four encodings of the new data (line 4-5 and 8-9). A *victim* is a cell in the amorphous state next to the cell being RESET (Fig. 1). A comparison finds the encodings (*Choices*) which lead to the minimum number of *victims* (line 11-12). Finally, from the possible *Choices*, the one which leads to the minimum bit flips is chosen as the encoded data (line 13-14). Unlike the conventional encoding techniques, MinWD reduces the number of WD-vulnerable cells without regards to the frequency of different patterns and it considers both the word-line and the bit-line errors together. For decoding, similar to DMPart [3], MinWD adds two auxiliary bits per encoding block, thus having the same storage overhead.

Fig. 3 shows an example of the proposed encoding. The example shows 16-bit old and new data for the accessed word-line and the data in the two adjacent word-lines (AdjLine<sub>A</sub> & AdjLine<sub>B</sub>). The number of *victims* (word-line WL + bit-lines BL) and the bit flips for different encodings of the new data are shown. A data-comparison write leads to 7 (4 WL + 3 BL) *victims* while *Shifted*<sub>2</sub> reduces the number of *victims* to 0 i.e. completely eliminating WD both within the word-line and the bit-lines. Thus, MinWD selects *Shifted*<sub>2</sub> as the encoded data. The example also demonstrates the difference of MinWD with the vulnerable-pattern-reduction techniques such as DMPart. The old data in the accessed as well as the adjacent word-lines is chosen to have the minimum number of 00 pattern (i.e. conventionally encoded). If DMPart is applied to the new data, the encoded data (bottommost), obtained using an XOR operation between the new data and the pattern 11 (the least-frequent pattern), has the minimum number of 00 pattern but the number of victims is 4 (unlike 0 in MinWD). In this case, DMPart reduces WD within the word-line but WD in the bit-lines remains the same. This example demonstrates

AdjLine <sub>A</sub>	1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1	RESET	0 Victim
Old Data	1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 1	SET	0 1 Idle
AdjLine <sub>B</sub>	1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1	(WL+BL)	
New Data	0 0 0 1 1 0 0 1 0 0 0 0 1 0 1	Victims = 7 (4+3), BitFlips = 8	
DCW → Shifted <sub>0</sub>	0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 1	Victims = 3 (0+3), BitFlips = 10	
Shifted <sub>1</sub>	0 1 1 0 1 1 0 1 0 0 1 0 1 1 0 1	Victims = 0 (0+0), BitFlips = 4	
Shifted <sub>2</sub>	1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1	Victims = 9 (3+6), BitFlips = 10	
Shifted <sub>3</sub>	1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 0	Victims = 4 (1+3), BitFlips = 8	
DMPart	1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0	Victims = 4 (1+3), BitFlips = 8	

Fig. 3. Example of the proposed MinWD Encoding

that reducing the actual number of *victims* is more effective in mitigating WD than reducing a certain bit pattern. By completely eliminating WD in the given example, MinWD avoids a VnC operation, thus improving performance, energy efficiency and memory lifetime. In the given example, MinWD minimizes WD while also minimizing the number of bit flips. The bit flip reduction mechanism of MinWD is effective in choosing the encoded data to have the minimum number of bit flips when more than one shift encodings of the new data lead to the same number of *victims* (practically there can be many such cases). Moreover, a reduction in the aggregate (WL+BL) WD errors also reduces the overall number of bit flips.

Keeping the explanation simple, the example in Fig. 3 does not show the impact of WD in the auxiliary bits (needed for decoding). In our evaluation, however, the auxiliary bits are considered as well. The number of *victims* and the bit flips are counted by comparing the encodings (including the auxiliary bits) of the new data to the old encoded data (with the auxiliary bits) in the accessed as well as the two adjacent word-lines.

#### A. Implementation of MinWD

MinWD encoder and decoder can be implemented on chip. Fig. 4(a) shows the overall architecture and the encoder control and datapath logic are shown in Fig. 4(b) and Fig. 4(c), respectively. Decoding logic would be similar to that shown in Fig. 4(c) (with an interchange of Shift 1 and Shift 3 logic). As illustrated in Fig. 4(b), the *Choice* for the encoded data from the four shift encodings (*Shifted*<sub>0</sub> - *Shifted*<sub>3</sub>) of the new data is based on the number of *Victims* and *BitFlips*. *level shifting* can be implemented using a simple logic as shown in Fig. 4(c). Among the four encodings, those with the minimum number of *Victims* are determined using ‘Check MinWD’ block. Meanwhile, the four choices are sorted in a descending order of the number of bit flips. This sorting can be efficiently implemented with a few comparators. A simple 4:2 priority encoder can be used to prioritize the choices (with the minimum number of *victims*) based on the number of bit flips. With a priority encoder, a choice which leads to the minimum bit flips as well as the minimum number of *victims*, is selected as the encoded data without considering the other choices.

Note that the overhead to read the adjacent word-lines is same in all WD-mitigation techniques because this operation is required prior to a write so that a post-write comparison may identify the cells affected by WD. A pre-read when the word-line to be written is still in the queue can be used to avoid any performance overhead, as proposed in [5]. The area, performance and energy overheads for the proposed implementation of MinWD are discussed in Section IV-B.

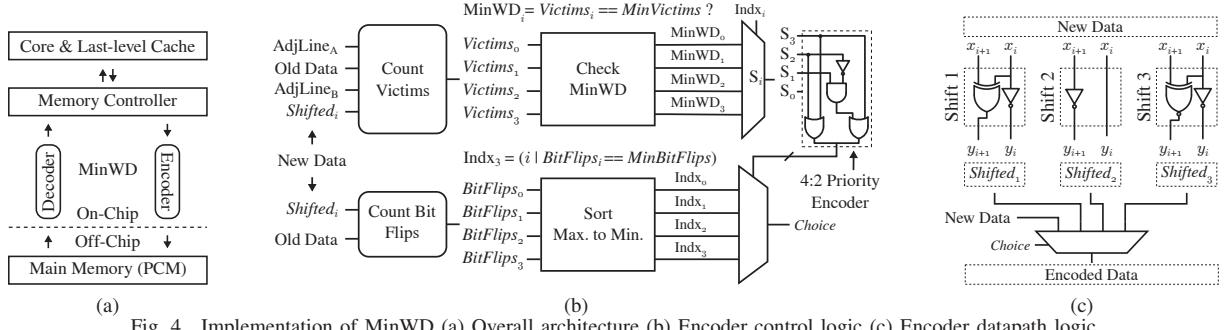


Fig. 4. Implementation of MinWD (a) Overall architecture (b) Encoder control logic (c) Encoder datapath logic

#### IV. EVALUATION

To evaluate the proposed encoding and compare it with the existing relevant works, we consider a trace-driven simulation based on the memory traces (address & 64B data) obtained by running 26 benchmarks from the SPEC CPU2006 benchmark suite [9] using gem5 CPU Simulator [10]. The system consists of an X86, 4-core, 2GHz, out-of-order CPU with L1 data and instruction caches of 32KB each and an L2 cache of 1MB. The main memory is 8 GB SLC PCM, organized as a single channel, single rank with x8 devices per rank. For each benchmark, we execute 10B instructions and record up to a maximum of 10M memory (write) accesses, thus considering a fairly large number of instructions for the workloads with less frequent writes and a reasonable number of accesses for the memory-intensive workloads. We consider SLC PCM SET and RESET latencies based on [5] and write and read energy values based on [3]. For estimating the instructions-per-cycle (IPC) in different methods, we perform a system-level simulation using gem5 with NVMain 2.0 [11] to model SLC PCM.

The proposed MinWD encoding is compared with a data-comparison write (DCW) [6], flip-and-write (FnW) [7], DIN [2], SD-PCM (using WD-free ECP-6 to correct 6 bit-line WD errors in each 64B line) [5], DMPart [3] and ADAM (compression and alternate storage alignment) [4]. Frequent pattern compression [12] is considered for DIN, SD-PCM (using DIN for word-line errors) and ADAM. VnC is used in all the techniques to address WD errors and a write does not finish until there are no more WD errors. We consider WD both within the word-lines and the bit-lines and consider the WD error rates based on the WD model in [2], [5]. The techniques which are orthogonal to the proposed work such as early read [5], disabling rows [5], skipping VnC for cached data [4] and page remapping [13], are not considered in the

comparison. These techniques can be integrated with MinWD for further enhancements in performance. The encoding block size for MinWD is chosen based on reduction in WD errors and the number of bit-flips (Section IV-A).

##### A. Analysis on Various Encoding Block Sizes and WD Errors

The effectiveness of encoding techniques usually varies with the encoding block size. For any encoding-based WD-mitigation technique, the two important aspects are WD errors and the bit flips. Reduction in WD errors improves performance and energy efficiency by reducing the VnC frequency. Since WD can also be eliminated by writing all the cells together, the number of bit flips is an important consideration for different encoding methods. Keeping in view these two aspects, we consider WD errors and the bit flips for various WD-mitigation methods by varying the encoding block size. The encoding block size for DMPart, FnW and MinWD is varied, while DCW, DIN, SD-PCM and ADAM have a fixed block size (64B). The geometric means of WD word-line, bit-line and aggregate errors as well as the bit flips (normalized to DCW) are depicted in Fig. 5. The results indicate that MinWD has fewer aggregate WD errors compared to the conventional techniques, regardless of the encoding block size. DIN and DMPart reduce word-line errors, however, they have higher bit-line errors. On contrary, ADAM is effective in reducing only the bit-line errors. With the increasing block size, the bit flips also increase as shown in Fig. 5(d). With the bit flip reduction mechanism of MinWD, the increase in bit flips with the increasing block size is much less as compared to DMPart. Though FnW has the least number of bit flips, it has significantly higher WD errors compared to MinWD.

Although MinWD mitigates aggregate WD errors for any encoding block size, for the sake of comparison, we choose the encoding block size to be 16-bit, considering the fact that

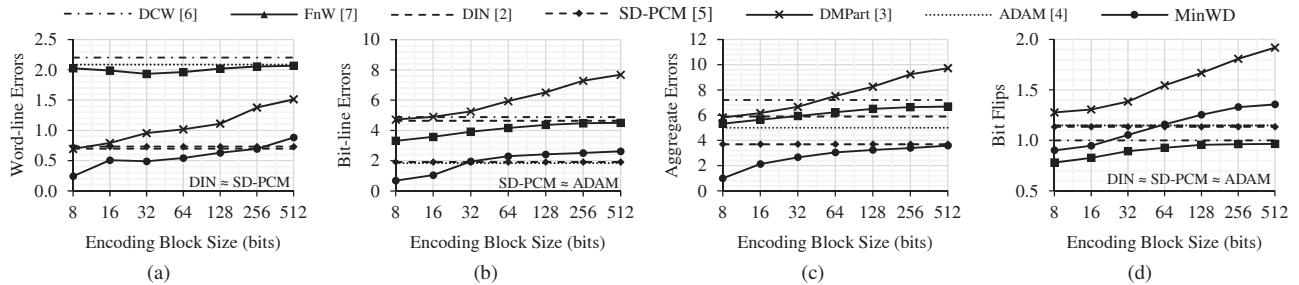


Fig. 5. Geometric mean of (a) Word-line Errors (b) Bit-line Errors (c) Aggregate Errors (d) Normalized (to DCW) Bit Flips, for various encoding block sizes

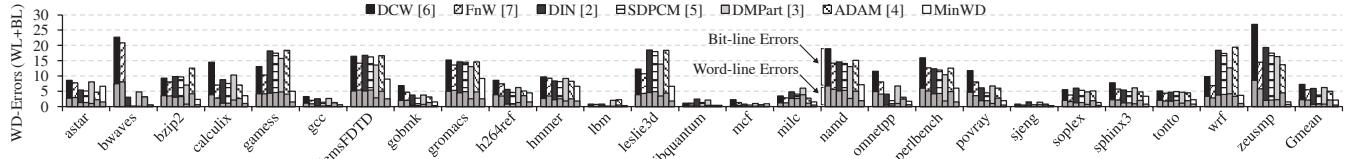


Fig. 6. Write Disturbance (Word-line + Bit-line) Errors in different WD-mitigation techniques

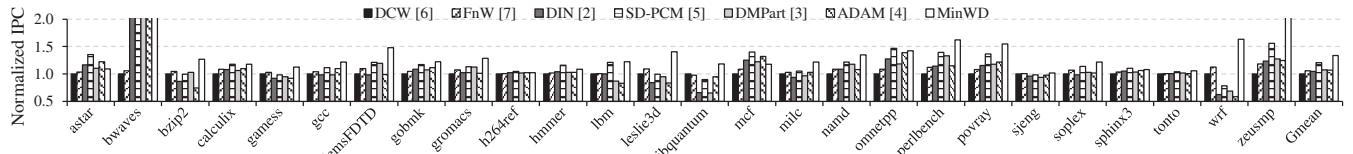


Fig. 7. Normalized (to DCW) instructions-per-cycle values for different WD-mitigation methods

the mean of bit flips at this block size is lower than DCW and the other techniques (except FnW) while both the word-line and the bit-line WD errors are also less than those in all other techniques. This results in a storage overhead of 12.5% for the auxiliary bits, which has been considered be nominal for various data encoding and error correction techniques [8]. Considering the same storage overhead, we choose the block sizes of 16-bit and 8-bit, respectively, for DMPart and FnW.

For the selected encoding block sizes, WD errors for all workloads are shown in Fig. 6. Among the conventional methods, DIN performs best in reducing the word-line errors, ADAM in reducing the bit-line errors and SD-PCM in reducing the aggregate errors. The effectiveness of these methods varies depending on the data patterns and compressibility of the data. Note that, ADAM, when integrated with the orthogonal technique of skipping VnC for cached data, shows a better performance than SD-PCM in [4]. However, considering only the alternate storage alignment technique, ADAM is less effective than SD-PCM because it relies on the data compressibility while SD-PCM can use ECP to correct up to 6 errors in the bit-lines, regardless of the data compressibility.

MinWD, which reduces the actual number of WD-vulnerable bits, is more effective than all the existing methods, regardless of the type of data. The geometric mean of aggregate errors for DCW, FnW, DIN, SD-PCM, DMPart, ADAM and MinWD is 7.20, 5.47, 5.89, 3.69, 6.16, 5.00 and 2.12, respectively. Compared to DCW, MinWD reduces the word-line, bit-line and the aggregate errors by 77% (27% compared to DIN), 78% (43% compared to ADAM) and 70% (42% compared to SD-PCM), respectively.

### B. Hardware Overhead of MinWD

To evaluate the area, latency and energy overheads of MinWD, the proposed architecture shown in Fig. 4 is implemented using a 28nm standard cell library. With a 16-bit block size, the encoder in Fig. 4 requires an area of  $2783\mu m^2$ . The encoding path delay is  $3.25ns$  and the energy consumption is  $3.7pJ$ . The (16-bit) decoder's area, delay and energy consumption are  $92\mu m^2$ ,  $0.26ns$  and  $4.11fJ$ , respectively. With CPU operating much faster than memory, typical PCM latencies in hundreds of nanoseconds (especially with the multiple writes to eliminate WD) and only decoding being on the performance-critical read path, the impact of

TABLE I  
RESULTS FOR PERFORMANCE, ENERGY EFFICIENCY AND LIFETIME

Method	Writes	Verifies	Write Latency	IPC	Write Energy	Bit Flips
DCW	1x	1x	1x	1x	1x	1x
FnW	0.89x	0.89x	0.90x	1.06x	0.88x	0.78x
DIN	0.87x	0.95x	0.93x	1.04x	0.95x	1.15x
SD-PCM	0.94x	0.59x	0.70	1.21x	0.71x	1.13x
DMPart	0.85x	0.92x	0.90x	1.07x	0.95x	1.30x
ADAM	0.82x	0.89x	0.88x	1.07x	0.89x	1.15x
MinWD	0.57x	0.55x	0.58x	1.33x	0.62x	0.95x

MinWD encoding/decoding on the performance would be negligible. We incorporate the impact of encoding energy (for 64B array) in estimating the write energy.

### C. Performance, Write Energy and Memory Lifetime

Performance is compared in terms of the effective write latency (including the time for VnC to eliminate WD errors) and the instructions-per-cycle (IPC). Write energy is estimated based on the average number of SET, RESET and verify operations per write for different WD-mitigation methods. For MinWD, the encoding energy is also considered. Memory lifetime is compared based on the number and the cost of the bit flips. A comprehensive summary of various results (geometric means) is given in Table I. DCW is considered as a baseline for comparison and the results for other techniques are normalized to it.

Note that *Writes* and *Verifies* in Table I refer to the number of write and verify operations required to finish one actual write request. These reflect the frequency of VnC to eliminate WD during a write. ADAM reduces the bit-line errors by reducing the overlap between the useful data. This helps to reduce the number of writes but the the frequency of necessary verify operations is not significantly reduced. SD-PCM is more effective in reducing the verify operations by using ECP. However, it does not reduce the write operations significantly. This is because of the additional writes to ECP as well as the writes when the errors are beyond the correction capability of ECP. MinWD reduces both the write and verify operations more significantly by eliminating WD earlier than the other techniques. MinWD reduces the effective write latency by, 42%, 30% and 12% compared to DCW, ADAM and SD-PCM, respectively. Fig. 7 compares the results for IPC. Memory-intensive workloads show more improvement in performance due to a reduced write latency by MinWD. On average,

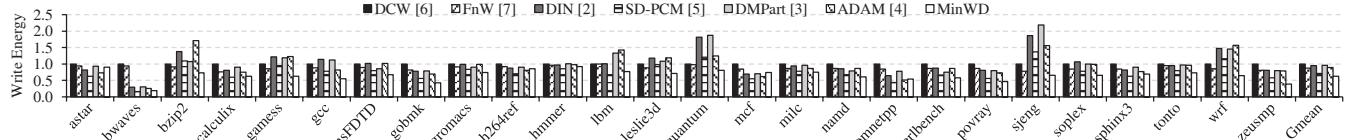


Fig. 8. Average write energy (normalized to DCW) in different WD-mitigation techniques

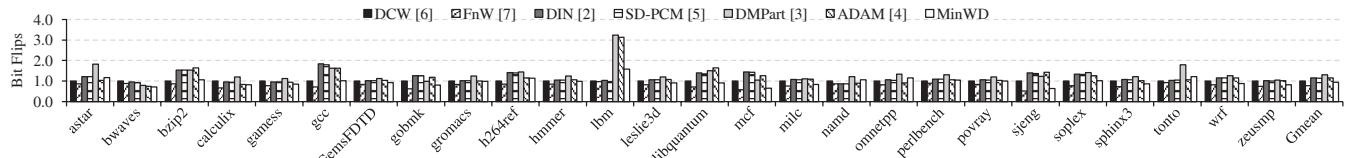


Fig. 9. Average bit flips (normalized to DCW) in different WD-mitigation techniques

MinWD improves the IPC by 33%, 26% and 12% compared to DCW, ADAM and SD-PCM, respectively.

The results for the average write energy (normalized to DCW) are shown in Fig. 8. Improvement in write energy comes mostly by the reduction in the number of verify operations. MinWD is 38% more energy-efficient than DCW and 9% more than SD-PCM.

Fig. 9 compares the average number of bit flips (normalized to DCW) per write for different methods. Only the bit flips in data chip (and not the ECP chip) are considered for SD-PCM. FnW has the least number of bit flips. However, MinWD significantly improves the performance and write energy by reducing WD compared to FnW. MinWD reduces the bit flips by 5% and 18% compared to DCW and SD-PCM, respectively. A RESET operation is considered to be more detrimental to cell endurance than a SET [14]. With fewer WD errors, MinWD also has 38% and 58% less frequent RESET operations compared to DCW and SD-PCM, respectively. Considering asymmetric cost of bit flips, MinWD can have higher improvement in the lifetime compared to DCW and SD-PCM. If a RESET is assumed to be twice as costly as a SET [14], MinWD reduces the cost of bit flips by 15% and 37% compared to DCW and SD-PCM, respectively.

Although, the proposed method has been compared against SD-PCM (using ECP), however, it is also orthogonal to the use of ECP for correcting the bit-line errors. By integrating them, it can further enhance the performance and energy efficiency. Moreover, it can also use data compression to mitigate the storage overhead of the auxiliary bits.

## V. CONCLUSION

Write disturbance is a critical reliability concern in a high-density PCM with below 20nm technology node. The data-dependent nature of this problem has motivated data encoding techniques to solve this problem. While the conventional WD-mitigation methods try to reduce the probability of WD by mitigating certain data patterns, this paper introduces a more effective method which directly reduces the number of WD-vulnerable cells as well as the bit flips. Moreover, the proposed method considers WD both within the word-lines and the bit-lines, unlike the existing encoding techniques. The proposed method is more generally applicable regardless of the data type. Experimental evaluations show an average reduction of

42% in WD errors, 12% in write time and 9% in write energy compared to the existing state-of-the-art (SD-PCM). With reduction in average write time, IPC is improved by 12%. The proposed method also improves the memory lifetime, by 18% reduction in the bit flips compared to SD-PCM.

## REFERENCES

- [1] S. Yu and P. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, Spring 2016.
- [2] L. Jiang, Y. Zhang, and J. Yang, "Mitigating write disturbance in super-dense phase change memories," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2014, pp. 216–227.
- [3] M. K. Tavana and D. Kaeli, "Cost-effective write disturbance mitigation techniques for advancing pcm density," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2017, pp. 253–260.
- [4] S. Swami and K. Mohanram, "Adam: Architecture for write disturbance mitigation in scaled phase change memory," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2018, pp. 1235–1240.
- [5] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "Sd-pcm: Constructing reliable super dense phase change memory under write disturbance," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '15. New York, NY, USA: ACM, 2015, pp. 19–31.
- [6] B. Yang *et al.*, "A low power phase-change random access memory using a data-comparison write scheme," in *2007 IEEE International Symposium on Circuits and Systems*, May 2007, pp. 3014–3017.
- [7] S. Cho and H. Lee, "Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2009, pp. 347–357.
- [8] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ecp, not ecc, for hard failures in resistive memories," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 141–152.
- [9] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.
- [10] Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.
- [11] M. Poremba, T. Zhang, and Y. Xie, "Nvmain 2.0: A user-friendly memory simulator to model (non)-volatile memory systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140–143, July 2015.
- [12] A. R. Alameldeen and D. A. Wood, "Frequent pattern compression: A significance-based compression scheme for l2 caches," in *University of WisconsinMadison, Technical Reports*, 2004.
- [13] R. Wang, S. Mittal, Y. Zhang, and J. Yang, "Decongest: Accelerating super-dense pcm under write disturbance by hot page remapping," *IEEE Computer Architecture Letters*, vol. 16, no. 2, pp. 107–110, July 2017.
- [14] R. Maddah, S. M. Seyedzadeh, and R. Melhem, "Cafo: Cost aware flip optimization for asymmetric memories," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2015, pp. 320–330.