2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks

Mitigating Write Disturbance in Super Dense Phase Change Memories

Lei Jiang † Youtao Zhang ‡ Jun Yang †
† *Electrical and Computer Engineering Department* ‡ *Computer Science Department*
University of Pittsburgh, Pittsburgh *University of Pittsburgh, Pittsburgh*
†{*lej16,juy9*}*@pitt.edu* ‡*zhangyt@cs.pitt.edu*

***Abstract*—Constructing a highly scalable and dense main memory subsystem with large access bandwidth has become a major challenge for modern computing systems. Traditional memory technologies, like DRAM and NAND Flash, suffer from either poor scalability or limited access bandwidth. Recent studies have identified emerging Phase Change Memory (PCM) as one of the most promising low power main memory technology candidates, because of its short read latency and good scalability. However, PCM still faces serious write disturbance problem below 20nm technology.**

Write disturbance leads to more cell programming errors, and thus degrades write reliability. Simple solutions, such as allocating large inter-cell space and adopting strong error correction code (ECC), either reduce memory density or incur large performance overhead. In this paper, we propose DIN, a Data encoding based INsulation technique, to mitigate write disturbance in highly dense PCMs. DIN improves memory density by eliminating inter-cell thermal band along a word-line. The non-negligible disturbance errors, are then minimized by disturbance-aware data encoding, based on how PCM cells are programmed at device level. Our experimental results show that DIN gains write disturbance resistance in high density PCM chips while achieving comparable performance for a wide range of applications.

***Keywords*-Phase Change Memories; Write Disturbance; Error Correcting Code;**

I. Introduction

The unprecedentedly fast technology scaling has enabled the integration of more and more cores in a single chip, resulting in a large demand for main memory in modern computing systems. However, it is challenging to construct a scalable memory subsystem with large density by traditional and mature memory technologies. DRAM faces difficulty in scalability — the path for DRAM to scale down beyond $20nm$ is still unclear [1]. Although NAND Flash has been successfully taped out at $16nm$ [2], it has slow reads and limited cell endurance, which prevent it from being configured as main memory. Recent architectural studies [3], [4], [5] have advocated adopting the emerging Phase Change Memory (PCM) as main memory, due to its fast read speed, 10^8 write cycles cell endurance and better cell scalability [6]. However, when scaling below 20nm technology, PCM faces a non-negligible write disturbance (WD) problem.

This work was supported in part by NSF CSR #1012070, NSF CCF #1242657 and NSF CAREER #0747242.

The WD problem in PCM arises from inter-cell thermal disturbance during programming. A PCM cell uses two stable states, amorphous state and crystalline state of phase change material (GST [6]), to store bit '0' or '1'. Programming PCM cells requires to inject large currents to melt (RESET) or crystallize (SET) phase change material. The heat generated for programming one cell may disseminate beyond this cell and disturb the values stored in its neighboring cells. This is referred to as WD problem in PCM. WD in PCM was first reported at $54nm$ technology [7], and has become non-negligible at and below $20nm$ technology node [8], [9].

The WD in DRAM is trivial as DRAM has low disturbing voltage on sneak paths [10]. NAND Flash, however, suffers from inter-cell WD caused by inter-cell floating gate coupling [11], i.e., programming one Flash page may corrupt the data values of neighboring pages in the same block. The schemes proposed for mitigating WD in NAND Flash include disturbance-aware MLC (multi-level cell) programming [11], adding verify and restore iterations [11], compensated sensing [12], and heavyweight ECC protection [12].

Unfortunately, it is still arduous and challenging to alleviate WD in PCM. Byte-addressable PCM based memory system controls write operations at bit granularity [3], [4]. Moreover, system performance is very sensitive to the access latency of PCM based main memory. Therefore, naïvely and directly adopting WD reduction techniques proposed for Flash, such as adding extra verify iterations and adopting heavyweight ECC, slows write and read operations down, and thus degrades the overall system performance. Even though an enlarged inter-cell space [13], [14] is effective in minimizing WD, it greatly reduces PCM array density and memory capacity within the same die area.

In this paper, we propose **DIN**, a **D**ata encoding based **IN**sulation technique, to mitigate WD in PCM. Our contributions are summarized as follows:

- We build a WD model for PCM. Our model first calculates the interference temperature elevations during a disturbing RESET with different inter-cell thermal bands at each technology node. Based on the temperature elevation, our model computes the disturbance error bit rate for both SLC and MLC PCMs.
- We propose DIN, a data encoding based insulation scheme, to minimize disturbance errors by reducing

978-1-4799-2233-8/14 $31.00 © 2014 IEEE
DOI 10.1109/DSN.2014.32

IEEE computer society

occurrences of WD vulnerable data patterns. DIN allows the reduction of inter-cell spacing along a word-line, which enables the construction of PCM chips with higher density.

- We compare our proposed scheme to existing disturbance mitigation schemes for PCM. Our experimental results show that DIN gains WD resistance in a highly dense PCM based main memory while achieving comparable performance for a wide range of applications.

The rest of our paper is organized as follows. Section II introduces background information on PCM and WD in PCM, and then discusses why existing solutions do not work for PCM. Section III builds our model of PCM WD for both SLC and MLC PCMs. Section IV elaborates the details of our data encoding based WD mitigation scheme. Related works are discussed in Section V. The experimental methodology and results are presented in Section VI and Section VII, respectively. Section VIII concludes this paper.

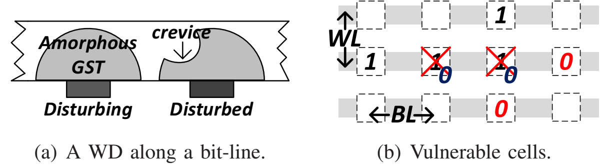
II. Background and Motivation

A. PCM Basics

Phase Change Memory (PCM) [6] stores data using two stable states of $Ge_2Sb_2Te_5$ (GST) chalcogenide: high resistance amorphous RESET state (bit '0') and low resistance crystalline SET state (bit '1'). A PCM cell is programmed by injecting electrical pulses onto Joule heater to heat GST into different resistances. When a short duration but large amplitude current is applied, the PCM cell can be programmed into highly resistant RESET state. During RESET operation, the inner-cell temperature goes above the GST melting point ($600\,°C$). When a long duration but small amplitude current is injected, the target cell is written into the SET state with low resistance. In a SET operation, the inner-cell temperature goes above GST crystallization point ($100\,°C \sim 150\,°C$), but below GST melting point. Due to the large resistance difference between fully amorphous and fully crystalline states, one PCM cell is able to store multiple bits, referred to as MLC (multiple-level cell). On the contrary, the cell storing only one bit is called SLC (single level cell). To ensure programming accuracy and remove non-determinism, MLC PCM widely adopts iteration based programming schemes such as single-RESET-multiple-SET (SRMS) [15] and single-SET-multiple-RESET (SSMR) [16].

B. Write Disturbance

The write disturbance (WD) problem in PCM arises from inter-cell thermal disturbance during programming. When programming a PCM cell, particularly resetting a cell, the generated heat may disseminate to its neighboring cells and thus disturb their stored values. Figure 1(a) shows two cells along one bit-line. A write request may only reset the left cell and leave the right cell unchanged. The latter is referred to as *idle* cell in this paper. Due to the sneak heat flow along the bit-line, if the right cell is in RESET state, a portion of its amorphous volume may crumble into crystalline fractions, which greatly reduces its resistance, losing the stored bit '0' information [8], [9].



(a) A WD along a bit-line. (b) Vulnerable cells.

Figure 1. WD and vulnerable data pattern in PCM.

1) Inter-cell space: Besides the magnitude and width of programming pulses, PCM WD is also heavily affected by the cell-to-cell distance. Previous studies [13], [8] have revealed that WD becomes more significant when cell-to-cell distance is decreased. WD in PCM was first reported at $54nm$ technology node [7]. However, as fast technology scaling keeps reducing inter-cell distance, the WD in PCM has become a non-negligible reliability issue at $20nm$. And it is considered as a major scaling bottleneck for the practical deployment of PCM [8], [9].

2) WD vulnerable data pattern: PCM WD depends on value patterns being written. In particular, WD can occur only between a cell-X that is under RESET, while its neighboring cell-Y that is idle and in RESET state. This is because: (i) Since SET current is less than 50% of RESET current, the temperature increment during SET is about four times lower than that during RESET [17]. Thus, the disturbing influence of **SET** operation can be **ignored** [18]. (ii) Since heat dissemination decays fast horizontally, the temperature that its neighboring cells can reach is always below the melting temperature but may be higher than the crystalline temperature. Thus, a neighboring cell may be **crystallized** if it stays in **RESET** state (i.e., amorphous state), but cannot be melt if it is in SET state (i.e., crystalline state) [8]. Figure 1(b) illustrates the vulnerable PCM cells in red color. For a cell being RESET, its neighbor cell is *vulnerable* if it stores bit '0' and is idle. Vulnerable cells may appear between bit-lines along a word-line or between word-lines along a bit-line. Vulnerable cells do not always produce disturbance errors. However, in the worst case, one RESET may disturb four neighboring cells.

3) Difference from resistance drift: WD is a different problem from resistance drift [19], [20] in PCM. *Resistance drift* indicates the fact that the resistance of a PCM cell spontaneously increases, when there is no operation accessing the cell. Furthermore, it happens at room temperature [19]. Instead, *write disturbance* happens when an amorphous state cell is idle but its neighboring cell is under RESET. Moreover, it can only occur under a higher disturbing temperature (i.e., above crystallization point). The disturbed cell can be considered as "partially" SET.

4) WD exacerbation along a word-line: PCM WD also depends on the memory line programming strategy used in PCM chips. When writing a PCM line, the control logic has

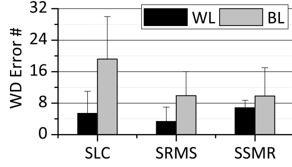


Figure 2. WD errors # (64B).

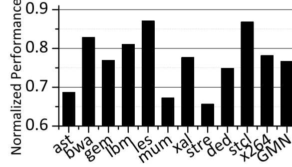


Figure 3. VnR performance.

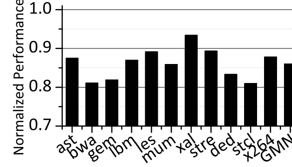


Figure 4. BCH performance.

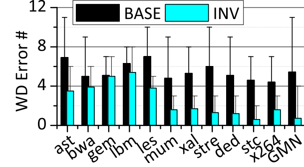


Figure 5. INV WD errors.

the ability to control write operations at cell granularity [3], [4]. A write operation will not have any disturbance error along the word-line, if it always updates all cells in a memory line [18]. Compared to the heat generated by SET or RESET currents, the disseminated heat is low. However, to extend PCM chip lifetime, *Differential Write* [3] and *Flip-N-Write* [4] proposed to only write cells which are actually changed. These schemes leave many cells idle during writes, so disturbance errors may appear along a word-line.

5) WD errors per write: Based on the model we construct in Section III, we study the manifested disturbance errors when writing a PCM line into the device. For the changed cells in Flip-N-Write [4], only RESETs may disturb their neighboring idle cells. And only a subset of these WD vulnerable cells actually manifest disturbance errors.

Figure 2 reports the number of disturbed cells per line write in a PCM chip with no extra inter-cell space, i.e., the inter-cell space is one pitch size. The solid/error bars indicate the average/maximum numbers of errors, respectively. WL denotes the error number in the same PCM line (i.e., the line being written), while BL indicates the error number in two neighboring lines. There are on average 5.4, 3.3 and 6.8 disturbance errors in the same line for SLC, MLC with SRMS and SSMR, respectively. The worst case number of WD errors along the same word-line further increases by 33%-112%. And there are on average 19.2, 9.9 and 9.8 disturbance errors in two neighboring lines for SLC, MLC with SRMS and SSMR, respectively. SRMS (Single-RESET-multiple-SET) and SSMR (Single-SET-multiple-RESET) are two MLC writing strategies, as we will discuss in Section IV. The worst case number of WD errors along the same bit-line further increases by 50%-73%. These results are reported during normal program executions with our baseline configuration in Section VI. And if a program is maliciously constructed, there could be more disturbance errors.

6) WD errors along bit-lines: WD errors along bit-lines also manifest after eliminating inter-cell spacing, as shown in Figure 2. To ensure write reliability, one line write may trigger two correction writes to its neighboring lines. With more lines being disturbed in a cascade way, a PCM write from the core may lead to 1+2^n line writes. From our experiments, one write can introduce 2-4 extra correcting writes, which seriously degrades system performance — 32% on SLC PCM and 48% on MLC PCM. On the contrary, handling WD along a word-line does not need to access extra lines. In the worst case, if proposed WD mitigation schemes fail to eliminate WD errors, the control logic can write all cells in the line and thus eliminate WD errors along the word-line. Unfortunately, writing all cells in the line may generate more WD errors along bit-lines. Therefore, in this paper, we only shrink inter-cell distance along a word-line while keeping enough inter-cell space in bit-line direction. In this paper, our WD mitigation technique, DIN, is proposed to tolerate WD errors along word-lines.

C. The Existing Solutions

We next discuss existing solutions that may be adopted to mitigate write disturbance (WD) in PCM. In this section, we focus on mitigating errors appearing in the same memory line (along the word-line). And we only take SLC as an example, while MLC related results can be found in Section VII. The motivational results in this section rely on the models in Section III and the experimental settings in Section VI.

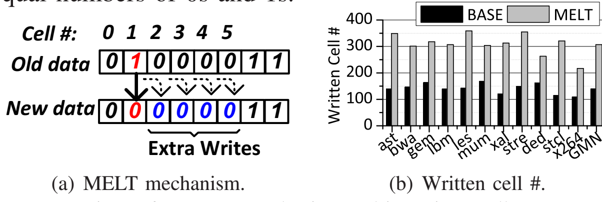
1) Verify and Restore (VnR): Given WD occurs only during RESET, a simple technique is to identify disturbed cells by extra verify operations and, if necessary, restore these cells with their original values. This verify-and-restore (VnR) process may proceed several times before reaching a limit (5 times in this paper). VnR either corrects all WD errors before reaching the limit, or reverts to write all cells in the line if the limit is reached. The latter ensures WD-error-free write along the word-line. VnR integrates well with iteration-based programming strategies, as the latter already needs to verify the resistance of cells being written in each iteration. Checking disturbed cells can be done simultaneously. However, VnR introduces large performance overhead on SLC PCM. Figure 3 shows the performance degradation when VnR is adopted to mitigate WD by SLC PCM. From the figure, VnR exhibits 24% performance loss compared to our baseline.

2) Heavy Weight BCH Code (BCH): Another disturbance mitigation scheme is to add strong error correct code (ECC), such as BCH, to cover as many disturbed cells as possible during writes. To correct t-bit errors in k-bit data, a BCH code typically requires $r = t \cdot ceil(log_2 k) + 1$ redundant parity check bits [21]. From our experiments, one write may disturb 11 cells in a 64B memory line. Therefore, we need at least 100 bits per line, or around 20% extra storage to ensure write reliability. The latency overhead of adopting heavyweight BCH hurts read operation performance, i.e., it takes $48ns$ at $22nm$ [22] to decode this 100-bit BCH code (the storage overhead of the BCH, which can correct 11 errors, is 100 bits). The smaller the number of errors appears,

the faster the decoding operation can become [22]. Figure 4 shows that the 20-bit 11-error correctable BCH code incurs 14% performance degradation on average.

3) Inversion if more 0s in one line (INV): A memory line containing more 0s is more vulnerable to WD. This is because only RESET disturbs its neighboring cells and only neighboring cells in RESET may be disturbed. To mitigate WD, a simple yet effective scheme is to invert a line if it contains more 0s than 1s, denoted as INV. INV stores 0s as 1s and vice verse, and attaches one bit to each line to indicate if the line is inverted or not.

Figure 5 compares the number of disturbance errors when writing a PCM line before and after adopting INV. Both the average (bar) and the maximum (error bar) numbers are reported in the figure. For integer benchmarks, such as $astar$ and $xalancbmk$, INV greatly reduces the number of disturbance errors. However, it is less effective for floating point applications, such as $gemsFDTD$ and lbm. This is because many lines of these benchmarks have approximately equal numbers of 0s and 1s.



(a) MELT mechanism. (b) Written cell #.

Figure 6. MELT mechanism and its written cell #.

4) Melting write (MELT): WD occurs between a cell-X being RESET and a neighboring cell-Y that satisfies two conditions: (i) Y is in RESET state; (ii) Y is idle. If Y is not idle, i.e., X and Y are written simultaneously, it is no longer vulnerable as the heat generated by writing Y dominates and prevents disseminated heat from changing the value of Y. Therefore, another disturbance reduction approach, denoted as MELT, is to identify all vulnerable cells and write these cells even though their values are not changed. Since MELT writes more cells, it has two drawbacks: (i) it wastes write energy; (ii) it shortens PCM chip lifetime.

MELT has to write more cells in a cascade style. For example, in Figure 6(a), cell-1 needs to be RESET while all other cells keep the same value. Since cell-2 is also 0, MELT re-writes it to prevent WD. Accordingly, cell-3 to cell-5 need to be written simultaneously. This process stops until meeting a neighboring cell that stores bit 1 (i.e., cell-6). To quantify the written cell increase, Figure 6(b) compares the average number of written cells per each write in MELT and our baseline. MELT writes 115% more cells on average for SPEC2006, BioBench, STREAM and PARSEC benchmarks. Given PCM endurance remains a major concern, MELT is not a practical approach to mitigate PCM WD.

III. Modeling Write Disturbance

In this section, we present our model on write disturbance (WD). First, we model the temperature elevation among the

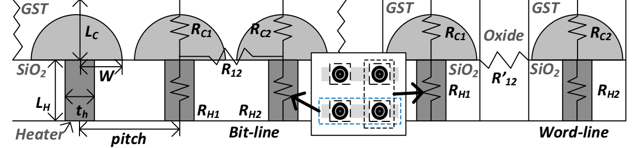


Figure 7. Modeling inter-cell thermal interference.

neighboring cells due to the inter-cell heat dissemination during a RESET operation on the target cell. And then, we model the WD bit error rate caused by the interference temperature elevation for both SLC and MLC.

A. Interference Temperature Elevation

1) PCM cell thermal model: Figure 7 illustrates the cell array structure used in this paper. In the figure, L_C and L_H are the heights of the chalcogenide layer and the heater, respectively; W indicates the width of the GST trench; t_h is the thickness of the thin-film deposited heater; and p represents the pitch, i.e., the distance between identical PCM cells in an array. The typical half-pitch size is the process node size. For example, at $90nm$ technology node, the typical pitch size is $180nm$ [1].

We adopt the inter-cell heat dissemination model from [18] and summarize it in Equation 1 and Equation 2. The model calculates the temperature increment ΔT_1 of a idle cell-X_1 when there is an ongoing RESET operation on its adjacent cell-X_2.

$$\Delta T_1 = \frac{R_1}{R_1 + R_{12}} \cdot \Delta T_2 = \frac{\Delta T_2}{1+\alpha} \quad (1)$$

$$R_i = \frac{1}{\frac{1}{R_{C_i}} + \frac{1}{R_{H_i}}} \quad (i = 1 \ or \ 2) \quad (2)$$

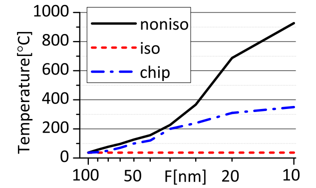


Figure 8. The inter-cell thermal effect with PCM scaling.

where R_i (i=1 or 2) indicates the parallel thermal resistance of each cell; R_{12} is the thermal resistance between two cells; R_{C_i} and R_{H_i} (i=1 or 2) represent the thermal resistance of the GST layer and the heater of each cell, respectively. To ensure reliable RESET, ΔT_2 is 1500K (1226.85 °C) [18]. With a simple transformation, we get $\alpha = R_{12}/R_1$. The value of R_{12} depends on not only the cell characteristics but also the distance between cells. R_{12} is larger if two cells are farther from each other.

2) PCM cell scaling model: The physical parameters of PCM cell scale as technology scales. A parameter is *isotropically* scalable if it can be scaled linearly with feature size. Otherwise, the parameter is *non-isotropically* scalable. Although many parameters of a PCM cell structure can be scaled isotropically, several parameters are non-isotropically scalable. For example, t_h has already been very small ($8nm$) at the $90nm$ technology [18], and can hardly be scaled.

If PCM cell is fully isotropically scalable, all components and their corresponding thermal resistances will scale

proportionally, making α a constant, which keeps inter-cell heat dissemination insignificant, as shown in Figure 8. Another extreme case is that PCM cell is completely non-isotropically scalable, which results in an exponential increase of inter-cell thermal interference and defeats the perspective of adopting PCM in future memory systems. Instead, both PCM research and industrial chips combine isotropic and non-isotropic scaling and make a better trade-off in each process technology generation. We extracted parameters from industrial chips at different technology nodes (i.e., 90nm [23], 58nm [24], 32nm [25] and 20nm [26]), fed these values into PCM RESET scaling model [27] to construct cell parameters (i.e., L_C, L_H, W and t_h), and summarized the thermal scaling effect in Figure 8. The value at 10nm is predicted using the cell parameters at 20nm [28] and the proportional scaling ratio from 32nm to 20nm.

3) Inter-cell space expansion at 20nm: An effective design to mitigate thermal interference is to increase inter-cell space. For 20nm technology, 40nm cell-to-cell space represents the minimal distance, i.e., two cells are placed next to each other. The interference temperature elevation along a word-line for 40nm inter-cell space is 310 °C. A larger cell-to-cell space means a wider thermal band. For example, 60nm inter-cell space has 20nm thermal band between cells, which dramatically reduces the interference temperature elevation from 310 °C to 173 °C. However, the 20nm thermal band along a word-line results in 50% cell area expansion. And 80nm inter-cell space, i.e., 40nm thermal band, only achieves 87 °C interference temperature elevation and has to pay 100% cell area expansion.

B. Write Disturbance Bit Error Rate

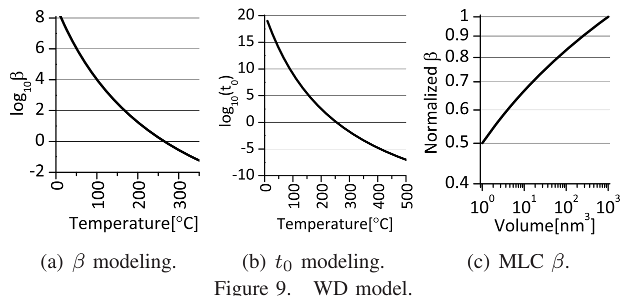
PCM WD is a crystallization process due to inter-cell heat dissemination. Recent studies [29], [13], [14] have constructed a mathematical model to reveal the relationship between the disturbing error rate and the crystallization (disturbing) time. We adopt this model in the paper to quantify disturbance error rate.

1) The erratic characteristic: Data loss due to crystallization exhibits a stochastic behavior. Starting from the same reset initial state, although the number of failed cells caused by crystallization in a PCM array remains consistent, different cells fail at different times [29], [13]. As a result, even though vulnerable cells can be identified deterministically (based on data pattern), the actual failed cells appear randomly and erratically.

2) Modeling thermal disturbance: In [13], the disturbance error probability F follows Weibull distribution and can be calculated by

$$F = 1 - exp(-(\frac{t_{fail}}{t_0})^{\beta}) \quad (3)$$

where t_{fail} indicates cell failure time, i.e., the time a cell spends in losing the stored value; t_0 represents the characteristic failure time for $F = 0.63$; and β is the distribution shape factor deciding the slope of the distribution curve. We adopt the models of t_0 and β from [13], as shown in Figure 9(a) and Figure 9(b), respectively.

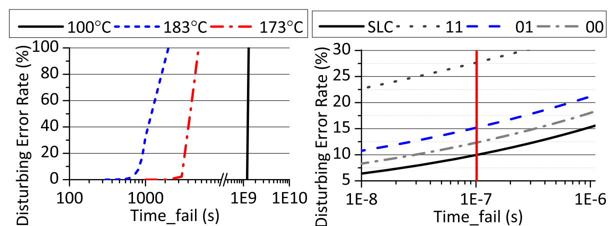


(a) β modeling. (b) t_0 modeling. (c) MLC β.

Figure 9. WD model.

Based on Equation 3, Figure 10 plots the probability of disturbance errors as a function of the time of cells being exposed at different temperatures. For example, at 173 °C, around 0.01% cells are disturbed if they are exposed for 100ns, the width of a RESET pulse. Since t_0 and β decrease fast with increasing temperatures, it takes much less time for disturbance errors to manifest under higher temperatures.

3) Errors on SLC and MLC: Figure 9(c) shows that a larger volume of amorphous GST fraction indicates a larger β in Equation 3. By controlling the volume of amorphous GST in one PCM cell, more than two resistance states can be represented in a PCM cell. Therefore, the fully amorphous resistance state and the other two partially crystalline intermediate resistance states in MLC PCM achieve different disturbance error rates under the same temperature by having different values of β in the disturbance error model. We strictly follow the assumptions in [14]. In this paper, we adopt $10^5\Omega$ ($10^3 nm^3$ in Figure 9(c)) as the resistance of fully amorphous state and $10^3\Omega$ ($0nm^3$) as the fully crystalline state in SLC. And the reference resistance in SLC is $10^4\Omega$. Two MLC intermediate resistances are $10^{3.67}\Omega$ and $10^{4.33}\Omega$. Three reference resistances in MLC are $10^{3.33}\Omega$, $10^4\Omega$ and $10^{4.67}\Omega$. When write thermal disturbance makes the cell resistance degrade below the nearest reference resistance, a WD error occurs.



(a) WD at low temperature. (b) WD at 310 °C.

Figure 10. Disturbance probability at different temperatures.

As Figure 10(a) shows, when exposed to relatively low temperatures (100 °C $\sim$ 183 °C), cells can sustain a much longer time. However, when disturbance appears, nearly all cells get disturbed within a short duration. In Figure 10(b), at 310 °C, the curves at high temperatures have long flat tails of low disturbance error rates, which indicate a small portion

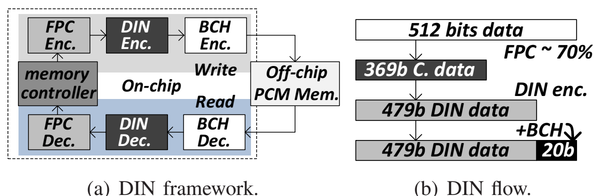
of cells fail much earlier than others. At $310\,^{\circ}$C, 9.9% SLC cells are disturbed after 100ns. The disturbance error rates of three MLC non-fully crystalline states at $310\,^{\circ}$C after 100ns disturbing duration are 12.3%, 15.2% and 27.6%.

4) Thermal band selection: The WD error rate for 60nm inter-cell space is around 10^{-5}, which is smaller than the non-WD bit error rate of 6.74×10^{-5} in a recent PCM main memory system [19]. To make a ideal baseline and conservatively estimate the memory density improvement, we assume our baseline has no WD error and the inter-cell space in our baseline is 60nm at 20nm technology node. In our design, we reduce the inter-cell space along a word-line to $40nm$ and propose mitigation schemes to tolerate increased WD errors.

IV. The Design Details of DIN

In this section, we elaborate DIN, a **D**ata **IN**sulation framework for minimizing write disturbance (WD) in PCM. We first present an overview of the scheme, and then illustrate how to determine appropriate encoding schemes for different SLC and MLC programming strategies. At last, we discuss the hardware implementation issues.

A. An Overview of the Scheme



(a) DIN framework. (b) DIN flow.

Figure 11. An overview of DIN.

DIN is designed for a highly dense PCM chip in which WD is non-negligible. The baseline chip with no WD sets cell-to-cell distance as 60nm at 20nm technology. In contrast, DIN reduces cell-to-cell distance along a word-line from 60nm to 40nm, representing 33% $(= 1 - (40nm \times 60nm)/(60nm \times 60nm)$ cell area reduction. DIN improves density along a word-line while keeping inter-cell space along a bit-line unchanged. Therefore, disturbance errors only appear along word-lines.

DIN is developed based on verify-n-restore (VnR). When writing a PCM line, DIN checks the existence of disturbed cells, and, if necessary, restores original values only to disturbed cells. A write operation tries five times and terminates only if there is no un-correctable disturbance error. Otherwise, a full write covering all cells without Flip-N-Write is performed to guarantee data correctness of a memory line. DIN is built based on an (n,m) encoding mechanism that encodes n-bit data using m-bit code ($m > n > 1$). Since m is bigger than n, there are more bits than needed. DIN chooses those codes with WD-resistant bit patterns. The encoding of DIN cannot completely remove all vulnerable data patterns in a line. Instead, DIN reduces the occurrence frequency of vulnerable data patterns. We will illustrate how to construct (n,m) encoding in following sections.

Figure 11(a) shows the framework of our DIN WD mitigation technique. And Figure 11(b) describes the working flow of DIN. We adopt (3,4) encoding and 64 bytes PCM line in our illustrative example to show how DIN works. We summarize its working flow as follows.

- Given a 64B PCM line, DIN first compresses it using FPC [30], a low-overhead compression scheme that has been adopted in PCM main memory system [5].
- If the compressed line has no more than 369 bits, the compressed line is encoded using (3,4) encoding. The encoded line has no more than 492 bits.
- And then, DIN adds a 3ED2EC (3 errors detectable, 2 errors correctable) BCH code (20-bit BCH code), whose storage overhead is 20 bits, to protect the encoded data.
- If the PCM line cannot be compressed, the line is left un-compressed and un-encoded. DIN adds one bit per line to distinguish whether a line is encoded or not.
- When writing an encoded line to PCM, DIN checks the number of disturbance errors and finishes a write operation if there are no more than 2 disturbance errors, otherwise it continues to restore and verify. At most 2 disturbance errors can be corrected by the 20-bit BCH code. For writing a normal PCM line (i.e., uncompressed and un-encoded), the write terminates only when there is no disturbance error.

B. DIN for SLC PCM

As we discussed in Section II-B, a SLC cell-X may be disturbed by resetting its neighboring SLC cell-Y, only if X is idle and X stores 0. Out of 00, 01, 10, and 11 bit patterns, the only two-bit data that introduces disturbance is 00. Even though not all 00s will result in disturbance errors, by removing the appearance of 00s, we can eliminate most disturbance errors. In particular, if the data to be written does not contain 00, the write is free from WD.

~~000~~ ~~100~~ 00: 101
~~001~~ 101 01: 110
~~010~~ 110 10: 011
011 111 11: 111
2-bit data

~~0000~~ ~~0001~~ ~~1000~~ ~~1001~~ 000: 0101 100: 1011
~~0010~~ ~~0011~~ 1010 1011 001: 0110 101: 1101
~~0100~~ 0101 ~~1100~~ 1101 010: 0111 110: 1110
0110 0111 1110 1111 011: 1010 111: 1111
3-bit data without 00

Figure 12. The (2,3) and (3,4) encoding for SLC PCM.

Therefore, the (n,m) encoding chosen for SLC PCM focuses on eliminating 00s. Figure 12 gives the code books of (2,3) and (3,4) encoding for SLC PCM. Taking (2,3) encoding as an example, DIN removes the appearance of 00s from the eight 3-bit codes, leaving four codes to be used to encode 2-bit data. For example, data "0001" is transformed to "101110" before being saved to PCM. (2,3) encoding requires 64% compression ratio, i.e., a 64B can be encoded and fit in 512-bit line if the line can be compressed to 328 bits or less. (3,4) encoding needs 72% compression ratio.

We next discuss how to choose an appropriate (n,m) for minimizing WD. Given a fixed `m` value, the value of `n`

is often chosen to maximize compression ratio. We first eliminate all m-bit codes that contain 00, and then determine n that can be represented by the left codes. For example, we have eight codes left after removing 4-bit codes that contain 00. The value of n is chosen to be 3 as eight codes are enough to represent 3-bit data. To adopt (3,4) or (2,4) encoding in DIN, a line needs to be compressed to less than 369 or 246 bits, respectively. Clearly, (3,4) is preferred as more PCM lines can be encoded.

To determine the value of m, we try to find a better tradeoff between the benefits in minimizing disturbance and the storage/latency overheads. A small m cannot completely eliminate the appearance of 00s after encoding. For example, data "0110" would be encoded to "110011" using (2,3) encoding. There is still one occurrence of 00 in the encoded data, which may introduce disturbance errors at runtime. Clearly, choosing (...,256) encoding results in at most one occurrence of 00 as a 64B or 512-bit line needs two 256-bit codes. Choosing (...,512) encoding can completely eliminate 00s. However, the code book used for (n,m) encoding is stored in an onchip table. Choosing a large m introduces not only a large onchip storage but also long decoding and encoding latencies.

C. DIN for MLC PCM

Due to its large resistance difference between fully SET and fully RESET states, a MLC PCM cell can exploit partially crystalline states to store more than one bits. Compared to SLC, MLC PCM uses narrower resistance ranges to represent information. To increase programming accuracy, MLC PCM widely adopts iteration-based programming strategies such as single-RESET-multiple-SET [15] (SRMS) and single-SET-multiple-RESET [16] (SSMR). SRMS employs a RESET pulse to initialize each cell to a similar fully amorphous state, and then issues a sequence of SET pulses to crystallize each cell to its target resistance range. A verify operation is performed after each SET to check whether the write operation can be terminated or not. Similarly, SSMR employs a SET pulse and a sequence of RESET pulses. After each RESET, SSMR also needs to verify to determine whether the write can finish.

In this paper, we assume 2-bit MLC uses gray code, and the four resistance ranges from the largest to the smallest represent cell values 00, 01, 11 and 10. Similar to that in SLC, only the RESET operation produces heat that may disturb neighboring cells. Since RESET operation is applied diversely in SSMR and SRMS schemes, DIN adopts different encoding schemes for SSMR and SRMS.

1) (n,m) encoding for SSMR based programming: In SSMR MLC programming, it requires one SET to initialize the cell and different numbers of RESET iterations for writing different cell values. Writing 10 can finish after the initial SET iteration while writing 00 still needs another full RESET and then finishes. Writing 01 and 11 requires several low-magnitude RESET pulses: the RESET voltages used in writing 01 and 11 cell values are 80% and 70%, respectively, of the fully RESET voltage [16]. The heat disseminated to neighboring cells reduces quadruply with decreasing voltages [17], resulting in a very low disturbance effect on neighboring cells: the disturbance error rate due to writing 01 and 11 cell values is smaller than 10^{-12}; while, based on our model in Section III-B3, the disturbance error rates for neighboring cells with data 00, 01 and 11 due to writing 00 are 12.3%, 15.2% and 27.6%, respectively.

The disturbance starts to happen during the RESET of 00 after the initialized SET iteration. The (n,m) encoding chosen for SSMR minimizes the presence frequency of 00. The code book of (3,4) encoding in Figure 12 can be adopted to mitigate write disturbance on SSMR MLC as well. However, our encoding cannot guarantee to completely remove all 00s, if the compression ratio is not small enough. For the rest WD errors, a simple mitigation scheme is adopted by performing verify-n-restore after the RESET of 00, i.e., it performs verify operation after the RESET of 00 to check whether there are disturbed cells, and restore original values if necessary. This correcting verify operation can be merged into the normal verify iteration after each RESET iteration. Since our encoding significantly reduces the presences of 00 and RESET operation has a smaller latency, the negative impact of extra verify and restore (RESET) iterations on write latency and system performance is insignificant, which can be proved by our experiments. Theoretically, restoring values to disturbed cells may further disturb their neighboring cells and run into a loop. As we show in the experiments, cascade write seldom happens. In addition, we disable encoding if cascade write does happen. For rarely appearing disturbed errors due to writing 01 or 11, they can be corrected by either 20-bit BCH code or additional verify-n-restore iterations.

0000 ~~0001~~ | 1000 ~~1001~~ | 000: 1110 | 100: 1000
0010 0011 | 1010 1011 | 001: 1100 | 101: 0011
~~0100~~ ~~0101~~ | 1100 ~~1101~~ | 010: 1011 | 110: 0010
~~0110~~ ~~0111~~ | 1110 ~~1111~~ | 011: 1010 | 111: 0000
3-bit data without 01

Figure 13. The (3,4) encoding for SRMS-based MLC.

2) (n,m) encoding for SRMS based programming: For SRMS MLC programming, RESET is performed to initialize all changed cells in the first iteration. Therefore, disturbance errors may occur only in this iteration. But all four data values (00, 01, 11, 10) need RESET, (n,m) encoding reducing the number of one value appearance but adding that of another value cannot reduce the WD vulnerable data patterns. All WD errors have to be corrected by verify-n-restore. Our experiments show that, in most cases, we have to perform RESET iteration 3-5 times to remove all WD errors and then continue SET iterations, resulting in noticeable performance degradation. To compensate performance loss in verify-n-restore, we choose (n,m) encoding

as follows. When constructing the code book, we eliminate m-bit codes containing 01 (while SLC and SSMR remove 00). Studies [15], [5] have shown that programming different values in 2-bit MLC cell by SRMS requires different average numbers of SET iterations: 00, 01, 10 and 11 require 0, 7, 1 and 5 SET iterations [5]. When there is no write disturbance, writing 00 only needs one RESET iteration while writing 01 needs eight iterations (1 RESET and 7 SETs) on average. By eliminating 01 from the code book, we eliminate the need to write 01 values. The code book of (3,4) encoding for SRMS MLC is shown in Figure 13.

As shown in Figure 14, verify-n-restore may prolong write operation if a cell containing 01 is disturbed. Restoring such a cell finishes in 9 iterations, i.e., one extra RESET after the first RESET. With (n,m) encoding, DIN does not write 01 such that the restore operation can finish in a similar number of iterations as that in baseline. Restore operations are performed on disturbed cells and thus can be conducted in parallel with SET operations on other (non-idle) cells.

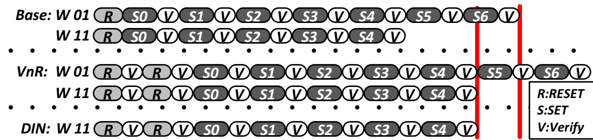


Figure 14. Removing 01 accelerates writes in SRMS MLC PCM.

When choosing (n,m) for SRMS 2-bit MLC, we always choose m to be an even value, e.g., (3,4). Since an encoded line is a concatenation of 4-bit code and each code is stored in two 2-bit cells, we can guarantee that no cell will be written as 01 in the encoded line. In general, when choosing (n,m) encoding for k-bit MLC, we always ensure m can be divided by k.

D. Alternative (n,m) Encoding

Until now, DIN takes a conservative approach in choosing (n,m) encoding. For example, it eliminates all m-bit codes that contain bit pattern 00 for SLC. While DIN minimizes the presence frequency of 00 in the encoded line, it often demands for a small compression ratio. For example, (3,4) encoding requires a compression ratio smaller than 72%, i.e., a line can be compressed to at most 72% of its original size. When m=5, there are 2^5=32 5-bit codes. If we eliminate all codes that contain 00, we have 12 codes left indicating we can choose (3,5) at best. (3,5) encoding requires the compression ratio to be 58% or smaller. In fact, we can still choose (4,5) encoding by allowing some 5-bit codes containing 00s. While it may introduces 00s in the encoded line, (4,5) encoding requires 77% compression ratio, allowing more lines to be compressed and encoded in DIN.

E. BCH Encoding

Upon our (n,m) encoding, BCH code is adopted to correct the remaining no more than two disturbance errors. There are two kinds of encoding methods for BCH code: *systematic* encoding and *non-systematic* encoding [31]. In systematic encoding, the original data is embedded in the output encoded codeword. In non-systematic encoding, parity bits are inter-mixed with the original data forming a valid encoded codeword. The effectiveness of each type of encoding method is the same. In this paper, to avoid destroying our (n,m) encoding, we select *systematic* encoding method [32] to produce BCH code. The encoding method in [32] has been proved to have better performance than traditional systematic and non-systematic methods.

F. Implementation Issues

1) FPC compression: The detailed hardware, energy and latency overhead of compression engine can be found in [30]. The compression and decompression latencies for 64B are 12 FO4 and 60 FO4 [30] (1 FO4 is less than $10ps$, at 22nm). At $22nm$, the hardware overhead of a 64B FPC en/decoder is $25800\mu m^2$ [5], a similar area of 16K PCM cells. For a 64B line, the en/decoding energy is $2.1pJ/1.2pJ$ [5] (less than 0.2% of one cell RESET energy).

2) DIN overhead: The DIN encoder and decoder simply lookup a 2^n-entry table. For (3,4) encoding, DIN encoder includes a 3-to-8 decoder and a CAM with 8 4-bit entries, while the decoder has a 8-to-3 encoder and 4B SRAM. For a 64B line, 128 copies of DIN encoder and decoder are needed. The area overhead is $3100\mu m^2$ at $22nm$. The encoding and decoding latencies are less than 12 FO4. And the energy consumption in encoding and decoding is about $1.5pJ$. DIN also needs to add 63 (63/4KB/8=0.19%) cells along a 4KB row to separate each 64B section, so that WD in a 64B line cannot affect other lines in the same row.

3) BCH overhead: The detailed tradeoff between hardware, energy and latency overheads of BCH ECC circuit can be found in [22]. We adopt the BCH code, which can correct 2 errors within 64B data field, in our DIN. To correct 2 errors in 64B line, the storage overhead is 20 bits [21]. We implemented the BCH encoding method in [32] and decoding method in [22]. The circuits cost $0.03mm^2$ at $22nm$. The BCH circuits spend $900ps$ and $0.4nJ$ in correcting 2 errors. The encoding latency is $1ns$. Checking whether there is any error and correcting 1 error in 64B line cost $700ps$ and $210.2pJ$.

4) Impact on PCM write latency and power allocation: Without WD errors, SLC write latency is deterministic while MLC latency is not. Since DIN encounters erratic WD errors, VnR makes DIN have non-deterministic write latency even for SLC. DIN integrates well with MLC write schemes as non-determinism has already been covered. To address latency non-determinism on SLC, DIN needs to adopt a non-deterministic write logic in MLC chips [33], [34].

By ensuring no more than 50% cells in a line are written, Flip-N-Write guarantees write power is no more than half of the full write power and thus supports more concurrent

writes. However, the VnR used in DIN may require full-line write and demand more write power. To simplify power allocation, full-line writes need to be rescheduled when there is enough power. In our experiments, we observed few full-line write happens. The performance impact is insignificant.

V. Related Work

Phase change memory has emerged as a promising main memory technology. Recent studies have investigated many reliability issues in PCM. PCM chip lifetime is improved by wear leveling [35], differential-write [3] and Flip-N-Write [4]. Resistance drift triggers errors in MLC PCM, which can be mitigated by scrubbing based technique [19]. Recent work utilized three out of four states of 2-bit MLC to tolerate resistance drift [20]. By removing 01 for SRMS 2-bit MLC, DIN shares the similarity with this work. The difference is that (n,m) encoding presents a generalized encoding framework and DIN is the first architectural work to mitigate write disturbance in PCM.

VI. Experimental Methodology

A. Baseline configuration

To evaluate the effectiveness of DIN, we adopted the same baseline configuration as [34] and compared our technique to existing solutions. We simulated an 8-core in-order CPU system with PCM based main memory using macsim [36], which models not only the entire memory hierarchy including L1, L2, DRAM last-level caches, the memory controller and PCM based main memory, but also timing constraints including cache-memory related bus contention, memory bank conflicts and DDR scheduling constraints.

The details of our baseline parameters can be found in Table I. Each core in the baseline has a 32MB private write-back DRAM last level cache to alleviate heavy write traffic in PCM main memory. The entire memory hierarchy shares the same 64B line size. The memory controller schedules reads with higher priority. Writes are scheduled only when there is no read. When the write queue is full, a write burst is triggered, which blocks any pending read until the write queue is empty again [34].

Table I
Baseline Configuration

CPU	8-core, 4GHz, single issue, in-order
L1	private, I/D separate, 32KB per core, 64B line
L2	private, 2MB per core, 4-way LRU, 64B line, write back
L3	DRAM offchip, private, 32MB per core, 8-way LRU, 64B line, write back, 50ns (200-cycle) hit
Mem. Ctrl	onchip, 24-entry R/W queues
PCM Main Memory	8GB, two channels, 1 rank per channel, 8 banks per rank, 64B line, SLC: read 100ns, RESET 100ns, SET 150ns; 2-bit MLC: SSMR write, RESET iteration 200ns, critical word read 100ns, 20nm PCM chip

We modeled a main memory system with two 4GB PCM channels. One 4GB channel contains 8 banks. Each memory line has 64B. We adopted SLC PCM SET and RESET latencies from [26] and 2-bit MLC write model from [34]. Our main memory also obeys the power constraints in [34]. The FPC, DIN and BCH energy and latency in Section IV-F were also modeled in our simulator.

B. Simulated workloads

We chose a subset of benchmarks from SPEC2006, BioBench, STREAM and PARSEC to construct our simulated workloads and studied write disturbance on different benchmarks. The applications from SPEC2006, BioBench and STREAM can be combined into multi-programmed workloads. Each core runs one copy of these applications. The multi-thread benchmarks in PARSEC with native inputs are divided into 8 threads, each of which runs on one core. We adopted performance metric *speedup* as $Speedup = IPC_{tech}/IPC_{base}$ from [5], where IPC_{tech} and IPC_{base} are the IPCs of the setting with scheme tech and the baseline setting, respectively.

VII. Evaluation

We implemented and compared the following schemes in Table II. All schemes are evaluated with density enhanced PCM chips. In these highly dense PCM chips, the inter-cell thermal band along a word-line is removed, while the inter-cell space along a bit-line keeps unchanged.

Table II
Compared Schemes

BASE	Verify-n-Restore(VnR) to correct WD errors.
INV	The same as BASE except inverting 1/0 if a line has more 0s.
COM	The same as BASE except memory line is compressed.
CINV	The same as INV except memory line is compressed.
BCH	Correcting all errors with de/encoding latency overhead.
DIN	(3,4) encoding, 20-bit BCH correcting 2 errors and VnR.
IDL	The same as BASE except there is no VnR latency.

A. PCM Chip Area

We modeled chip area according to the latest $20nm$ PCM chip [26]. In this prototype chip, PCM cells occupy 46.6% of the total chip area. DIN reduces the inter-cell distance along a word-line from $60nm$ to $40nm$, representing 33% memory capacity improvement. In other words, the total chip area is reduced by 15.4% for the chips of the same capacity.

B. Write Disturbance Errors along word-lines

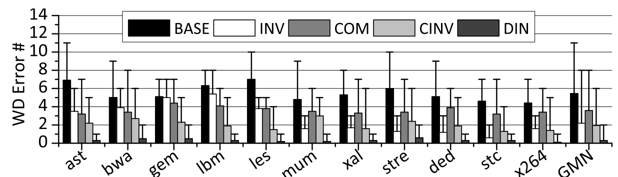


Figure 15. The WD errors along word-lines in SLC PCM.

Figure 15 summarizes the number of write disturbance (WD) errors along word-lines in a SLC PCM memory system. The solid bar indicates the average WD error (WDE) number, while the error bar represents the maximum WDE number in one memory line. BASE has 5.4 WDEs on average and 11 WDEs at most in every PCM write. INV

reduces the WDE number for integer benchmarks, such as $astar$ and $mummer$, by more than 50% over BASE. However, for floating point applications, such as $bwaves$, $gemsFDTD$, and lbm, INV only slightly reduces WDEs. This is because floating point applications have comparable 0s and 1s in most lines. COM shows that WDE number is smaller for the benchmarks that can be compressed. Since a line has fewer 0s after compression, CINV cannot achieve a significant WDE reduction. DIN has 0.3 WDEs on average and 2 WDEs at most, showing significant improvement over existing schemes.

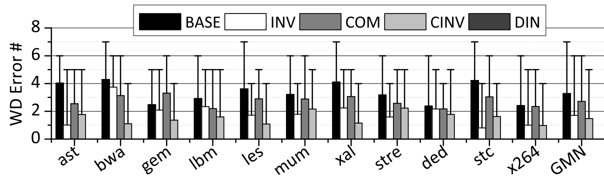


Figure 16. The WD errors along word-lines in SSMR MLC.

Figure 16 shows the WDE number along a word-line in a SSMR (Single-SET-Multiple-RESET) based MLC PCM memory system. From the figure, BASE has 3.27 WDEs on average and 7 WDEs at most per PCM line write. In SSMR, only the fully RESET operation may disturb other idle neighboring cells along word-lines. INV reduces WDEs slightly for floating point applications as they have comparable 1s and 0s. COM is higher than INV showing that the compressed data has more 00s than inverted data does. Since 00s are more than 11s in compressed data, CINV further reduces WDE number. For 2-bit MLC, DIN completely eliminates 4-bit code like '00XX' and 'XX00' (X can 0 or 1), i.e., no cell will be fully RESET. Therefore, there is no WD for SSMR 2-bit MLC. In the figure, DIN related numbers are 0.

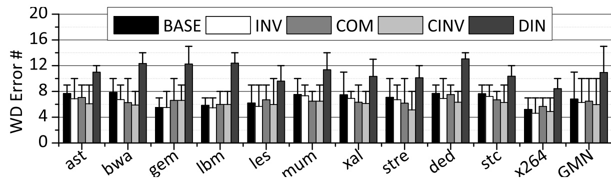


Figure 17. The WD errors along word-lines in SRMS MLC.

Figure 17 summarizes the WDE number along a word-line in a SRMS (Single-RESET-Multiple-SET) MLC PCM memory system. The average WDE number increases by 108% over SSMR. This is because all changed cells are initialized by RESET, which increases vulnerable cells accordingly. The maximum WDE number also increases. INV, COM and CINV cannot significantly reduce the average or the maximum WDE number. DIN is even worse than BASE because DIN needs to write more bits after encoding and the encoded line contains more 11s (as it has no 01s) having a larger WD error rate. The design goal of DIN for SRMS MLC is to hide restore latency within normal write iterations. An increase of WDE number is expected.

C. Write Disturbance Errors along bit-lines

Figure 18 summarizes the WDEs along a bit-line for SLC and two MLC cases. In this experiment, we further eliminate inter-cell thermal band along a bit-line, which leads to disturbance errors in neighboring lines, i.e., the line on the left/right side of the written line. We reported both the average (the solid bar) WDE number and the maximum (the error bar) WDE number due to writing one PCM line. From the figure, the average and the maximum WDE numbers for BASE are 19.2 and 30, respectively, for SLC. Similar to the results along word-lines, INV can effectively mitigate disturbance errors on integer benchmarks but not floating point benchmarks. DIN still achieves the best reduction: the average and the maximum are 6.5 and 10 respectively. There are similar trends for two MLC cases.

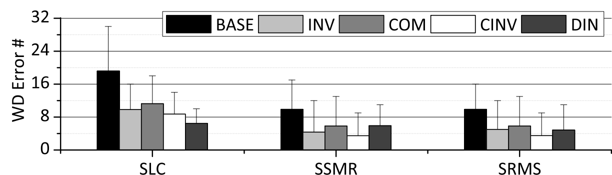
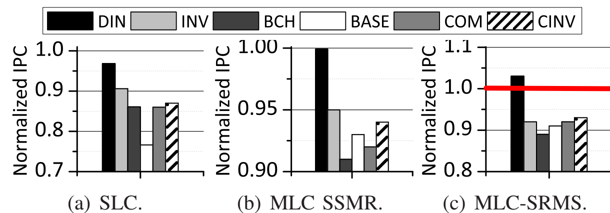


Figure 18. WD errors along bit-lines (no inter-cell thermal band).

Even though DIN greatly reduces the WDE numbers, errors are still significant, so either strong ECC or long latency verify-n-restore (VnR) is required. In particular, since these errors appear in different lines, it requires additional address decodings, and VnR cannot be combined with the line write operation. The additional VnR may trigger WD errors in two its neighboring lines in a cascade style, resulting in significant performance degradation. Therefore, we do not remove the thermal band along a bit-line.

D. System Performance



(a) SLC. (b) MLC SSMR. (c) MLC-SRMS.

Figure 19. IPC of SLC, SRMS and SSMR (normalized to IDL).

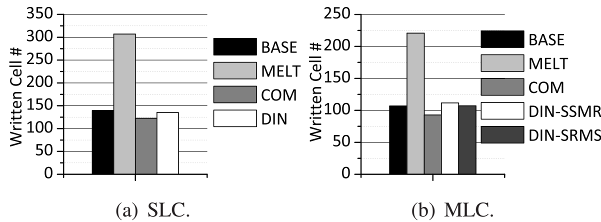
Figure 19(a) compares system performance using different mitigation schemes on SLC PCM. All results are normalized to IDL. We observed that after adopting (n,m) encoding, there are still 1 or 2 disturbance errors. While the 20-bit BCH embedded in the encoded line can correct up to two errors, DIN is built based on VnR, which requires a verify sequence to determine if the write operation can terminate. This verify operation can be naturally hidden in MLC programming strategies, but shows 4% performance degradation on SLC PCM. INV, COM and CINV all achieve better performance over heavyweight BCH. Although INV, COM and CINV depending on VnR suffer from large write latency, large

decoding latency on the critical path makes heavyweight BCH have worse performance. Compared to IDL, BCH exhibits 14% performance degradation. BASE with VnR has 24% performance degradation, the largest performance degradation of all schemes. The is because nearly all writes have extra verifies and restores.

Figure 19(b) and 19(c) compare performance for SSMR and SRMS MLC PCMs, respectively. We observed that DIN only has 0.1% performance degradation when using SSMR and even obtains 3% performance improvement over IDL when using SRMS. After removing codes containing 01 for SRMS MLC, DIN does not need to write cells containing 01. Since writing 01 is usually slower than writing other cell values, removing 01s accelerates write operations. After removing codes containing 00s for SSMR MLC, DIN writes are disturbance free, and thus its performance is almost the same as IDL. The slight performance degradation is caused by BCH error checking latency.

E. Number of Changed Cell and Write power

PCM write power depends on the changed cell number in a write. Figure 20(a) compares cell changes per write in SLC PCM. BASE on average writes 140 cells (27.3%) per 64B line. MELT writes 304 cells, an increase over 115%. DIN slightly reduces the changed cell number by 3% over BASE in SLC PCM. Figure 20(b) reports the changed cell number per write in MLC PCM. DIN increases the changed cell number by 4.4% with SSMR and by 0.1% with SRMS.



(a) SLC. (b) MLC.

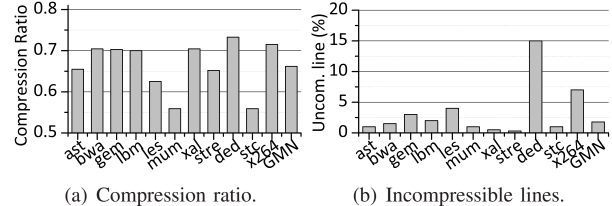
Figure 20. The number of cells written per write request

F. System Energy

Previous study [37] has shown that the DRAM cache and CPU consume 75% of total system power with a SLC PCM main memory system. Based on the power and energy model in [3], we calculated system energy consumption. Since MLC adopts iteration-based programming strategies, MLC PCM consumes more energy in writes than SLC does. Since SET iteration consumes more energy than RESET, SRMS MLC consumes more energy than SSMR MLC. The write energy percentages in the system energy of SLC, MLC SRMS and SSMR are 3.1%, 10.4% and 7.2%, respectively. For different PCM types, DIN has 1%-3% write energy increase over IDL (ideal) due to extra verify operations. For all settings, the CPU/DRAM cache energy consumption dominates the overall system energy consumption. If DIN improves/degrades performance, the system energy decreases/increases accordingly. For SLC, DIN slightly increases system energy over IDL by 2.9%. For SRMS MLC, DIN reduces system energy of IDL by 2.1%. For SSMR MLC, DIN has about the same performance as IDL, so the system energy increases by 1.9%. However, compared to BASE, thanks to better performance, DIN improves system energy by 9.7%, 4% and 9.9% on SLC, SSMR and SRMS MLC PCMs, respectively.

G. Compression Effectiveness

To enable (3,4) encoding in DIN, a memory line needs to be compressed to 72% of orginal size or less. We next study the effectiveness of compression in the paper. Figure 21(a) reports the average compression ratio of different applications. The average compression ratio varies from 65% to 73%, indicating (3,4) encoding is sufficient for these applications. Figure 21(b) reports the percentage of lines that are not compressible under (3,4) encoding. As we can see from the figure, most applications have less than 5% lines incompressible. Only ded and $x264$ have large percentages. ded is a compression application and $x264$ is a media processing benchmark. The output data of these two benchmarks are not highly compressible.



(a) Compression ratio. (b) Incompressible lines.

Figure 21. The compression ratio and incompressible lines.

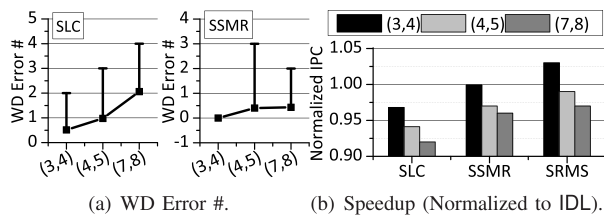
In the worst case, if a benchmark is fully incompressible, i.e., nearly all its memory lines are not compressible, DIN degenerates to BASE (VnR). We expect to see as high as 24% performance degradation, the same as BASE.

H. Design Space Exploration

As discussed in Section VII-B, while (3,4) encoding can eliminate all codes containing 00s for SLC, neither (4,5) encoding nor (7,8) encoding can. As a result, the code books of (4,5) and (7,8) encoding may still contain WD vulnerable data patterns such as 00s for SLC and SSMR MLC or 01s for SRMS MLC.

From the Figure 22(a), (4,5) and (7,8) encodings introduce more disturbance errors. However, the numbers are still much smaller than those in BASE and those achieved by other techniques, showing they are still effective. We also observed that SSMR MLC programming is less sensitive to encoding variations. Even with (7,8), the encoding for SSMR programming still has a lot of chances to eliminate 00s. The (4,5) encoding has the largest WD error number. Since the last bit in 5 bits has to be stored with another bit from other encoded word, it is easy for some lines to form 00 in these unaligned 2-bit MLC cells. As shown in Figure 22(b), (3,4) gives the best performance as it

eliminates more WD vulnerable data patterns than the other two encoding schemes.



(a) WD Error #. (b) Speedup (Normalized to IDL).

Figure 22. Studying relaxed (n,m) encoding.

However, different (n,m) encodings require different compression ratios. For example, (3,4), (4,5), and (7,8) need 72%, 77%, and 84% compression ratios respectively. In general, the compression ratio required by (n,m) encoding can be calculated as (512-20)n/(512m). Here we assume DIN uses a 20-bit BCH per line to cover two disturbance errors that cannot be mitigated by encoding. If a benchmark only has 80% compression ratio, (3,4) encoding cannot be adopted. Instead, it is possible to employ (7,8) encoding to reduce disturbance errors.

VIII. Conclusion

With fast technology scaling, write disturbance has become a major scaling challenge for PCM. In this paper, we proposed DIN, an (n,m) encoding scheme to eliminate bit combinations leading to write disturbance vulnerable cells. We adopted lightweight data compression and BCH code to enable (n,m) encoding with low overheads. Our experimental results show that DIN gains write disturbance resistance in high density PCM chips while achieving comparable performance for a wide range of applications.

References

[1] "The international technology roadmap for semiconductors report," 2009. http://www.itrs.net/

[2] "Micron 16nm mlc nand," 2013
. http://investors.micron.com/releasedetail.cfm?ReleaseID=777402

[3] P. Zhou *et al.*, "A durable and energy efficient main memory using phase change memory technology," in *ISCA*, 2009.

[4] S. Cho *et al.*, "Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance," in *MICRO*, 2009.

[5] L. Jiang *et al.*, "Improving write operations in mlc phase change memory," in *HPCA*, 2012.

[6] S. Raoux *et al.*, "Phase-change random access memory: A scalable technology," *IBM J. RES. & DEV.*, 2008.

[7] S. Lee *et al.*, "Programming disturbance and cell scaling in pcm: for up to 16nm based $4f^2$ cell," in *VLSIT*, 2010.

[8] B. Kim *et al.*, "Current status and future prospect of phase change memory," in *ASICON*, 2011.

[9] S. J. Ahn *et al.*, "Reliability perspectives for high density pram manufacturing," in *IEDM*, 2011.

[10] D. G. Wilson *et al.*, "Single transistor memory with immunity to write disturb," *US PATENT*, vol. 8148759, 2012.

[11] K.-T. Park *et al.*, "A zeroing cell-to-cell interference page architecture with temporary lsb storing and parallel msb program scheme for mlc nand flash memories," *JSSC*, 2008.

[12] G. Dong *et al.*, "Using data postcompensation and predistortion to tolerate cell-to-cell interference in mlc nand flash memory," *TCS*, vol. 57, no. 10, 2010.

[13] U. Russo *et al.*, "Intrinsic data retention in nanoscaled phase-change memories - part i: Monte carlo model for crystallization and percolation," *TED*, vol. 53, no. 12, 2006.

[14] A. Redaelli *et al.*, "Intrinsic data retention in nanoscaled phase-change memories - part ii: Statistical analysis and prediction of failure time," *TED*, vol. 53, no. 12, 2006.

[15] A. Cabrini *et al.*, "Voltage-driven multilevel programming in phase change memories," in *IWMTDT*, 2009.

[16] S. Braga *et al.*, "Voltage-driven partial-reset multilevel programming in phase-change memories," in *ESSDERC*, 2010.

[17] U. Russo *et al.*, "Modeling of programming and read performance in phase-change memories - part i: Cell optimization and scaling," *TED*, vol. 55, no. 2, 2008.

[18] U. Russo *et al.*, "Modeling of programming and read performance in phase-change memories - part ii: Program disturb and mixed-scaling approach," *TED*, vol. 55, no. 2, 2008.

[19] M. Awasthi *et al.*, "Efficient scrub mechanisms for error-prone emerging memories," in *HPCA*, 2012.

[20] N. H. Seong *et al.*, "Tri-level-cell pcm: toward an efficient and reliable memory system," in *ISCA*, 2013.

[21] C. Wilkerson *et al.*, "Reducing cache power with low-cost, multi-bit error-correcting codes," in *ISCA*, 2010.

[22] D. Strukov, "The area and latency tradeoffs of binary bit-parallel bch decoders for prospective nanoelectronic memories," in *ACSSC*, 2006.

[23] K. Lee *et al.*, "A 90 nm 1.8 v 512 mb diode-switch pram with 266 mb/s read throughput," *JSSC*, 2008.

[24] H. Chung *et al.*, "A 58nm 1.8 v 1gb pram with 6.4 mb/s program bw," in *ISSCC*, 2011.

[25] K. Lu *et al.*, "Optimized scaling of diode array design for 32nm node phase change memory," in *VLSI-TSA*, 2008.

[26] Y. Choi *et al.*, "A 20nm 1.8v 8gb pram with 40mb/s program bandwidth," in *ISSCC*, 2012.

[27] C. Bergonzoni *et al.*, "Reset current scaling in phase-change memory cells: Modeling and experiments," *TED*, 2012.

[28] M. Kang *et al.*, "Pram cell technology and characterization in 20nm node size," in *IEDM*, 2011.

[29] B. Gleixner *et al.*, "Data retention characterization of phase-change memory arrays," in *IRPS*, 2007.

[30] A. R. Alameldeen *et al.*, "Frequent pattern compression: A significance-based compression scheme for l2 caches," *Univ. Wisconsin-Madison, Tech. Rep*, 2004.

[31] S. Lin *et al.*, *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 1983.

[32] R. El-Din *et al.*, "A novel high-speed systematic encoder for long binary cyclic codes," *IEEE Communications Letters*, vol. 17, no. 5, 2013.

[33] K. Fang *et al.*, "Memory architecture for integrating emerging memory technologies," in *PACT*, 2011.

[34] L. Jiang *et al.*, "Fpb: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory," in *MICRO*, 2012.

[35] M. K. Qureshi *et al.*, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *MICRO*, 2009.

[36] "macsim." https://code.google.com/p/macsim/

[37] M. K. Qureshi *et al.*, "Preset: Improving read write performance of phase change memories by exploiting asymmetry in write times," in *ISCA*, 2012.