ADAM: Architecture for Write DisturbAnce Mitigation in Scaled Phase Change Memory

Shivam Swami and Kartik Mohanram
Department of Electrical and Computer Engineering, University of Pittsburgh, PA
shs173@pitt.edu kartik.mohanram@gmail.com

Abstract

With technology scaling, phase change memory (PCM) has become highly vulnerable to write disturbance (WD) errors. A PCM WD error occurs when a cell write dissipates heat to idle cells in the same/adjacent word lines (WLs), disturbing the states of those cells. Whereas state-of-the-art solutions, e.g., data insulation (DIN) and super dense PCM (SD-PCM), have successfully addressed WL PCM WD errors, reducing (i) bit line (BL) WD errors and (ii) the performance penalties of aggregate (WL+BL) WD error recovery remain areas of active research and development.

***Architecture for Write DisturbAnce Mitigation, ADAM,** is a low cost, high performance pattern-based data compression and alignment solution to reduce the aggregate (WL+BL) WD error rate in PCM. At no impact to inter-cell spacing, ADAM increases the lateral separation between the cells storing useful data in adjacent WLs, ensuring that the heat dissipated to adjacent WLs minimally impacts the cells storing useful data. For one compression tag bit per 512-bit cache line, ADAM provides an effective solution to reduce the number of WL and BL cells vulnerable to WD errors. ADAM also integrates a novel Deferred WD Correction scheme, DEFT, that opportunistically defers latency-intensive WD error recovery of cached data in the adjacent WLs without impacting memory reliability. ADAM is evaluated on single-/multi-level cell (SLC/MLC) PCM using the SPEC CPU2006 benchmarks. Results for SLC (MLC) PCM show that in comparison to state-of-the-art SD-PCM, ADAM reduces the aggregate WD error rate by 32% (60%); this translates to a 50% (61%) reduction in error correction energy and a 7% (15%) improvement in system performance.*

1. Introduction

Whereas phase change memory (PCM) [1, 2] has emerged as a scalable, dense, and low power replacement for DRAM in main memories, recent research [3–5] has shown that PCM is highly vulnerable to write disturbance (WD) errors below the 20nm technology node. A PCM WD error occurs when programming a cell in the reset state dissipates heat to the idle cells within the same/adjacent word lines (WLs), disturbing the states of those cells. The problem of PCM WD is exacerbated by technology scaling, i.e., reducing feature size and increasing data density (i.e, multi-/triple-level cell (MLC/TLC) technology), and has emerged as a first-order obstacle to PCM commercialization [3–5].

State-of-the-art WD mitigation techniques like data insulation (DIN) [3] and super-dense PCM (SD-PCM) [4] reduce WL WD errors by encoding the data to a less WD-vulnerable pattern before it is written to the memory. To mitigate BL WD errors, which are more severe than WL WD errors [3–5], these schemes insert thermal guard bands between the WLs [3], or employ idle error-correcting pointers (ECPs) in the adjacent WLs for BL WD error recovery [4]. Whereas thermal guard bands increase PCM cell area by 100%, the effectiveness of ECP-based BL WD error recovery decreases with time, since the availability of idle ECPs reduces as the memory ages. Furthermore, both DIN [3] and SD-PCM [4] rely on verify-and-restore (VnR) [3] for addressing WD errors beyond their error correction capabilities. However, VnR increases memory wear and degrades system performance in practice (discussed in section 2.2). In summary, whereas state-of-the-art solutions have made significant inroads in addressing WL PCM WD errors, reducing BL PCM WD errors and reducing the high performance penalty of VnR for aggregate (WL+BL) WD error recovery remain areas of active research and development.

Architecture for Write DisturbAnce Mitigation (ADAM) is a low overhead, high performance WD mitigation solution for PCM-based main memories. ADAM makes two core contributions. First, ADAM proposes a *pattern-based data compression and alignment* solution to reduce the aggregate (WL+BL) WD error rate. ADAM performs pattern-based data compression to reduce the write-back width, which reduces the number of aggressor as well as victim cells when the write-back data is stored in the active WL. Thereafter, ADAM reorganizes the compressed data such that consecutive WLs in memory store the compressed data in alternate alignments, i.e., compressed data in even-numbered WLs is right-aligned and compressed data in odd-numbered WLs is left-aligned; uncompressed data is stored as-is. At no impact to inter-cell spacing or system endianness, this logical alignment of data increases the lateral separation between the PCM cells that store useful compressed data in the adjacent WLs. As a result, when data is written in a WL, the heat dissipated to the adjacent WLs minimally impacts the cells storing useful data. Thus, at the cost of one compression tag bit per 512-bit cache line, ADAM is an effective solution to reduce the number of WL and BL cells vulnerable to WD errors.

Second, *Deferred WD Correction, DEFT,* opportunistically reduces the overhead of WD error recovery of cached data in the adjacent WLs. Whereas both DIN [3] and SD-PCM [4] invariably employ VnR-based error recovery for WD errors in the adjacent WLs, DEFT defers WD error recovery in the adjacent WLs if the adjacent WL data is also present in the last-level cache (LLC). In such scenarios, DEFT only sets the dirty bit corresponding to that data in the LLC, and WD error recovery occurs when the dirty data is evicted from the LLC and written back to the memory. Hence, for no additional overhead, DEFT improves system performance without impacting memory reliability.

We evaluate ADAM on single-/multi-level cell (SLC/MLC) PCM using (i) the NVMain memory simulator [6] and (ii) the MARSS full-system simulator [7] on the SPEC CPU2006 benchmarks [8]. ADAM is compared to state-of-the-art SD-PCM [4], which employs DIN and idle ECPs for WL and BL WD mitigation, respectively. Results for SLC (MLC) PCM show that in comparison to SD-PCM, ADAM reduces the aggregate WD error rate by 32% (60%); this translates to a 50% (61%) reduction in error correction energy and a 7% (15%) improvement in system performance.

The rest of the paper is organized as follows. Section 2 introduces the PCM WD problem and motivates ADAM. Section 3 describes the theory and architecture of ADAM. Section 4 details the evaluation methodology, section 5 presents results, and section 6 is a conclusion.

This research was supported by NSF Award CCF-1217738

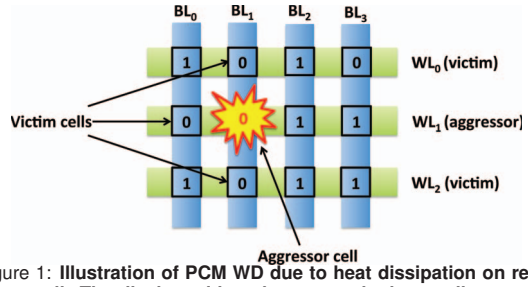

Figure 1: **Illustration of PCM WD due to heat dissipation on resetting a cell. The dissipated heat impacts only those adjacent cells that are idle and already in the reset state.**

2. Background and motivation

This section covers PCM write disturbance (WD), its impact on memory reliability, and WD error recovery to motivate ADAM.

2.1 PCM basics

PCM stores data by altering the resistance of its phase change material (e.g., $Ge_2Sb_2Te_5$ (GST)) between the fully amorphous state with high resistance (reset state) and the fully crystalline state with low resistance (set state) [1]. To reset a PCM cell, a high amplitude reset pulse is applied to melt the GST and then abruptly cut-off to quench the molten GST into the amorphous state. To set a PCM cell, a small amplitude set pulse heats the GST above its crystallization temperature for a sufficiently long duration to transform the GST into the crystalline state. The information stored in a PCM cell is read by measuring the resistance of the cell.

2.2 PCM write disturbance (WD)

A PCM WD error occurs when resetting a cell dissipates heat to the idle reset cells within the same/adjacent word lines (WLs), disturbing the states of those cells [3–5]. Since the heat produced during a PCM reset operation is $> 2\times$ the heat produced during a PCM set operation, WD errors are more prominent while resetting a cell [3–5]. Further, due to a partial decay of the dissipated heat as it reaches the adjacent cells, the temperature rise in the adjacent cells is below the GST melting point, but higher than the GST crystallization temperature. As a result, an adjacent cell in the amorphous state may be partially transformed into the crystalline state, losing its stored value. Hence, WD errors occur *only* when a cell is being reset and its adjacent cells are idle and in the reset state. This is illustrated in figure 1, where resetting a cell in WL_1 dissipates heat to the adjacent cells, making them vulnerable to WD.

As stated in [3, 4], the sneak heat from the reset operation decays faster horizontally along the WL than vertically along the BL. As a result, the temperature rise of the adjacent cells in a WL is lower than the temperature rise of the adjacent cells in a BL. Thus, the BL WD error rate is greater than the WL WD error rate [3, 4]. Figure 2(a) reports the WL and BL WD errors per write for SLC (MLC) PCM, evaluated using NVMain [6] on memory traces from the SPEC CPU2006 [8] benchmark suite. As shown in the figure, the number of BL WD errors for SLC (MLC) PCM is $1.83\times$ ($8\times$) higher than the number of WL WD errors. Hence, in order to mitigate WL and BL WD errors, a 2F×2F (WL×BL) PCM cell for feature size F is provided with extra inter-cell space of F and 2F along the WL and BL, respectively, increasing the cell area by $3\times$ (to 3F×4F) over the ideal 2F×2F PCM cell [3, 4].

Any attempt to reduce PCM cell area by decreasing inter-cell spacing results in a high number of WD errors along both the WL and the BL. However, since a large PCM cell undermines the benefits of technology scaling, architects trade-off PCM cell area with WD errors and employ verify and restore (VnR) [3] to completely eliminate WD errors on a memory write. In VnR, the disturbed cells are identified using extra verify-read operations, and restored to their fault-free values. However, VnR may require multiple iterations before it converges and completely eliminates all WD errors. Due to its iterative nature, VnR increases the effective latency of the write operation and degrades system performance. Additionally, by repeatedly restoring the faulty cells, VnR also increases memory wear, decreasing its lifetime. Figures 2(b), 2(c), and 2(d) report the impact of VnR (in a practical SLC/MLC PCM) on write latency, system performance, and memory lifetime, respectively. The results are normalized to ideal PCM, i.e., PCM without WD errors. We observe that practical SLC (MLC) PCM experiences $5\times$ ($7\times$) increase in write latency, 40% (60%) degradation in system performance, and 10% (20%) reduction in memory lifetime over ideal SLC (MLC) PCM.

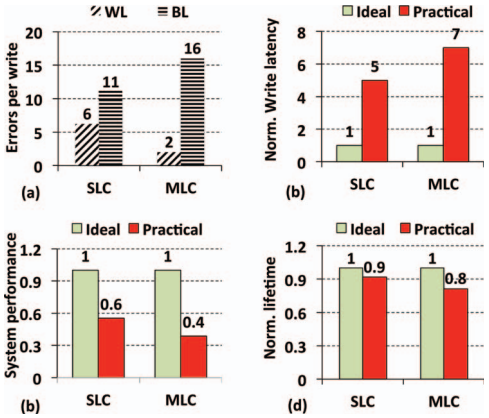

Figure 2: **This figure reports (a) WL and BL WD errors per write for SLC (MLC) PCM and compares ideal PCM (i.e., PCM with no WD) with practical PCM (which employs VnR for WD-free operation) in terms of (b) write latency, (c) system performance, and (d) memory lifetime. These results were obtained using the NVMain [6] memory simulator and the MARSS [7] full-system simulator on SPEC CPU2006 [8] benchmarks. The high overhead of practical PCM is due to VnR, which increases memory wear, write latency, and degrades system performance by repeatedly reading and writing memory cells in order to completely eliminate WD errors.**

2.3 State-of-the-art in PCM WD mitigation

To reduce the impact of VnR in SLC/MLC PCM, DIN [3] targets WD errors in the active WL. DIN employs pattern-based data compression to compress the write-back data and then encodes the compressed data using (3,4) encoding, which removes adjacent zeros from the compressed data in the active WL. Additionally, DIN appends a 20-bit BCH code [9] to the (3,4) encoded data for enhanced protection against WD errors. Finally, to eliminate BL WD errors, DIN allocates thermal bands between the WLs; however this increases the PCM cell area by 100% (from 2F×2F to 2F×4F).

To mitigate WD errors without impacting cell area, SD-PCM [4] proposed the use of idle error-correcting pointers (ECPs) in the adjacent WLs to temporarily recover from BL WD errors in the adjacent WLs, while simultaneously integrating DIN for mitigating WL WD errors. Due to the transient nature of WD errors, the ECP entries can be cleared on the next write. However, since ECPs are required for correcting hard errors, which increase as the memory ages, the availability of idle ECPs reduces with time. As a result, the ability of SD-PCM to mitigate BL WD errors by using idle ECPs also reduces with time. Furthermore, due to increased cell activity of ECPs, SD-PCM increases the vulnerability of ECP cells to WD errors. To mitigate WD errors in ECP cells, SD-PCM uses a low-density ECP chip (i.e., $8F^2$ per ECP cell), increasing the ECP cell array size by $2\times$.

In summary, although solutions exist for PCM WD error mitigation, without exception, these solutions usually impose heavy overheads on area, lifetime, and system performance in practice.

3. ADAM

ADAM integrates two orthogonal approaches: (i) WD mitigation and (ii) Deferred WD Correction (DEFT) to reduce the aggregate (WL+BL) WD error rate and improve system performance, respectively, in the presence of WD errors.

3.1 WD Mitigation

This section develops the theory and architecture of WD mitigation in ADAM based on two observations.

Observation 1: As shown in [3–5], a PCM WD error requires two conditions: (i) an aggressor cell that is to be programmed to the reset state, which may dissipate enough heat to disturb the states of the adjacent cells, and (ii) one or more idle adjacent cells in the amorphous (i.e., reset) state that may lose their stored values. *Therefore, WD errors can be reduced if the number of aggressor cells are reduced and/or if the cells adjacent to the aggressor cell are in the set/active state.*

Inversion (INV) [3] is a direct approach to mitigate WD errors by reducing the number of aggressor cells, i.e., reducing the number of reset operations on a memory write. INV inverts the data on a memory write if the data contains more zeros than ones. However, INV is ineffective when the number of zeros and ones in the data are comparable, e.g., floating point applications [3]. Additionally, for an all-zero cache line, INV increases memory write latency by 1.5× as it chooses to re-write all ones (PCM set latency is 1.5× PCM reset latency [3, 4]). Since real-world applications consist of a large number of all-zero cache lines [10–12], INV degrades system performance by unnecessarily increasing the write latency.

Reducing the number of victim cells by reprogramming WD-vulnerable idle cells on a write-back is another solution to mitigate WD errors. However, forced reprogramming of victim cells (despite no change to their stored values) on a write-back mitigates WD errors within the active WL at the cost of (i) increased write energy, (ii) increased WD errors in the adjacent WLs due to more aggressor cells, and (iii) increased memory wear. Therefore, in order to reduce WL WD errors without impacting write energy, WD errors in the adjacent WLs, and memory wear, ADAM employs pattern-based data compression to reduce the write-back data width, which in turn reduces the number of aggressor as well as victim cells when the write-back data is stored in the active WL.

ADAM uses frequent pattern compression (FPC) [10] to compress the write-back data before it is written to the memory. FPC leverages program data statistics to successfully compress a wide range of data. It was reported in [12] that on average, FPC compresses 60% of the write-back data with the compression ratio (i.e., original data width / compressed data width) ≥ 1.82. In this work, we follow the FPC framework proposed in [3, 13] to compress a 512-bit cache line on a write-back. This FPC framework requires a 1-bit compression tag per cache line (stored with the cache line); the set (reset) state of the compression tag indicates compressed (uncompressed) data. Note that ADAM is agnostic to the choice of compression scheme and can be easily integrated with other cache-level compression schemes such as Base-Delta-Immediate (BΔI) [11] without any modifications to its architecture.

ADAM's WL WD mitigation mechanism is explained using figure 3. Along the lines of [1, 3, 12], we assume that bit-level data comparison write (DCW, i.e., classical read-modify-write) [14] is integrated in the memory system. DCW reduces memory wear by eliminating redundant bit writes to memory. Without loss of generality, let the write-back width (i.e., the last-level cache (LLC) line size) be 16 bits. Note that although a WL stores multiple cache lines, this figure only depicts the region of interest in a WL. Figure 3(a) shows that writing an all-zero uncompressed data to conventional PCM results in 8 aggressor and 8 victim cells. Figure 3(b) shows that ADAM employs FPC-based data compression to compress the all-zero data to 3-bit data, which in turn reduces the number of aggressor and victim cells to 1 each.

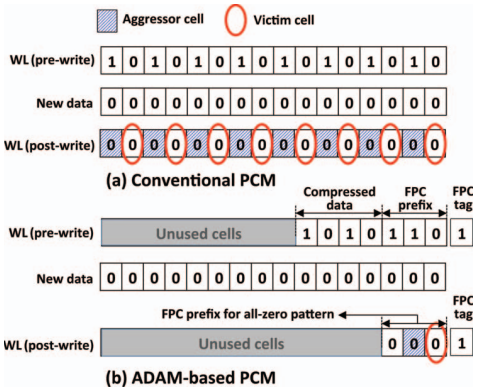

Figure 3: **WL WD errors in (a) conventional PCM and (b) ADAM-based PCM. Note that the existing data in ADAM-based PCM is also shown in FPC-compressed format. ADAM overwrites the existing data (using DCW) with FPC-compressed new data, thereby reducing the number of aggressor (victim) cells from 8 (8) to 1 (1).**

Observation 2: Due to a slow decay of the sneak heat along BLs, the BL WD error rate is higher than the WL WD error rate [3, 4]. As a result, the BL thermal guard bands – required to prevent WD errors – are 2× wider than the WL thermal guard bands (refer Sec. 2.2). *Hence, mitigating BL WD errors (i.e., WD errors that occur in the adjacent WLs) at no cost to cell area and performance is critical for improving memory reliability.*

Based on this observation, ADAM reorganizes compressed data such that consecutive WLs store compressed data in alternate alignments, i.e., compressed data in even-numbered WLs is right-aligned and compressed data in odd-numbered WLs is left-aligned; note that storing uncompressed data is not impacted by a WL's alignment. Figures 4(a) and 4(b) illustrate memory organization without and with ADAM, respectively. In figure 4(b), ADAM employs compression to compress 16-bit (16-bit) useful data in WL_0 (WL_1) to 7-bit (3-bit) useful data and aligns the compressed data to the right (left) in WL_0 (WL_1). Data in WL_2 is uncompressed and right-aligned by default. The alignment of compressed data stored in the memory is decided based on its WL address parity (i.e., even or odd), and hence does not require additional book-keeping. Note that ADAM does not alter the endianness of a system; the ADAM codec decompresses and realigns the data fetched from memory to its original format before forwarding it to the processor.

In summary, by combining data compression with alternate data alignment in consecutive WLs, ADAM minimizes the overlap between the PCM cells that store useful data in the adjacent WLs. As a result, when data is written to a WL, the heat dissipated minimally impacts the cells storing useful data in the adjacent WLs. Figure 5 illustrates BL WD mitigation in ADAM. As shown in the figure, WL_1 is currently active and causing WD in its adjacent WLs, i.e,

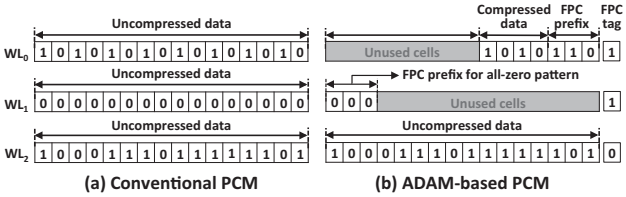

Figure 4: **This figure contrasts data organization in ADAM-based PCM with conventional PCM. Unlike conventional PCM, which stores the write-back data as-is, ADAM employs pattern-based data compression to compress the write-back data and then stores the compressed data in right-/left-aligned format for even-/odd-numbered WLs. This reduces the overlap between useful data in consecutive WLs. However, if the data is incompressible, e.g., WL_2, ADAM stores the data as-is.**

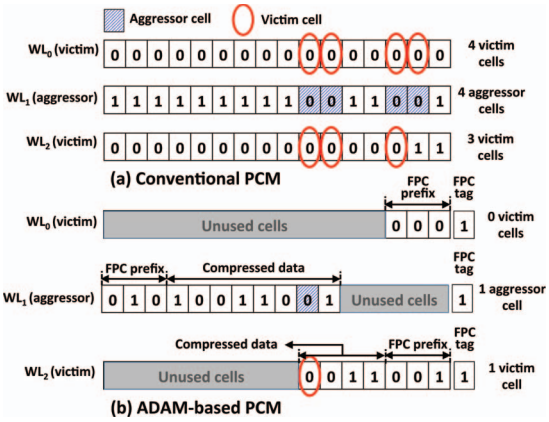

Figure 5: **This figure depicts BL WD in (a) conventional PCM and (b) ADAM-based PCM. By combining data compression with alternate data alignment in consecutive WLs, ADAM reduces the number of aggressor (victim) cells from 4 (7) to 1 (1).**

WL_0 and WL_2. When data is written to conventional PCM (figure 5(a)) the 4 aggressor cells can affect up to 7 idle cells in the adjacent WLs. In contrast, due to data compression and alternate data alignment, ADAM-based PCM (figure 5(b)) reduces the number of aggressor (victim) cells to 1 (1). Note that similar to DIN [3] and SD-PCM [4], ADAM also employs VnR for WD error recovery. However, since ADAM effects a significant reduction in potential WD errors, the number of VnR iterations required to converge to a WD-free state is small in practice.

3.2 Deferred WD Correction (DEFT)

Conventionally, VnR is employed to eliminate all WD errors (i.e., in active as well as adjacent WLs) on a memory write. The steps involved when a cache line (which belongs to word line WL_n) is written are as follows. First, since a write can disturb memory cells in the adjacent WLs, the victim cache lines from WL_{n-1} and WL_{n+1} are read and stored in two 64-byte (512-bit LLC line size) registers on the DIMM. This step involves two full read cycles. Second, the dirty cache line is written to WL_n in 1 write cycle. Third, VnR is employed to recover from WL WD errors in WL_n. Each VnR step incurs a latency overhead of one read and one write cycle. Finally, VnR is employed to recover from WL WD errors (induced as BL WD errors by WL_n) in the adjacent WLs, i.e, in WL_{n-1} and WL_{n+1}. Note that VnR requires additional processing of the adjacent WLs to recover from the WD errors that occur during VnR. As a result, the effective write latency of a cache line is increased from one write cycle to several read and write cycles, negatively impacting performance.

State-of-the-art SD-PCM [4] employs PreRead, which performs opportunistic prefetching of the data from the adjacent WLs, while the write request is still waiting in the write-back queue. Although PreRead eliminates the latency overhead of reading data from adjacent WLs, SD-PCM still needs to perform the post-write VnR on the adjacent WLs for BL WD error recovery.

In order to improve system performance in the presence of WD errors, ADAM complements WD error mitigation with DEFT, which opportunistically defers latency-intensive WD error recovery in the adjacent WLs to reduce the effective write latency. *DEFT leverages the key observation that if a cache line is already in the LLC, it will not be read from the main memory, and hence, even if the main memory data for that cache line is corrupted, it does not impact system reliability.* Thus, DEFT defers WD error recovery in the adjacent WLs if their data (i.e., affected cache lines) is already present in the LLC. In order to ensure that the corrupted data in the adjacent WLs is overwritten by the correct data subsequently, DEFT sets the dirty bit of the affected cache lines in the LLC. Hence, for no additional overhead, DEFT opportunistically eliminates VnR of the adjacent WLs to reduce the effective latency of a write operation. Note that the overhead of 2 LLC lookups for WL_{n-1} and WL_{n+1} is insignificant in comparison to the write latency of PCM.

Since DEFT does not perform instant WD error recovery, it may appear that integrating DEFT with PCM-based main memory renders the main memory partially stale, thereby undermining the ability of the memory (which is non-volatile) to ensure instant data recovery in the face of power/system failures. However, stale main memory is a common problem for any system with a non-volatile main memory (NVMM) and volatile write-back caches. To address this problem, designers either replace the volatile LLC with a non-volatile LLC [15], or modify the software to combine cache flush (e.g., clflush) and memory fence (e.g., mfence) instructions to flush the LLC data to the main memory in a regular and orderly manner [16]. More recently, asynchronous commit of transactions [17] and reordering of persistent memory writes [18] have also been proposed as high performance consistency-preserving solutions for NVMMs. A detailed discussion of these mechanisms is beyond the scope of this work. We assume that a consistency-preserving mechanism is present in the system, and hence, the modification introduced by DEFT – delayed WD error recovery of the cached data in the adjacent WLs – does not render memory partially stale.

3.3 ADAM: Hardware Overhead

ADAM incurs memory overhead of 1-bit compression tag per 512-bit cache line (i.e., 0.19% memory overhead). Further, the ADAM codec consists of a 64-bit FPC-based data compression engine and a data alignment unit to preserve endianness of the system. A 64-bit FPC engine has a hardware overhead of $\approx$ 10k NAND gates and encoding/decoding latency of 3/2 cycles [12]. The data alignment unit consists of eight 64-bit 2-to-1 multiplexers, which has a hardware overhead of 2.5k NAND gates and a shift latency of 1 cycle [19]. The entire ADAM codec is integrated within the DIMM-side memory controller. The logic area as well as latency of the ADAM codec is negligible in comparison to a 16GB SLC PCM module that has a read/write latency of 400/600 cycles [3].

4. Evaluation Methodology

Our evaluation of ADAM uses (i) trace-based simulations to evaluate WD errors per write and its impact on write latency, error correction energy, and memory lifetime, and (ii) system-level simulations to study the impact of WD mitigation on system performance.

WD error modeling: We adopted the PCM heat dissipation model and PCM thermal disturbance model from [3] to compute WD error rates for 20nm PCM. Based on the PCM heat dissipation model, resetting a cell increases the temperatures of the adjacent cells in the same WL and BL by 310°C and 320°C, respectively. This rise in temperature is used to compute WL and BL WD error rates using the PCM thermal disturbance model, which follows a Weibull distribution. The WD error rates for the victim cells in the WL and BL are 9.9% and 11.5%, respectively. This is consistent with the results reported in [4].

Trace-driven simulation: For deep, multi-billion instruction evaluation, we performed trace-based simulations using NVMain [6]. We configured NVMain to reflect 16GB main memory with a single channel, 1 rank, and eight x8 devices/rank. The memory controller performs first-ready first-come-first-serve (FR-FCFS) scheduling, with open page policy. We configure NVMain using SLC PCM energy/latency parameters from [1, 3]; MLC PCM energy/latency parameters are taken from [3, 20]. The memory traces were generated from the SPEC CPU2006 [8] benchmarks using the Intel Pin [21] binary instrumentation tool.

Workload	Benchmarks	MPKI	WPKI
WD_1	perlbench, soplex, bwaves, lbm	21.05	9.10
WD_2	perlbench, soplex, milc, povray	14.20	5.58
WD_3	h264ref, astar, milc, lbm	28.22	12.80
WD_4	h264ref, sjeng, bwaves, povray	20.40	8.79
WD_5	perlbench, leslie3d, GemsFDTD, lbm	12.71	5.02
WD_6	bzip2, gcc, sphinx3, xalancbmk	13.24	4.84
WD_7	bzip2, gcc, mcf, omnetpp	10.30	3.72
WD_8	bzip2, mcf, namd, omnetpp	8.30	2.86
WD_9	GemsFDTD, gcc, omnetpp, xalancbmk	7.69	2.17

Table 1: **Composite workloads comprising of SPEC CPU2006 benchmarks [8] for full system evaluation of ADAM.**

Full-system simulation: For system-level simulations, we used the MARSS full-system simulator [7]. MARSS is configured to simulate a standard 4-core out-of-order system running at 3GHz. Each core is assigned a private L1 instruction and data cache of 32kB each and a private L2 cache of 128kB; the L3 cache (LLC in this case) is a single, shared write-back cache of 8MB. Finally, the 16GB single channel PCM main memory has 1 rank and 8 banks.

Workloads: To evaluate system performance in real-world scenarios, we use 9 composite workloads (listed in Table 1) with each workload containing 4 benchmarks from the SPEC CPU2006 [8] benchmark suite. For fairness, we evaluate a variety of workloads with different (high, medium, and low) memory accesses per kilo-instruction (MPKI) and writes per kilo-instruction (WPKI) values.

5. Results

This section compares the aggregate (WL+BL) WD errors per write, error correction energy, write latency, system performance (measured in instructions per cycle (IPC)), and memory lifetime for the baseline, DIN [3], SD-PCM [4], and ADAM. Our **baseline** system uses 20nm PCM without any thermal guard bands and/or ECC support for WD mitigation; the baseline relies exclusively on VnR for WD error recovery. Furthermore, to keep the memory area overhead of DIN equivalent to other techniques, we assume that no thermal bands are available between WLs; VnR is employed for both WL and BL WD error recovery. Finally, all evaluated techniques integrate bit-level DCW [14] for bit write reduction.

5.1 Summary

We first provide a summary of our results; a detailed analysis is presented in the subsequent sections. Table 2 summarizes ADAM's performance on SLC and MLC PCM architectures. We observe that ADAM achieves maximum reduction in WD errors, which translates to a significant reduction in error correction energy and write latency, and improvement in IPC and memory lifetime.

Main memory	WD mitigation	Errors per write	Energy for error correction	Effective write latency	IPC	Lifetime
SLC PCM	Baseline	17.5	1×	1×	1×	1×
	DIN	6.64	0.46×	0.78×	1.14×	1.48×
	SD-PCM	4.16	0.25×	0.58×	1.26×	1.51×
	ADAM	**2.86**	**0.12×**	**0.49×**	**1.35×**	**1.78×**
MLC PCM	Baseline	18.5	1×	1×	1×	1×
	DIN	5.13	0.29×	0.81×	1.15×	1.57×
	SD-PCM	4.11	0.23×	0.72×	1.20×	1.59×
	ADAM	**1.64**	**0.09×**	**0.53×**	**1.37×**	**1.90×**

Table 2: **Comprehensive comparison of ADAM with the baseline, DIN [3], and SD-PCM [4]. WD error correction energy, effective write latency, IPC, and memory lifetime are normalized to the baseline.**

5.2 SLC PCM: WD errors per write

Figures 6(a), 6(b), and 6(c) report the aggregate (WL+BL), WL, and BL WD errors per write, respectively, for the baseline (i.e., raw bit error rate (RBER)), DIN, SD-PCM, and ADAM. ADAM employs data compression to reduce the write-back width, which in turn reduces the number of aggressor as well as victim cells when the write-back data is stored in the active WL. As a result, the number of WL WD errors per write for ADAM is low and comparable to DIN/SD-PCM, which specifically target WL WD errors by eliminating WD-vulnerable data patterns. Furthermore, by combining data compression with alternate data alignment, ADAM maximizes the lateral separation between the PCM cells that store useful data in the adjacent WLs, thereby significantly reducing BL WD errors on a memory write. In contrast, DIN does not employ any WD mitigation solution for BL WD errors and SD-PCM relies exclusively on idle ECPs in adjacent WLs for temporarily recovering from BL WD errors. Hence, the BL WD error rates for DIN and SD-PCM are significantly higher than ADAM. Overall, ADAM reduces aggregate WD errors per write by 6×, 2.3×, and 1.45× in comparison to the baseline, DIN, and SD-PCM, respectively.

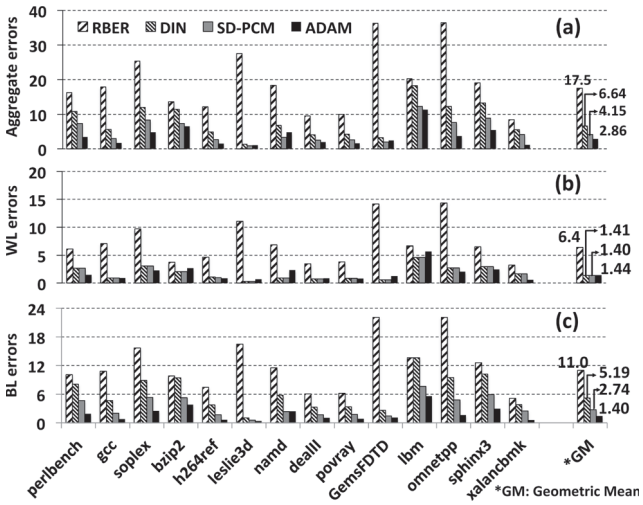

Figure 6: **This figure reports (a) aggregate (WL+BL) WD errors, (b) WL WD errors, and (c) BL WD errors per memory write for the baseline (RBER), DIN, SD-PCM, and ADAM. Our results show that ADAM reduces the aggregate WD errors by 6×, 2.3×, and 1.45× in comparison to the baseline, DIN, and SD-PCM, respectively.**

5.3 SLC PCM: WD error correction energy

Figure 7 reports WD error correction energy, normalized to the baseline for DIN, SD-PCM, and ADAM. Our results show that ADAM reduces WD error correction energy by 87%, 74%, and 50% in comparison to the baseline, DIN, and SD-PCM, respectively. The significant reduction in WD error correction energy of ADAM can be attributed to the reduced VnR operations required for WL and BL WD error recovery. In contrast, DIN and SD-PCM observe high BL WD error rates, which require multiple VnR operations, consuming high WD error correction energy.

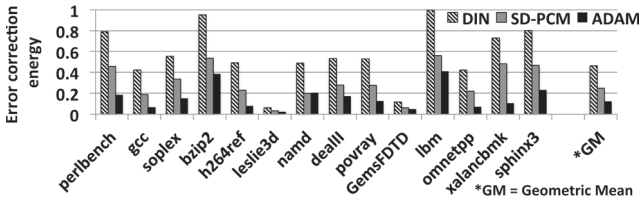

Figure 7: **WD error correction energy (normalized to the baseline) for DIN, SD-PCM, and ADAM. ADAM reduces WD error correction energy by 87%, 74%, and 50% in comparison to the baseline, DIN, and SD-PCM, respectively.**

5.4 SLC PCM: Effective write latency

Figure 8 reports the effective write latency, normalized to the baseline, for DIN, SD-PCM, and ADAM. Since ADAM achieves the lowest WD error rate, ADAM requires the least number of VnR operations for WD error recovery. Additionally, by employing DEFT, ADAM further eliminates latency-intensive VnR operations

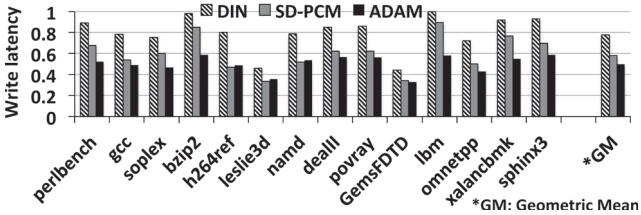

Figure 8: **Write latency (normalized to the baseline) of DIN, SD-PCM, and ADAM. ADAM reduces write latency by 51%, 38%, and 15% in comparison to the baseline, DIN, and SD-PCM, respectively.**

for the cached data in the adjacent WLs. Hence, we observe that the effective write latency of ADAM is 51%, 38%, and 15% lower in comparison to the baseline, DIN, and SD-PCM, respectively.

5.5 SLC PCM: System performance (IPC)

The impact of WD error mitigation on IPC (normalized to the baseline) for DIN, SD-PCM, and ADAM are reported in figure 9. Our results show that in comparison to the baseline, DIN, and SD-PCM, ADAM improves IPC by 35%, 18%, and 7%, respectively. The high IPC of ADAM can be attributed to its low write latency due to reduced VnR overhead. Furthermore, since ADAM optimizes memory writes to reduce WD error recovery, the effectiveness of ADAM becomes more evident for high WPKI workloads (e.g., WD_1 and WD_3). For low WPKI workloads (e.g., WD_8 and WD_9), the IPC improvements from ADAM are less significant.

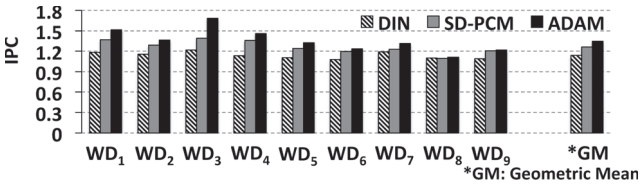

Figure 9: **IPC (normalized to the baseline) of DIN, SD-PCM, and ADAM. ADAM improves IPC by 35%, 18%, and 7% in comparison to the baseline, DIN, and SD-PCM, respectively.**

5.6 SLC PCM: Memory lifetime

Figure 10 reports the impact of WD error correction on memory lifetime for DIN, SD-PCM, and ADAM, normalized to the baseline. ADAM improves memory lifetime by 78%, 20%, and 18% in comparison to the baseline, DIN and SD-PCM, respectively. The high memory lifetime of ADAM results from (i) smaller write-back width due to data compression and (ii) reduction in WD errors, which reduces multiple re-writes of the erroneous cells. In contrast, DIN and SD-PCM experience a higher cell flip rate due to (i) (3,4) data encoding, which increases the data width, and (ii) more WD errors, which require multiple re-writes of the erroneous cells.

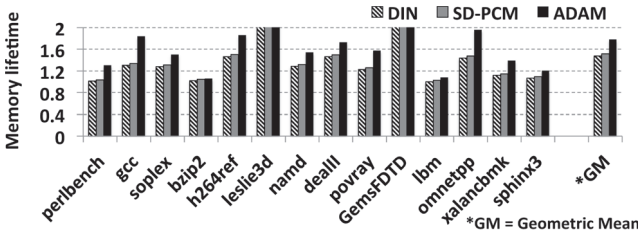

Figure 10: **Memory lifetime of DIN, SD-PCM, and ADAM, normalized to the baseline. ADAM reduces cell flip rate by employing data compression and minimizing re-writes due to WD errors. As a result, ADAM improves memory lifetime by 78%, 20%, and 18% in comparison to the baseline, DIN, and SD-PCM, respectively.**

5.7 Evaluation on MLC PCM

ADAM was also evaluated on a memory system based on MLC PCM. Since an MLC stores 2 logical bits per physical cell, a 512-bit cache line is stored in 256 physical cells. We used the WD error model for MLC PCM provided in [3]. Furthermore, we employ single-set multiple-reset (SSMR) program-and-verify (P&V) for writing data to MLC PCM. SSMR P&V causes fewer WD errors in comparison to single-reset multiple-set (SRMS) P&V. A summary of MLC PCM results is presented in Table 2. Similar to SLC PCM, ADAM achieves the lowest WD error rate, i.e., only 1.64 WD (WL+BL) errors per write-back. This translates to 61% reduction in error correction energy, 15% gain in IPC, and 19% higher NVM lifetime in comparison to state-of-the-art SD-PCM. A detailed discussion of MLC PCM results is omitted for brevity.

6. Conclusions

ADAM is a low overhead, high performance WD mitigation solution for scaled PCM-based main memories. ADAM combines pattern-based data compression with alternate data alignment to reduce both WL and BL WD errors. Furthermore, ADAM also integrates DEFT, which opportunistically defers WD error recovery of cached data in the adjacent WLs, improving system performance without impacting memory reliability. Results show that ADAM outperforms state-of-the-art WD mitigation solutions, with the lowest WD error rate, WD error correction energy, and effective write latency, as well as the highest IPC and memory lifetime.

References

[1] B. C. Lee *et al.*, "Architecting phase change memory as a scalable DRAM alternative," in *Proc. Intl. Symposium on Computer Architecture*, 2009.

[2] *International Technology Roadmap for Semiconductors*, 2011.

[3] L. Jiang *et al.*, "Mitigating write disturbance in super-dense phase change memories," in *Proc. Intl. Conference on Dependable Systems and Networks*, 2014.

[4] R. Wang *et al.*, "SD-PCM: Constructing reliable super dense phase change memory under write disturbance," in *Proc. Intl. Conference on Architectural Support for Programming Languages and Operating Systems*, 2015.

[5] S. Swami and K. Mohanram, "Reliable non-volatile memories: Techniques and measures," *IEEE Design and Test*, 2017.

[6] M. Poremba and Y. Xie, "NVMain: An architectural-level main memory simulator for emerging non-volatile memories," in *Proc. IEEE Computer Society Annual Symposium on VLSI*, 2012.

[7] A. Patel *et al.*, "Marss: A qemu-based micro-architectural and systems simulator for x86 multicore processors," in *Proc. Design Automation Conference*, 2011.

[8] J. L. Henning, "SPEC CPU2006 benchmark descriptions," in *ACM SIGARCH Computer Architecture News*, 2006.

[9] S. Lin and D. J. Costello, *Error Control Coding*. Pearson Higher Edu., 2004.

[10] A. R. Alameldeen and D. A. Wood, "Frequent pattern compression: A significance-based compression scheme for L2 caches," *Univ. Wisconsin-Madison, C.S. Tech. Rep 1500*, 2004.

[11] G. Pekhimenko *et al.*, "Base-delta-immediate compression: Practical data compression for on-chip caches," in *Proc. Intl. Conference on Parallel Architectures and Compilation Techniques*, 2012.

[12] P. M. Palangappa and K. Mohanram, "CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM," in *Proc. Intl. Symposium on High-Performance Computer Architecture*, 2016.

[13] L. Jiang *et al.*, "Improving write operations in MLC phase change memory," in *Proc. Intl. Symposium on High-Performance Computer Architecture*, 2012.

[14] B. D. Yang *et al.*, "A low power phase change random access memory using a data-comparison write scheme," in *Proc. Intl. Symposium on Circuits and Systems*, 2007.

[15] J. Zhao *et al.*, "Kiln: Closing the performance gap between systems with and without persistence support," in *Proc. Intl. Symposium on Microarchitecture*, 2013.

[16] K. Lee, S. Ryu, and H. Han, "Performance implications of cache flushes for non-volatile memory file systems," in *Proc. Intl. Symposium on Applied Computing*, 2015.

[17] S. Pelley, P. M. Chen, and T. F. Wenisch, "Memory persistency," in *Proc. Intl. Symposium on Computer Architecture*, 2014.

[18] Y. Lu *et al.*, "Loose-ordering consistency for persistent memory," in *Proc. Intl. Conference on Computer Design*, 2014.

[19] N. Weste and D. Harris, *CMOS VLSI Design: A circuits and systems perspective*. Addison-Wesley Publishing Company, 2010.

[20] M. Zhao *et al.*, "Minimizing MLC PCM write energy for free through profiling-based state remapping," in *Proc. Asia and South Pacific Design Automation Conference*, 2015.

[21] C.-K. Luk *et al.*, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proc. Conference on Programming Language Design and Implementation*, 2005.