WoM-SET: Low Power Proactive-SET-based PCM Write using WoM Code

XianWei Zhang†, Lei Jiang ‡, Youtao Zhang †, Chuanjun Zhang ⋆, Jun Yang ‡
† Computer Science Department, University of Pittsburgh, Pittsburgh, PA 15260
‡ Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, PA 15261
⋆ Intel Science and Technology Center for Embedded Computing, Pittsburgh, PA 15213
†, ‡{xiz81,lej16,youtao,juy9}@pitt.edu ⋆chuanjun.zhang@intel.edu

ABSTRACT

The emerging Phase Change Memory (PCM), while having many advantages, suffers from slow write operations. This is mainly due to its asymmetric write characteristic, i.e., for two types of write operations of PCM, SET is much slower than RESET. Recent study has shown that proactively setting dirty memory lines to all '1's can enable RESET-only writes when these lines are written back from the cache, which helps to reduce the effective write latency. Unfortunately, it results in higher write power demand. In this paper, we propose WoM-SET, a low power proactive-SET-based write strategy. By exploiting the WoM (write-once memory) code, we greatly reduce the number of RESETs per write and hence the write power demand. By applying our design only to write-intensive pages, we restrict the extra space requirement in WoM-SET. Our experiments show that WoM-SET achieves 40% RESET bit reduction, 40% write power reduction, and 12% energy-delay-product improvement over the PreSET scheme.

Keywords

Phase Change Memory, WoM code

1. INTRODUCTION

Modern multi-core systems, with more and more cores integrated on one single chip, exhibit increasing demand for large main memory capacity. Unfortunately, traditional DRAM technology faces serious challenges in high power consumption, serious process variation and poor scalability [6]. In particular, the path to scale DRAM under $22nm$ is still unclear [6]. In recent years, Phase Change Memory (PCM) [17] has emerged as a promising memory technology for future multi-core systems. Studies have shown that it is beneficial to replace a significant portion of DRAM with PCM in main memory subsystem [24, 13, 11].

While PCM has many advantages, such as better scalability, zero cell leakage, and DRAM-comparable read latency, one major drawback of PCM is its slow write operation. Slow writes not only degrade the overall performance but also the memory bandwidth of a PCM based system. Schemes have been developed to address this issue. The *write cancellation* [14] technique allows read operations to preempt on-going writes, which mitigates the effect of long latency writes on system performance. The *write truncation* mechanism [9] reduces MLC write latency leveraging ECC. The PreSET [15] scheme, which this paper is based on, exploits the asymmetric PCM write characteristic. Writing a '1' (SET) is much slower than writing a '0' (RESET). The PreSET scheme proactively sets dirty memory lines when the memory bank is idle (this operation is referred to as *proactive-SET* in this paper). When those lines are actually written back to the memory, only fast RESETs are performed, reducing the effective write latency. Since write-backs tend to have large impact on performance critical read operations, reducing the latency of these writes helps to improve the system performance.

Unfortunately, the PreSET scheme has two drawbacks. (i) It increases write power significantly. Studies have shown that a write request often changes a few bits within one memory line. *Differential-write* reduces write power by only writing those to-be-changed bits [24]. the PreSET scheme sets all bits non-discriminatively and unnecessarily introduces more bit changes. The write power was reported to increase by ∼225% [15]. (ii) It impairs the lifetime of the PCM. This is because the PreSET scheme increases bit changes in the RESET phase, and the lifetime of PCM depends mainly on the frequency of RESETs. It was reported that the lifetime of a PCM is reduced by ∼60% [15]. Given that write power and write endurance are major limitations of PCM, degrading both factors jeopardizes the applicability of the PreSET scheme in PCM-based memory systems.

In this paper, we propose WoM-SET, a proactive-SET operation based PCM write scheme using the WoM (Write-Once-Memory) code [18]. WoM code was originally developed for optical disks [18] and has recently been adopted in Flash memories [8]. We exploit WoM code taking advantage of the asymmetric write characteristic of PCM. On the one hand, encoding a memory line with write-twice WoM code allows us to perform one proactive-SET on every two writes, mitigating bit change increases. On the other hand, the write-twice WoM code requires 50% extra space. To alleviate the space requirement, we selectively perform WoM encoding on only the write-intensive pages. The contributions of our techniques are as follows.

- We present WoM-SET, a proactive-SET based PCM write scheme using WoM code. It reduces the frequency of proactive-SETs. By reducing bit changes per write, WoM-SET not only reduces write power but also extends PCM lifetime effectively.
- We present architectural designs to restrict WoM encoding to write-intensive pages only. A small table is integrated on-chip to track these pages, which guides when and how to enable WoM encoding for each PCM page. This approach achieves better tradeoff between bit change reduction and space overhead.
- We evaluate the proposed scheme and compare it with the PreSET scheme [15]. Our experimental results show that WoM-SET achieves 40% RESET bit reduction, 40%

*This work was supported in part by NSF CSR #1012070 and NSF CAREER #0747242.

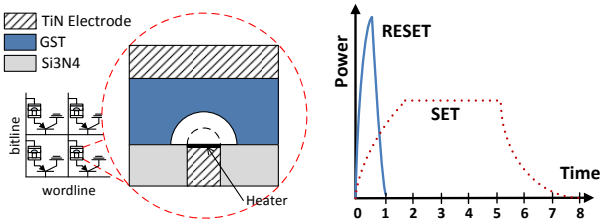
write power reduction, and 12% energy-delay-product improvement over the PreSET scheme.

The rest of the paper is organized as follows. Section 2 introduces PCM basics and WoM code. Section 3 motivates our design and elaborates the details of Wom-SET. Section 4 presents the experimental methodology. Section 5 analyzes our experimental results. Section 6 concludes the paper.

2. BACKGROUND

PCM basics: Phase Change Memory (PCM) technology stores information using phase changing material such as GST ($Ge_2Sb_2Te_5$) [17]. A PCM cell consist of GST and two electrodes attached from top and bottom. By injecting electrical pulses, the GST can be programmed into low or high resistance states, representing bit '1' and '0' respectively. A PCM chip has traditional array structure with one access transistor for each memory cell (a.k.a., 1T1C), as shown in Figure 1(a). [16] presents a detailed analytic PCM cell model. PCM has many advantages. PCM has zero leakage from the cell. PCM have very good scalability, e.g., $4F^2$ PCM cell size at $20nm$ has been reported [3]. PCM read is comparable to DRAM read [11].

There are two PCM write operations. A large magnitude, short duration pulse can RESET a PCM cell to large resistance state (bit '0'), while a low magnitude, long duration pulse can SET a cell to low resistance state (bit '1'). This is referred to as PCM's write asymmetry.



(a) PCM array (b) PCM write

Figure 1: PCM basics.

The major drawbacks of PCM are short write endurance, long write latency, and large write power. To prolong PCM lifetime, *differential-write* [24] only writes the changed bits of a memory line. *Flip-n-write* [2] guarantees that the number of changed bits is no more than half of memory line size. Wear leveling techniques [12, 24, 20] distribute writes across the whole memory space, which prevents write-intensive blocks from failing the PCM chip. The lifetime of PCM memory system can be further improved by error correction code [19, 23]. To alleviate long write latency on performance, *write cancellation* [14] allows performance critical read operations to preempt on-going write operations. *Write truncation* [9] uses error correction code to reduce the number of write iterations of multi-level cell PCM and thus improves write performance.

PreSET. PreSET, the scheme that our design is based on, exploits PCM's write asymmetry to improve system performance [15]. Once a cacheline becomes dirty, the scheme proactively sets its associated memory line to all 1s as long as the memory bank is idle. When the line is later expunged from the cache, only fast RESET operations are needed to update the information into PCM memory.

In this paper, the write that sets the line to all 1s during bank's idle interval is referred to as **proactive-SET** operation; the write that is due to the dirty line being expunged from cache is referred to as **write-back write**. Since write-back writes have large interference with performance critical read operations, reducing the latency of these writes helps to improve the system performance.

2-bit data	1st-write	2nd-write
00	000	111
01	001	110
10	010	101
11	100	011

Table 1: 3-bit WoM code for Write-twice Operation

Write once Memory (WoM) Code: WoM code was first introduced to store data on memories that change elements in only one direction, e.g., 0→1 on optical disks [18]. In addition, such transition usually can happen only once, i.e., a bit can be written only once. The 3-bit WoM code that encodes 2-bit information is shown in Table 1.

WoM code has been applied to Flash memory recently [8]. After a block erase, a flash page contains all 0s. For the first update after the erase, the page is encoded using the *1st-write* code, as shown in Table 1. For the 2nd update to the page, the changed bits are encoded using the *2nd-write* code. Unchanged bits are not updated in the second write. By allowing one more update on Flash before erasing the page, WoM code can extend Flash lifetime and improve its write performance.

While WoM code can be generalized to allow more than two writes, the space overhead would also increase dramatically [21]. Coset coding [5] was recently applied to represent one bit combination with multiple codes on PCM [7]. While it can help to minimize bit changes, coset code cannot guarantee one direction bit transition.

3. WOM-SET: LOW POWER PROACTIVE-SET BASED PCM WRITE

In this section, we first motivate our design by showing increased bit changes due to proactive-SET. We then use an example to illustrate how WoM-SET works and elaborate the architecture details for achieving better design tradeoffs.

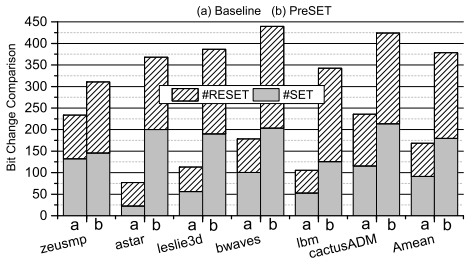


Figure 2: PreSET increases both SET and RESET operations (128B line size).

3.1 Motivation

While the proactive-SET operation enables fast RESET-only write-back writes, it increases the number of bits that each write needs to change. Figure 2 compares the bit changes before and after applying proactive-SET. The experiment setting is listed in Section IV. On average, for a 128B line size, the baseline that performs *flip-n-write* needs to set 91 bits and reset 77 bits, while the PreSET scheme needs to set 180 bits and reset 200 bits, representing 98% and 160% increases respectively. This is because proactive-SET unnecessarily sets many unchanged bits. Each such bit produces one extra SET and one extra RESET operations. As RESET consumes about 5× more write power than SET [11], and PCM lifetime depends mainly on RESET operations [10], the PreSET scheme has around 225% write power increase, 30% system power increase, and 60% lifetime degradation [15]. Since both write

power and write endurance are major limitations of PCM, degrading both factors jeopardizes the applicability of the PreSET scheme in PCM based memory systems.

3.2 WoM-SET: Low Power Proactive-SET based PCM Write

To address the increased bit changes in the PreSET scheme, we propose Wom-SET, a proactive-SET based write scheme using WoM code. Since PCM has slow 0→1 bit transition (i.e., SET operation) and fast 1→0 bit transition (i.e., RESET operation), WoM-SET adopts the WoM code as shown in Table 2.

2-bit data	1st-write	2nd-write
00	111	000
01	110	001
10	101	010
11	011	100

Table 2: WoM code for Phase Change Memory

Figure 3 illustrates how WoM-SET works. Given an 8-bit line (for illustration purpose), the PreSET scheme stores the line using 8 PCM cells while WoM-SET requires 12 cells — each 2-bit data requires 3-bit WoM code. Let us assume the line stores '01 01 01 01', the WoM-encoded format is '110 001 110 110' (we will explain why the data may be stored like this), and the new data is '10 01 01 00'. The baseline write scheme needs to SET the first bit, and RESET the second and the eighth bits. For this write, both the PreSET and WoM-SET schemes need to proactively SET the line to all 1s, resulting in 4 SET and 5 SET operations respectively. Then the PreSET scheme needs 5 bit RESET to get the data updated.

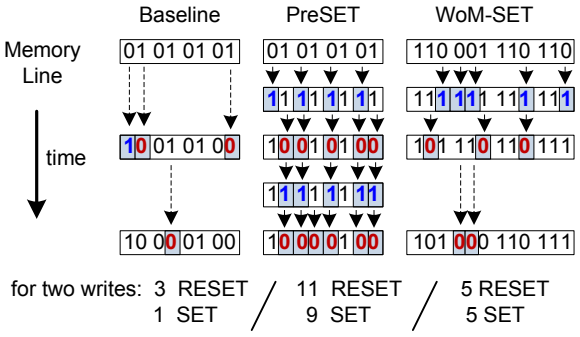


Figure 3: Comparing baseline, PreSET, and WoM-SET.

For the first write after proactive-SET, WoM-SET always uses the 1-write code in Table 2, and thus updates the WoM-encoded line to '101 110 110 111', which requires 3 bit RESET. Next, assume a new write request updates the data to '10 00 01 00'. The PreSET scheme has to go through the proactive-SET and RESET steps again, resulting in 5 bit SET and 6 bit RESET. WoM-SET, instead, can sustain another write before proactive-SET. It exploits the 2nd-write code for the changed bits in the line, i.e., the second bit combination needs to be changed from 01 to 00. Since the WoM-encoded line after the update is '101 000 110 111', we only need 2 bit RESET while no SET is needed.

After being written twice after proactive-SET, a WoM-encoded line contains both 1st-write and 2nd-writ codes, e.g., it contains both 000 and 111, and both codes represent 2-bit data 00. Since it is impossible to change 000 to other 2-bit data by just using RESET, future writes to the line need a proactive-SET before RESET write.

Encoding/Decoding overheads. Adopting WoM code requires bit encoding/decoding at runtime. Given 2-bit data 'a_1a_2', we first compute $r=a_1\&a_2$, and then get the 1st-write code using '$\overline{rrr}$ XOR $0a_1a_2$', and the 2nd-write code using 'rrr XOR $0a_1a_2$', respectively. Given 3-bit WoM code '$b_1b_2b_3$'(in either the 1st-write or the 2nd-write code), we get the 2-bit data using 'b_1b_1 XOR b_2b_3'. Our VHDL simulation showed that either the encoder or the decoder takes less than 1ns, and consumes less than 1μW power per PCM line. The overheads are negligible for the simulated PCM chip that has 125ns read latency, 1ms write latency, and 544 μW idel power (as shown in Section IV).

Comparison with CoSET. CoSet coding [5] generalizes the one-to-many coding principle in WoM coding [18]. A recent work [7] adopts CoSet coding to extend the lifetime of PCM by choosing the code that can minimize bit changes for each write. Since the controller does not differentiate the latency of 0→1 and 1→0 bit transitions, the design [7] still has mixed bit transitions in each write and thus cannot improve write latency. WoM-SET follows WoM coding and is a special format of CoSet coding. By eliminating 0→1 bit transitions for the two consecutive writes after proactive-SET, WoM-SET improves PCM write latency. Since some bits are unnecessarily SET during the proactive-SET operation, WoM-SET hurts PCM lifetime.

Comparison with PreSET. The PreSET scheme needs one proactive-SET for every line write while WoM-SET performs one proactive-SET for every other write to the same line. By reducing the frequency of proactive-SET operations, WoM-SET exhibits two advantages: (1) it requires less memory bandwidth; (2) it resets less number of bits. In particular, for the 2nd write of WoM-SET, only changed bits are updated, resulting in reduced write power and prolonged chip lifetime.

However, WoM-SET has a major drawback — encoding 2-bit data using 3 bits requires 50% extra space. Therefore, we need proper architectural innovation to find better tradeoff between reduced bit changes and increased space demand.

3.3 Architectural Designs

The architectural enhancement to enable Wom-SET is shown in Figure 4. Our intuitive is to apply WoM encoding only to write-intensive pages while leaving other pages in plaintext. Since WoM encoding benefits consecutive writes to the same line, by encoding only write-intensive pages, we maximize the benefits on bit change, write power, and memory bandwidth reduction while minimizing extra space demand.

Tracking write-intensive pages. In Figure 4, a 32K-entry 16-way set-associative table is added to track write-intensive pages and record the indices of all WoM-encoded pages. Each entry in the table consists of a page index (30-bit tag) and a counter (11 bits). The counter approximates the write frequency of the page. For each 16-entry set, the 8 pages with the largest counters are identified as write-intensive pages while the rest are candidates. Track potential write-intensive pages in the table helps to improve tracking accuracy with no frequent page conversion to and from WoM-encoded format.

A 4KB page, after being identified as a write intensive page, is remapped to a 8KB block in a reserved PCM memory region. The remapping is transparent to the OS such that neither does the OS update the page table nor is the OS notified. The original page frame in the physical memory is kept. By mapping one page to a two-page block, WoM-SET provides sufficient space for WoM encoding. The mapping is statically fixed, i.e., the first entry of the table is mapped to the first 2-page block of the reserved area, the second entry to the second block, and so on so forth. Thus, the size of the reserved space is 256MB (= 32K entry × 8KB/entry).

For a memory access to the last level cache, its page index

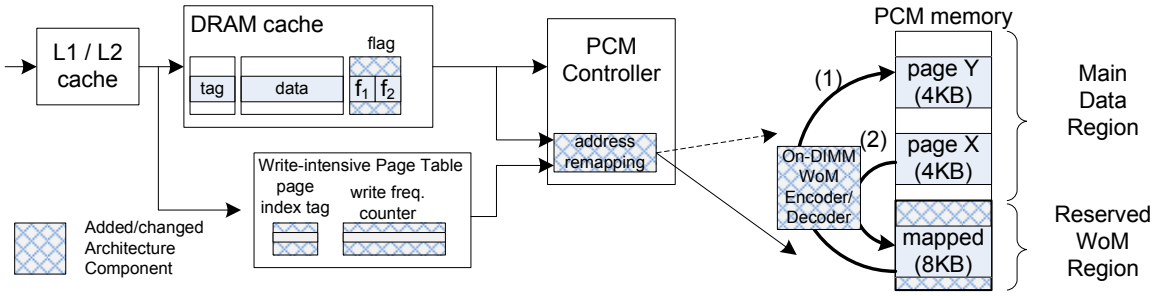


Figure 4: The architectural details of WoM-SET.

is searched in the table. A hit in the table increments its associated frequency counter while a miss replaces the entry with the smallest counter in the set. We left-shift the counter periodically (after each one-billion instruction epoch) to avoid keeping stale information in the table for too long.

Incrementing the frequency counter may bring a page from the 8-page least-frequently-written page group to the 8-page most-frequently-written page group, i.e., it becomes a write-intensive page. Assume the new page is X while the one falling to least-frequently-written page group is Y. At this time, the controller reads the WoM-content of page Y from its associated reserved block, decodes the contents, and writes the plain text to its OS-allocated page location. Then, the controller reads the plain text of page X, encodes it using WoM code, and writes the encoded content to the reserved block. The control is done by the a bridge chip on DIMM [4], which does not need to transfer the data to and from the processor chip. Figure 4 illustrates the transition steps.

While this replacement is very expensive, recent study showed that the working set of an application is much smaller than the resident set [1]. The set of write intensive pages is stable, resulting in rare page conversion. In this paper, we convert at most 256MB write intensive pages, which incurs less than 2% performance degradation.

Enhancement to the last level cache (LLC). We tag each cache line in LLC with a two-bit flag f_1f_2 to indicate how the line is stored in the PCM memory. The meaning of bits f_1f_2 is as follows — '01' indicates that the line is being proactively set to all 1s; '10' indicates that the line has been successfully set to all 1s; '11' indicates that the line has been written once after its proactive SET; and '00' indicates that the line's status is not one of above cases.

The flag is initialized when the cache line is loaded from PCM memory. Based on the write-intensive page table, the controller can determine if a memory line is from a write-intensive page. The flag is set to '00' if the line is not from such a page. Otherwise, the flag f_1f_2 is set as fllows. If the memory line contains only 1st-write code, then the flag is set to '11' indicating the memory line has been written once after proactive-SET. If the line contains at least one 2nd-write code, then the flag is set to '00' indicating a proactive-SET is preferred before the next write.

Once a cache line becomes dirty, WoM-SET determines if a proactive-SET request should be sent to the proactive-SET queue in the memory controller. Proactive-SET request has lower priority comparing to normal read and write, and is sent only when a WoM-encoded line has flag '00'. In particular, if the flag is '11', the request is not sent as the corresponding memory line can be written another time before proactive-SET. For a non-WoM-encoded line, the request is sent unless the line has been set or is being set to all 1s, which is the same as that in the PreSET scheme.

When a WoM-encoded dirty line is expunged from LLC, WoM-SET encodes the line using the 2nd-write code if the flag is '00', or using the 1st-write code otherwise.

To migrate a page into and out of the WoM region at runtime, the controller flushes the page content from LLC, performs bridge chip-assisted page migration, and then rebuilds the cache flags when lines are loaded from the new location.

Address remapping. If a page is identified as a write-intensive page, its page index will be replaced by the index of its mapped block. Instead of sending the write request to the OS-mapped page in main data region, the request is sent to the reserved WoM region, which encodes the line with WoM code before updating it to the PCM memory.

Other issues. Since WoM-SET continuously maps write-intensive pages to the reserved PCM region, there is a concern whether the lifetime of this region may be significantly shortened, which fails the PCM memory prematurely. Since we reserve WoM region from the same memory address space, it can be seamlessly integrated with existing wear leveling schemes such as Start-Gap [12] and Security-refresh [20]. The memory lines from either reserved region or main region can be randomized freely, which helps to redistribute the writes across the whole device space.

WoM-SET uses the same offchip memory bus width even though WoM-encoded lines have 1.5× normal line size. For a memory access to write-intensive page, while its address translation is done within the onchip memory controller, the data encoding/decoding is done on PCM DIMM by a bridge chip [4]. It is the plaintext that is transmitted between CPU and memory chips. In addition, WoM-SET requires that the PCM row buffer should be at least twice of the last level cacheline size. This ensures proper column decoding when writing data in PCM memory.

4. EXPERIMENTAL METHODOLOGY

To evaluate the effectiveness of WoM-SET, we adopted the same simulation framework from [15] and compared our scheme to the baseline using *flip-n-write* and the PreSET scheme. The simulator is built as a trace-driven PIN tool. It faithfully models the entire memory hierarchy, including L1, L2 and DRAM last-level caches, and PCM main memory. The 32GB memory is organized as 32 banks, and each bank has a 32-entry write queue (WRQ) that buffers pending write requests. The memory controller gives a higher priority to read requests. A write request is scheduled only when there is no read request. When the write queue is full, the memory controller schedules a write burst. There is a 128-entry proactive-SET request queue (PSQ) in each bank to buffer the proactive-SET requests. Proactive-SET request has the lowest priority and can only be scheduled when the memory bank is idle. Both PreSET and WoM-SET adopt adaptive write cancellation [14].

The baseline configuration is shown in Table 3 and follows [15]. The system has eight single-issue in-order cores operating

at 4GHz. Each core has a 32MB private write-back DRAM cache and the cache line size is fixed to be 128B.

Processor	8-core, 4GHz, Intel atom-like cores
I/D-L1	private, 32KB/32KB, 4-way, LRU, 128B line size, write-back
L2	private, 2MB, 4-way, LRU 128B line size, write-back
DRAM Cache	private, 32MB, 8-way, LRU 128B line size, write-back
PCM Main Memory	32GB, 128B line size, 32 banks, 4 ranks, 8/32-entry read/write queues Read first, write burst when full 4000-cycle write, 500-cycle read SET: 4000 cycles [15], 90μW/bit, 13.5pJ/bit [11] RESET: 500 cycles [15], 480μW/bit, 19.2pJ/bit [11]

Table 3: Baseline Configuration

The simulated workloads are summarized in Table 4. We selected six representative memory intensive benchmarks from SPEC2006 suite. Their traces were interleaved to create three multi-programmed workloads. We skipped the warm up phase of each workload, and ran five billion instructions to collect the results.

Workload	Description	RPKI	WPKI
zeusmp	SPEC-CPU2006, 8 copies	3.76	0.97
astar	SPEC-CPU2006, 8 copies	4.44	2.32
leslie3d	SPEC-CPU2006, 8 copies	4.41	1.69
bwaves	SPEC-CPU2006, 8 copies	3.86	2.35
lbm	SPEC-CPU2006, 8 copies	4.55	2.42
cactusADM	SPEC-CPU2006, 8 copies	1.46	0.67
mix1	zeu, bwa, cac, ast; $\times$2 each	3.11	1.37
mix2	les, lbm, zeu, asta $\times$2 each	4.28	1.74
mix3	bwa, cac, les, lbm; $\times$2 each	3.26	1.59

Table 4: Simulated Worloads

5. RESULTS ANALYSIS

In this paper, we studied the following schemes.

- Baseline — The baseline has no proactive-SET operation. It adopts *flip-n-write* and needs both SET and RESET for most write requests.
- PreSET — It implements the PreSET scheme [15], together with *flip-n-write* and *write-cancellation*.
- WoM-SET — It is the scheme we propose in this paper, and is applied only to write-intensive pages. Since WoM-SET performs one proactive-SET operation for every other write to the same line, there are two types of writes:

  i) WoM-SET-1 represents the write using the 1st-write code. It consists of a proactive-SET (during bank idle interval), and a RESET-only *write-back* write.

  ii) WoM-SET-2 represents the *write-back* write that writes to a line the second time after its proactive-SET. It only updated changed bits to the line, and these changed bits are encoded using the 2nd-write WoM code.

Hardware cost. We added 3 bits per LLC cacheline, which corresponds to 96KB for 32MB DRAM cache, or 0.3% space overhead. The table to track write-intensity is about 192KB (=32K$\times$(30+11)/8). It takes 10ns to access the table. Since the access is in parallel to LLC access, the access overhead is negligible. The reserved PCM region is 256MB, or 0.8% of the 32GB PCM memory. To summarize, WoM-SET has very modest hardware cost.

Bit changes. Figure 5 compares the average number of bits to be RESET for different schemes. The baseline and PreSET schemes require 76 and 198 bits to be RESET respectively. WoM-SET on average requires 120 bits to be REEST, representing 40% improvement over the PreSET scheme. From the figure, we can see that WoM-SET-2 is close to the baseline, even though the write is updating a wider line (1.5$\times$ line size). This confirms that differential update greatly reduces bits needed to be RESET. From the figure, the average of WoM-SET-1 is smaller than that of the PreSET scheme even though both schemes do RESET from all 1s. This is because for the 1st-write WoM code, only three code has one 0 each, while for the plaintext, two codes have one 0 each ('01' and '10') and one has two 0s ('00'). Thus the PreSET scheme is likely to RESET more bits (even with *flip-n-write*).

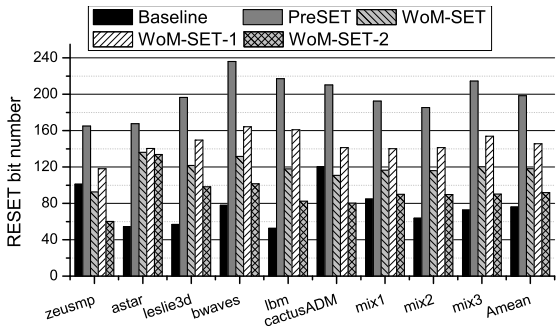


Figure 5: The RESET bit number comparison.

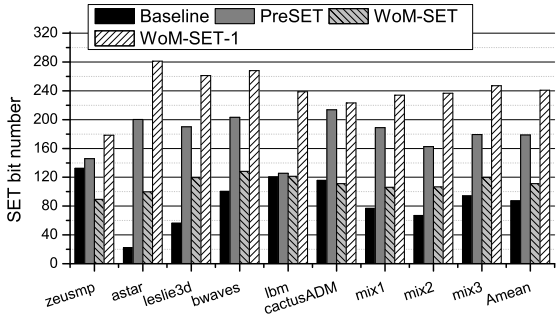


Figure 6: The SET bit number comparison.

Figure 6 compares the average number of bits to be SET per write. WoM-SET only sets bits for WoM-SET-1 type of writes. On average, WoM-SET-1 has the largest number because its line size is larger and it contains 0s produced from two consecutive writes. However, the number of WoM-SET-1 writes is roughly half of that in the PreSET scheme. Thus, WoM-SET achieves a lower average per write request — it exhibits 38% SET number reduction.

Write power. Figure 7 compares write power of different schemes. For the PreSET and WoM-SET schemes, we only consider the power of *write-back* writes as *proactive-SET* is a separate operation that is done when the memory bank is idle, and the power of *proactive-SET* is low comparing to that of write-back write. From the figure, WoM-SET achieves 40% write power reduction over the PreSET scheme. This is mainly due to the reduction of bits needed to be RESET.

System power and performance. Figure 8 compares system power. The results are normalized to the baseline. Due to proactive-SET operations, the PreSET scheme consumes larger read and processor power than the baseline. In addition, WoM-SET needs to track/encode/decode write-intensive pages, and thus consume 1% more processor power over the

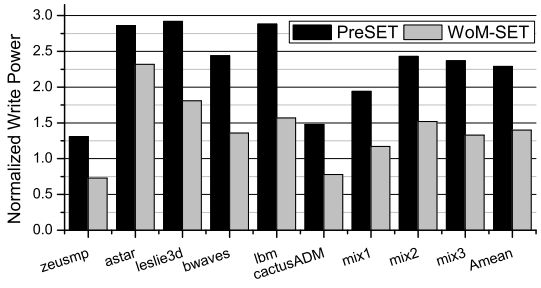


Figure 7: Write power comparison.

PreSET scheme. Due to reduced write power, WoM-SET consumes about 17% less system power than the PreSET scheme.

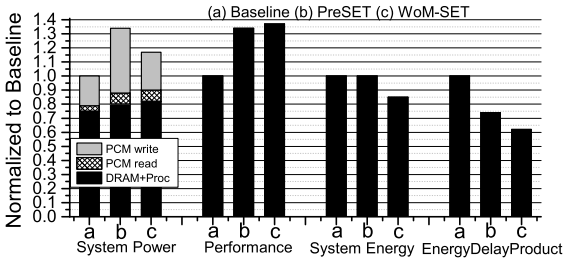


Figure 8: Performance comparison.

Figure 8 also compares the performance of different schemes. On average, the PreSET scheme achieves 34% performance improvement over the baseline. WoM-SET needs to track/ encode/decode write-intensive pages, which incurs around 2% overhead for our workloads. Since WoM-SET reduces the number of proactive-SET operations, it occupies less band bandwidth and has less interference on performance critical read operations. Overall, WoM-SET achieves slightly better performance than the PreSET scheme (especially for the workloads that access memory banks more often such that their banks have less spare time for proactive-SET).

To evaluate the overall effectiveness of WoM-SET with emphasis on both energy and performance, we reported the energy-delay-product in Figure 8. On average, WoM-SET achieves 12% EDP improvement over the PRESET scheme.

Fast write coverage. Figure 9 compares the percentage of write-back writes that can be completed with RESET-only operations. A proactive-SET, due to its low priority, may not be able to finish when its memory line is written back from cache. Such a write-back write needs both SET and RESET operation and thus is slow. The more the lines can be completed with RESET-only operations, the better the system performance is. From the figure, WoM-SET improves the percentage from 84% to 93%, which reduces around half of proactive-SET operations.

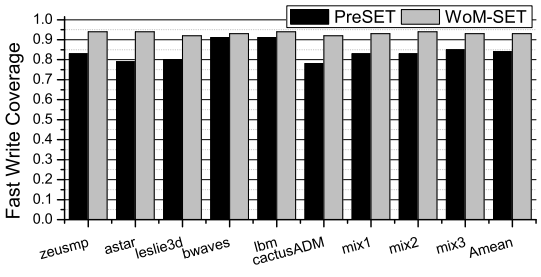


Figure 9: Percentage of fast writes.

6. CONCLUSIONS

In this paper, we present WoM-SET, a low power proactive-SET based write scheme for performance improvement. By encoding write-intensive pages in WoM code, WoM-SET allows two RESET-only writes after each proactive-SET, which reduces memory bandwidth demand, bit changes, and write power over the PreSET scheme. Our experimental results show that WoM-SET achieves on average 40% write power reduction and 12% energy-delay-product improvement.

In our future work, we will evaluate WoM-SET using large write-intensive applications. In particular, we will study if 256MB WoM region is sufficient and devise anti-thrashing mechanisms if write-intensive pages are migrated to and from the region too frequently. Recent works showed that there are negative impacts on bit error rate (BER) when adopting WoM code on flash memory [22]. The impact on PCM memory is yet to be studied at the device level.

7. REFERENCES

[1] S. Chhabra and Y. Solihin, "i-NVMM: A Secure Non-Volatile Main Memory System with Incremental Encryption," in *ISCA*, 2011.

[2] S. Cho and H. Lee, "Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance," in *MICRO*, 2009.

[3] Y. Choi, *et al.*, "A 20nm 1.8V 8Gb PRAM with 40MB/s Program Bandwidth," in *ISSCC*, 2012.

[4] K. Fang, *et al.*, "Memory Architecture for Integrating Emerging Memory Technologies," in *PACT*, 2011.

[5] G. D. Forney, "Coset Codes. I. Introduction and Geometrical Classification," in *IEEE Trans. Info. Theory, 34(5)*, 2009.

[6] ITRS, "The International Technology Roadmap for Semiconductors Report 2009," , 2009, `http://www.itrs.net/`.

[7] A. N. Jacobvitz, *et al.*, "Coset Coding to Improve the Lifetime of Memory," in *HPCA*, 2013.

[8] A. Jiang, "On the Generalization of Error-correcting WOM Codes," in *ISIT*, 2007.

[9] L. Jiang, *et al.*, "Improving Write Operations in MLC Phase Change Memory," in *HPCA*, 2012.

[10] K. Kim and S. J. Ahn, "Reliability Investigations for Manufacturable High Density PRAM," in *IRPS*, 2005.

[11] B. C. Lee, *et al.*, "Architecting Phase Change Memory as a Scalable DRAM Alternative," in *ISCA*, 2009.

[12] M. K. Qureshi, *et al.*, "Enhancing Lifetime and Security of PCM-based Main Memory with Start-Gap Wear Leveling," in *MICRO*, 2009.

[13] M. K. Qureshi, *et al.*, "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," in *ISCA*, 2009.

[14] M. K. Qureshi, *et al.*, "Improving Read Performance of Phase Change Memories via Write Cancellation and Write Pausing," in *HPCA*, 2010.

[15] M. K. Qureshi, *et al.*, "PreSET: Improving Read Write Performance of Phase Change Memories by Exploiting Asymmetry in Write Times," in *ISCA*, 2012.

[16] B. Rajendran, *et al.*, "Analytical Model for Reset Operation of Phase Change Memor," in *IEDM*, 2008.

[17] S. Raoux, *et al.*, "Phase-change Random Access Memory: A Scalable Technology," *IBM J. RES. & DEV.*, 2008.

[18] R. L. Rivest and A. Shamir, "How to Reuse a Write-Once Memory," *InC*, 1982.

[19] S. Schechter, *et al.*, "Use ECP, not ECC, for Hard Failures in Resistive Memories," in *ISCA*, 2010.

[20] N. H. Seong, *et al.*, "Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-change Memory with Dynamically Randomized Address Mapping," in *ISCA*, 2010.

[21] E. Yaakobi, *et al.*, "Multiple-write WOM-codes," in *Allerton*, 2010.

[22] E. Yaakobi, *et al.*, "Error Characterization and Coding Schemes for Flash Memories," in *ACTEMT workshop*, 2010.

[23] D. H. Yoon, *et al.*, "FREE-p: Protecting Non-volatile Memory against Both Hard and Soft Errors," in *HPCA*, 2011.

[24] P. Zhou, *et al.*, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in *ISCA*, 2009.