TriState-SET: Proactive SET for Improved Performance of MLC Phase Change Memories

Xianwei Zhang*, Youtao Zhang* and Jun Yang†

*Computer Science Department, University of Pittsburgh, Pittsburgh, PA, USA
{xianeizhang, zhangyt}@cs.pitt.edu
†Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, PA, USA
juy9@pitt.edu

***Abstract*—The emerging Phase Change Memory (PCM) has many advantages such as good scalability and low leakage. MLC (Multi-Level Cell) PCM further extends the benefits by storing two or more bits per cell and thus reducing the per bit cost. However, adopting MLC PCM in main memory often leads to long write latency, high energy consumption, and degraded performance. In this paper, we propose TriState-SET, a proactive-SET based write strategy for improving MLC PCM write performance. TriState-SET proactively places device cells of a dirty memory line in full SET state. By utilizing only three states of 2-bit MLC PCM, TriState-SET involves only fast state transitions when writing such a line at write-back time. Our experimental results show that TriState-SET increases performance by 11% and saves system energy by 6.7% (up to 12.2%), while achieving up to 25% (average 14.1%) energy-delay-product improvement.**

I. Introduction

Phase Change Memory (PCM) has recently emerged as a promising memory technology, which is projected to replace a significant portion of DRAM in future main memory subsystems. A single-level cell (SLC) PCM cell stores one bit information using two resistance states. The resistance state can be switched with RESET (writing '0') or SET (write '1') operations. A multi-level cell (MLC) stores two or more bits using more resistance states. While MLC PCM can achieve large capacity with low per bit cost, it has limitations. MLC PCM widely adopts binary search [9], [23] for read operations and iterative program-and-verify [3], [15] for write operations. As a result, a MLC PCM based memory system often suffers from long access latency and degraded system performance.

PCM exhibits significant write asymmetry, e.g., RESET operation is much shorter than SET operation in SLC PCM. Qureshi *et al.* [19] and Zhang *et al.* [24] proposed to exploit this asymmetry by proactively setting all cells of a memory line when its copy in cache becomes dirty. Only fast RESET operations are needed when writing such a line back to memory at write-back time, which shortens the write latency, speeds up the application execution, and reduces the system energy consumption.

However, even though the different values/resistance states of MLC PCM show significant access asymmetry, it is challenging to apply proactive SET to MLC PCM. This is because many state transitions require long latency writes, e.g., the transition from '00' to '10' and the transition from '10' to '01'. Proactively setting a memory line to any state cannot completely eliminate long latency transitions and thus has little impact on the overall write latency.

The MLC PCM access asymmetry has been exploited in the literature in the ways that are orthogonal to the proactive SET approach. Qureshi *et al.* [18] proposed to prioritize read operations by pausing the long latency MLC write operations. Hoseinzadeh *et al.* [9] proposed to speed up read intensive lines by storing them in the MSB (most significant bit). Yoon *et al.* [23] proposed to further speedup write accesses by exploring the write latency difference of the MSB and LSB (least significant bit) of MLC PCM. Wang *et al.* reduced the write energy by encoding a line with more '00' and '11' because writing them is often faster and consumes less energy than writing '01' or '10' [22]. Jiang *et al.* applied write truncation [12] to use ECC to save the time to write *difficult-to-write* cells, e.g., cells to be written to '01' and '10' values.

In this paper, we propose TriState-SET, a proactive-SET based write strategy for improving MLC write performance. TriState-SET adopts the two-way program-n-verify (P&V) write strategy [13], and utilizes only three states of 2-bit MLC PCM. By proactively placing device cells of a dirty memory line in full SET state, TriState-SET only involves fast state transitions when writing such a line at write-back time, which greatly reduces the write latency, and improves system performance and energy-delay-product (EDP). We evaluate the proposed scheme in the paper. The results show that TriState-SET improves the performance by about 11% (up to 18.4%) and achieves 14.1% EDP enhancement on average.

The rest of the paper is organized as follows. We introduce PCM basics in Section II. Section III elaborates the design motivation and the architectural details. Section IV presents the experimental methodology and Section V discusses the results. We conclude the paper in Section VI.

II. Background

PCM is an emerging non-volatile memory technology that stores data using phase changing materials (e.g., GST). In PCM, logic bits are represented by different resistance states,

[1]This work was funded in part by the National Science Foundation under grants CCF-1422331 and CNS-1012070.

e.g., amorphous state and polycrystalline state for SLC PCM. The stored bit can be changed by heating the material to transit between states: to RESET (1→0) a cell, a pulse with large magnitude and short duration is applied to melt the material to amorphous state; to SET (0→1) a cell, a pulse with small magnitude and long duration is applied to crystallize the material.

Due to the large resistance difference between RESET and SET states, MLC PCM can utilize the intermediate resistance states to store multiple bits per cell [3], [14], [15]. We consider 2-bit MLC in the paper. Compared to SLC, MLC PCM represents each state with a narrower resistance range, and thus requires more precise access control.

MLC programming strategy. Writing a MLC PCM cell widely adopts program-and-verify [3], [15], [13] strategies. With R2S P&V strategy, writing a MLC PCM line starts with a RESET pulse to place all to-be-written cells to full amorphous/RESET state, and then a sequence of SET pulses to place each individual cell to appropriate state. With S2R P&V strategy, writing a MLC PCM line starts with a SET pulse, which is followed by a sequence of RESET pulses. The recently proposed two-way P&V strategy [13] does not need to place all to-be-written cells to similar initial states. Instead, it injects RESET or SET pulses to program a cell, based on the current and the target resistant states. [13] shows that the two-way P&V strategy achieves better write performance for MLC PCM. In this paper, we adopt the two-way P&V strategy.

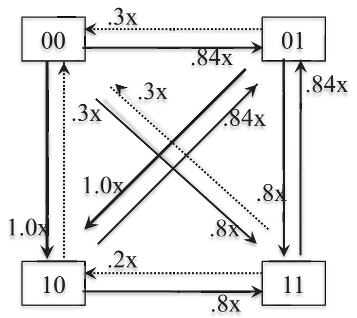


Fig. 1: The state transition latency for 2-bit MLC [13], [23] (normalized to 00→10 program latency). The two bits ‘XY’ for each state are MSB/LSB bits, respectively.

Figure 1 illustrates the programming latency for 2-bit MLC PCM when adopting the two-way P&V strategy [13], [23]. The latency is normalized to the ’00’→’10’ transition latency. Yoon *et al.* [23] observed that the MSB bit is more difficult to switch in general.

MLC asymmetry. MLC PCM write, no matter which P&V strategy is used, exhibits significant asymmetry in writing different values. For two-way P&V, ‘00’→‘11’ takes more than twice amount of time of ‘11’→‘00’ (as shown in Figure 1). For R2S P&V, writing ‘00’ finishes after one RESET pulse while writing ‘01’ needs several additional SET pulses to finish [12]. To ensure write reliability at memory line level, MLC PCM write needs to accommodate to the slowest state transition in the line, which leads to long write latency and degraded memory performance.

Existing schemes cannot eliminate long MLC PCM write in general. Write pausing [18] prioritizes reads but writes are still slow. Write truncation [12] only eliminates excessive long cell writes while many cell writes are still slow. Encoding a memory line with more ’00’/’11’ states [22] reduces write energy but not the write latency. Storing write intensive lines using LSB bits only reduces the write latency by 16% on average [23]. Storing read intensive lines using MSB bits reduces read latency but not the write latency [9], [23].

Recent studies proposed using three levels of 2-bit MLC to improve lifetime [11], or to address the resistance drift [21].

Proactive SET and cell encoding. For SLC PCM, PreSET [19] proactively places a memory line in full SET state such that a write-back write needs only fast RESET operations, which reduces the write latency and improves the system performance. WoM-SET [24] exploits WoM (write-once memory) code to reduce the number of RESET operations. Different encoding schemes have also been proposed for extending PCM lifetime [10] and reducing write energy [11].

III. The TriState-SET Scheme

In this section, we motivate the proactive SET in MLC PCM, and then discuss how TriState-SET works and present the architectural enhancement to enable TriState-SET.

A. Proactive SET in MLC PCM

Based on the state transition in Figure 1, a proactive SET based MLC write strategy can be devised as follows:

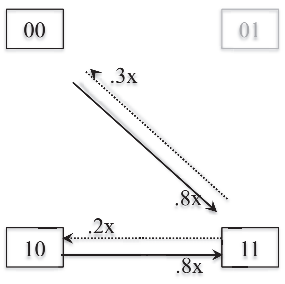
(i) We first discard the ‘01’ state for 2-bit MLC PCM so that each cell only uses three states (i.e., ‘00’, ‘10’ and ‘11’ states). Similar to that in [11], [21], we store 3-bit information using two three-state cells. This is possible because two 3-state cells can represent 9 combinations in total while encoding 3 bits needs only 8 codes. As an example, the code for value ‘011’ is ‘10.11’ in Figure 2(a), indicating that the two cells should be in ‘10’ and ‘11’ states, respectively.
(ii) We then proactively SET all cells in a PCM line to ‘11’ state if it becomes dirty in cache. When there is a need to write back such a line to memory, the write only involves two possible transitions, i.e., ‘11’→‘00’ and ‘11’→‘10’. Both transitions are fast as shown in Figure 2(b).

Figure 2(a) also shows the latency to change ‘111’ (encoded as ‘11.11’) to other values. From the figures, the latency is bound by 0.3×, i.e., within 0.3 times the latency of the ‘00’→‘10’ transition.

The example in Figure 3 illustrates why proactive SET in MLC PCM is beneficial. Assume we need to write 6-bit information ‘01 01 10’ to the MLC PCM memory, the original data in memory is ‘01 11 00’, and L is the latency of ‘00’→‘10’ transition. The cache line is updated to the new data at time ❶ and is written back at time ❷. The baseline scheme (Figure 3(a)) stores the data using three 2-bit MLC cells. The actual write starts at time ❷ and takes L to finish due to the

Value	2-cell encoding (C1.C2)	Latency (11.11→ C1.C2)
000	00.00	0.3x
001	00.10	0.3x
010	10.10	0.2x
011	10.11	0.2x
100	11.00	0.3x
101	11.10	0.2x
110	00.11	0.3x
111	11.11	–

(a) Store 3 bits with 2 cells



(b) 3-state cells.

Fig. 2: TriState cells eliminate slow '11'→'01' transition.

long latency '00'→'10' transition. A naive implementation of proactive SET (Figure 3(b)) stores the data using the same number MLC cells. It proactively sets the memory line to '11' after time ❶ such that it takes 0.84×L to finish at time ❷. This is because the slowest transition at time ❷ is '11'→'01'.

With our proposed proactive SET scheme (Figure 3(c)), the data is encoded using four 3-state MLC cells. Since it eliminates the slow '11'→'01' transition, we expect 70% write latency deduction at time ❷. Similar to the naive scheme, the memory line is proactively SET when its copy in cache becomes dirty and the memory bank is idle. Since this is done off the critical path, it has little performance impact.

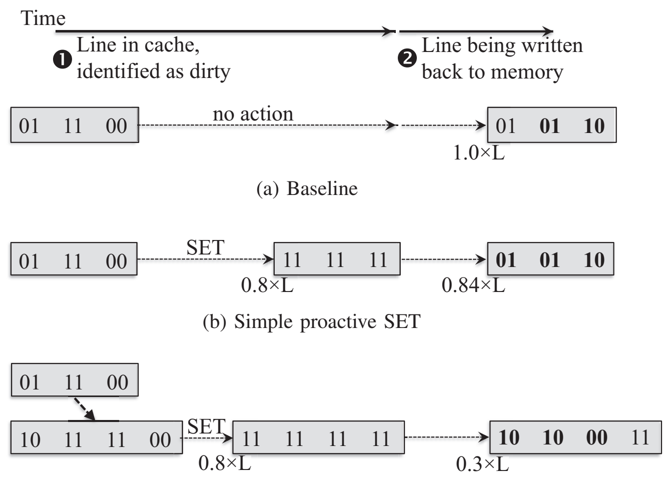


Fig. 3: A motivational example. (a) Baseline. No proactive SET operation, write latency is 1.0×L; (b) Simple proactive SET. Proactively SET dirty cache lines, lower latency of 0.84×L; (c) TriState SET. Encode data using 3 states and proactively set dirty cache lines, 0.3×L write latency.

B. The TriState-SET Scheme

The above discussion demonstrates that our 3-state based proactive SET can greatly improve the write performance. However, it has two issues.

- Space overhead. Since two 2-bit MLC cells stores three bits instead of four bits in the baseline, there is a 33% space overhead.
- Proactive SET dependency. The fast state transition is possible only when the corresponding line has been proactively SET. If proactive SET is not done at write-back time, we cannot enable fast write mode.

To address these issues, we propose TriState-SET that adopts FPC (frequent pattern compression) [1] to compress the memory line and switch between fast and slow write modes based on whether, at write-back time, the new data is compressible and the memory line has been SET.

Given a dirty cacheline to be written to PCM memory, Figure 4 illustrates the three write modes in TriState-SET:

(i) TriState-SET performs the baseline write (1.0×) if the memory cells have not been proactively SET successfully. This includes the case that the proactive SET request has not been sent, and the case that the request has been sent but the operation has not been done yet. With the line being proactively SET, any state transition (as shown in Figure 1) is possible,. The write latency is the same as the baseline.

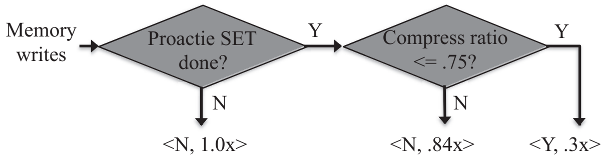


Fig. 4: The three write modes in TriState-SET.

(ii) TriState-SET performs the fastest write (0.3×) if the memory line has been proactively SET and the new data can be compressed to 75% or less of its original size. The compressed data is then encoded with 3-state MLC cells. TriState-SET spends 70% less time than the baseline to store the encoded line to the memory.

(iii) TriState-SET performs only slightly faster (0.84×) than the baseline if the memory line has been proactively SET, but the new data is not compressible. Without compression and encoding, the slow '11'→'01' transition may still exist, which places a 0.84× bound on the write latency.

C. The Architectural Designs

An overview of TriState-SET architecture is shown in Figure 5. We integrate TriState-SET in the memory controller and enhance the last level cache to enable TriState-SET.

- We add a two-bit flag $f1f2$ to each cacheline in the last level cache to indicate its status in device memory. $f1$ is set to '1' if a proactive SET operation has been sent to the memory controller. $f2$ is set to '1' if the proactive SET operation has finished successfully.
- We add a two-bit flag $f2f3$ to each entry in the write queue WRQ. $f2$ is the same as the one attached to cacheline, i.e., an evicted cacheline sends both its data and $f2$ flag to the write queue. $f3$ indicates if the data can be compressed to 75% or less of its original size.
- We add a simple PSQ (proactive SET queue) to buffer the proactive SET requests. PSQ stores only the addresses of

the memory lines to be proactively SET. Without storing data, PSQ is much smaller than a write queue with the same number of entries.

- We add FPC [1] compression and TriState encoding logic in the memory controller and one bit flag $f4$ to each memory line to indicate if the line is stored in compressed/encoded format.

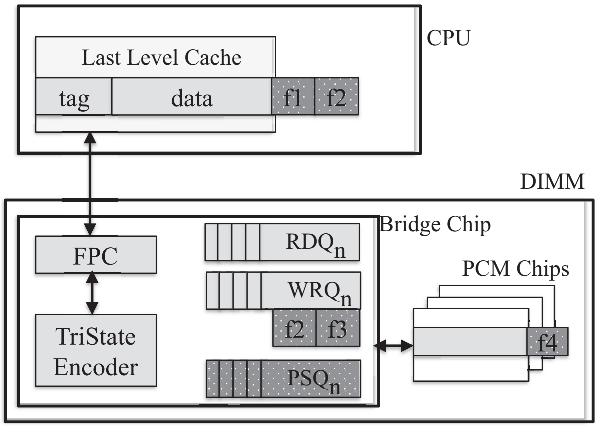


Fig. 5: The TriState-SET architecture (dark shaded parts are added ones).

When a cacheline becomes dirty, both the line address and its $f1$ bit are sent to PSQ. The $f2$ is cleared indicating that the proactive SET is in progress. When the memory bank is idle, the controller issues proactive SET requests for the addresses in PSQ. Retiring an entry from PSQ sets the $f2$ bit of the corresponding cacheline.

When a dirty cacheline is evicted from the last level cache, its data and $f2$ are sent to WRQ. The line address is cleared from PSQ if it is still in the queue. If the cacheline is compressible and $f2$ is set, the controller sets the $f3$ bit and saves the encoded data in WRQ. For the lines in WRQ, the controller issues a normal write to the memory module if $f2$ is ‘0’; a semi-fast write if $f2f3$ is ‘10’; and a fast write if $f2f3$ is ‘11’.

When loading a memory line from the device, the $f4$ flag bit is read simultaneously. If the bit is set, the loaded line is decoded and decompressed before being stored in the last level cache.

Architectural impact. The proposed architecture enhancement has minimized impact on existing hardware. Our baseline architecture adopts the bridge chip design [4] that integrates a bridge chip on the NVM DIMM to enforce the timing control of each read and write operation. Fang et al. showed that this design helps to maximize the bandwidth utilization when exploiting the non-uniform NVM memory access latencies. In our design, the $f2$ flag is communicated using the “data ready” line between DIMM and onchip memory controller [4]. The latency difference of different write modes is handled by the bridge chip controller. The FPC compression unit was originally proposed for on-chip data compression. Recent PCM studies showed that adopting FPC for PCM compression has negligible performance and area overheads [12].

IV. Experimental Setup

A. *Configuration*

To evaluate the effectiveness of our proposed design, we used a Pin-based in-house cycle-accurate simulator that faithfully models the whole memory hierarchy, including L1, L2, DRAM last-level caches and PCM main memory. The system is configured as an in-order 8-core CMP. Similar to those in [19], [9], the cache line size is set as 128B.

The main memory is organized as 4 ranks with 8 banks each. For the baseline, each bank has a separate 8-entry read queue (RDQ) and 32-entry write queue (WRQ). Memory controller schedules the requests by giving a higher priority to reads: read request is always serviced unless the write queue reaches 80% full, or there is no pending reads (i.e., the RDQ is empty). The MLC PCM read and write latency are set as 250ns and 2μs [23], [18], respectively. Detailed settings can be found in Table I. For Proactive SET based schemes, each bank has an extra proactive SET queue (PSQ) of 128 entries. Since Proactive SET operations take use of the bank idle time, the requests are only issued when both RDQ and WRQ are empty. In addition, as [19], read requests can cancel proactive SET operations.

TABLE I: System Configuration

Processor	8-core, 4GHz, single-issue in-order
Cache	linesize: 128B L1(private): 32KB per core, 2-way, 1 cycle L2(private): 512KB per core, 4-way, 10 cycles DRAM write buffer [18], [17]: 16MB/core, 8-way, 100 cycle
Main Memory	32GB MLC PCM, 4 ranks of 8-banks each Queues: per bank, entries: RDQ-8, WRQ-32, PSQ-128 scheduling: prioritize reads if WRQ<80% full
Memory Latencies	normal read: 250ns (1000 cycles) normal write: 2μs (1x, 8000 cycles) preset: 1.6μs (0.8x, 6400 cycles) fast write(base): 1.68μs (0.84x, 6720 cycles) fast write(code): 0.6μs (0.3x, 2400 cycles)

B. *Workloads*

To evaluate the schemes, we ran several typical benchmarks which consists a subset of workloads from the MediaBench II video benchmark suite [5], MiBench [6], STREAM[7], and SPEC2006 benchmark suite. Table II lists the read and write MPKI for each workload. While most are memory intensive benchmarks, we include some non-intensive ones, such as $qsort$ and $jpg2000$ for comparison purpose. The benchmarks are executed in rate mode, with 1 billion instructions simulated after skipping the warmup phase.

V. Results

In this paper, we evaluated the following schemes:

— `WP`. This is the baseline scheme. It integrates the write pausing scheme [18]. The cell writes follow the transition graph in Figure 1, i.e., the write latency is 1.0×.

TABLE II: Simulated Benchmarks

Name (Abbr.)	Description	RPKI	WPKI
astar (ast)	SPEC-CPU2006	6.12	4.05
GemsFDTD (Gem)	SPEC-CPU2006	11.09	5.85
leslie3d (les)	SPEC-CPU2006	3.20	1.24
zeusmp (zeu)	SPEC-CPU2006	1.65	0.75
jpg2000 (jpg)	MediaBench II	0.51	0.12
qsort (qso)	MiBench	0.86	0.51
stream (str)	STREAM	17.24	6.34

— WP+P_SET. This scheme integrates naive proactive SET (as described in Section III-A) in the baseline. If a memory line has been proactively SET at write back time, the write latency is 0.84×; otherwise, the write latency is 1.0×. This also approximately simulates the write effect of the scheme proposed in [23], which speeds up most writes by 16%.

— WP+T_SET. This scheme integrates TriState-SET in the baseline. As described in Section III-B, the write latency depends on the write mode that the hardware chooses at write-back time. It may be 0.3×, 0.84×, or 1.0×.

Next, we compare the schemes on memory access latency, system performance and energy, and study their sensitivity to different configurations.

A. Hardware Overhead

The timing overhead comes from FPC compression and TriState cell decoding. We charged 2ns [1] in the simulation and observed less than 0.1% performance degradation, i.e., the timing overheads is also negligible. FPC compression is a lightweight compression scheme that was originally designed for cache compression. Our results confirmed previous studies [1], [11], [12] that integrating FPC compression in PCM introduces negligible hardware overhead. For other hardware overhead in TriState-SET, we add one bit to each 128B memory line, two bits to each LLC line, a 128-entry PSQ queue, and an eight-entry code book. The space overhead is less than 0.1% of the main memory and 0.2% of the LLC. The space overheads of PSQ and code book are negligible.

B. Memory Read Latency

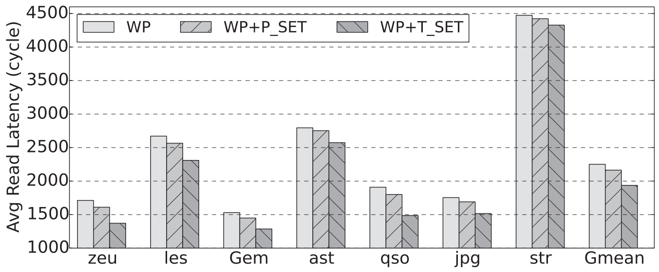


Fig. 6: The comparison of memory read latency.

System performance heavily depends on the memory read access latency. Figure 6 shows the average memory read latency for different schemes. The results are normalized to the WP and the geometric mean of all evaluated workloads is presented as well. The figure shows that for WP, all workloads, except *Gem*, have an effective read latency over 1600 cycles, indicating that there's severe memory contention. Compared to other workloads, *str* shows a much higher latency because the workload is highly memory access intensive. With WP+P_SET, the average latency is reduced from 2251 to 2163 cycles, which is a small 4% reduction. WP+P_SET is less effective since slightly reducing write latency to 0.84× provides little help, especially after adopting write pausing, which reduces the impact of long write latency on read operations.

From the figure, WP+T_SET achieves a 14% reduction due to its significant lower write latency at write-back time. On average, the memory read latency is reduced to 1936 cycles, indicating that WP+T_SET effectively alleviates the blocking effect of writes on reads.

C. System Performance

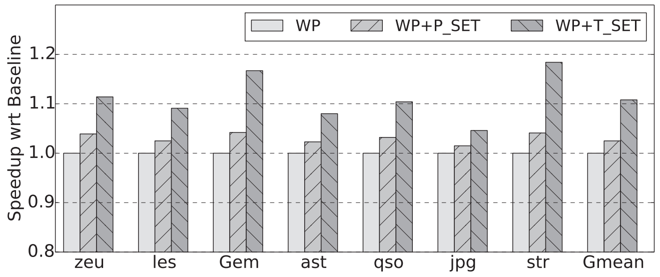


Fig. 7: The performance comparison of different schemes.

Figure 7 compares the system performance of different schemes. The reduction in read latency effectively translates into performance improvement. On average, while WP+P_SET slightly improves the performance by 2.5%, WP+T_SET achieves significant improvement of about 11%. Among the workloads, *str* shows the largest improvement of 18.4%, which is attributed to the memory access intensiveness in the workload. Compared to other benchmarks, *jpg* has many fewer accesses, resulting in a moderate speedup of 4.6%.

D. System Energy

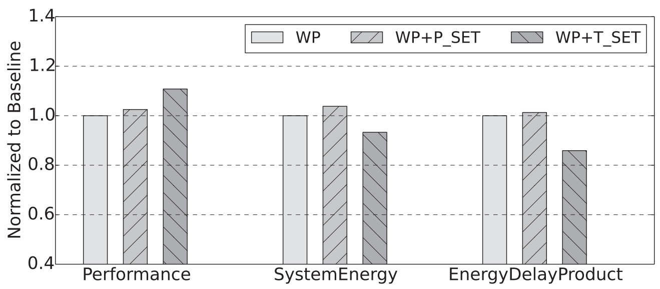


Fig. 8: The comparison of system energy.

Proactive SET based schemes introduce extra off-the-critical path SET operations, which brings extra state changes and further incurs extra power and energy consumption. Note that proactive SET operation does not increase the peak power consumption of the memory system, because SET is much more power efficient than RESET in PCM[11]. Figure 8 compares the system energy consumption, calculated following energy models in [20] and [19], under different schemes. Given that PCM write power accounts for 21% of the overall system power [19], and TriState-SET shortens the program execution time, which converts into lower system energy — on average, we observed 6.7% reduction on system energy consumption for `WP+T_SET`. With the performance gain and energy saving of `WP+T_SET`, we expect improvements on EDP, whose reduction is up to 25% (14.1%, on average).

From the figure, `WP+P_SET` increases the energy by 3.8%, on average, and slightly increases EDP by 1.3%. This is because `WP+P_SET` achieves only 16% write latency reduction (up to 70% for `WP+T_SET`), which has very limited effect on write pausing involved scenarios. Thus, the energy consumption reduction resulted from slightly improved performance cannot compensate the large loss due to extra write activities.

E. Fast Write Coverage

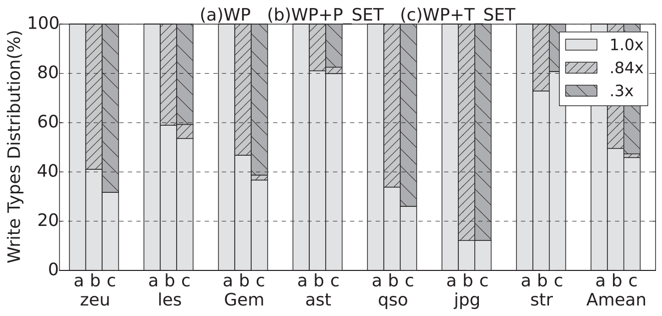


Fig. 9: The distribution of different writes: normal write (1.0×), semi-fast write (0.84×), and fast write (0.3×).

Different writes in `WP+P_SET` and `WP+T_SET` may have different write latencies. For example, a write operation needs to roll back to the baseline write if the proactive SET operation (to the device line) has not been done at the write-back time. Figure 9 summarizes the distribution of different writes — the baseline `WP` has only one type of writes (i.e., 1.0× mode), `WP+P_SET` has two types of writes (i.e., 1.0× and 0.84× modes), and `WP+T_SET` has three types of writes (i.e., 1.0×, 0.84×, and 0.3× modes). *Amean* bar in the figure shows the arithmetic mean of the evaluated workloads.

From the figure, on average, about 50% of writes are semi-fast writes in `WP+P_SET`. *jpg* achieves the highest coverage of 87.8%, which is contributed by the sparse memory accesses. For `WP+T_SET`, 52.7% writes are fast ones and less than 2% lines are semi-fast writes. The latter represents those lines whose memory line has been proactively set but cannot encode due to compression ratio being larger than 0.75.

[19] reported that 80% of SLC lines can be proactively set. This ratio is reduced to around 50% in MLC PCM. The reason is that proactive SET can only be done in memory bank idleness and the memory bank is less idle in MLC PCM due to longer MLC write latency. Another reason is that `WP` prioritizes reads and thus cancels many proactive SET at runtime.

F. Memory Lifetime

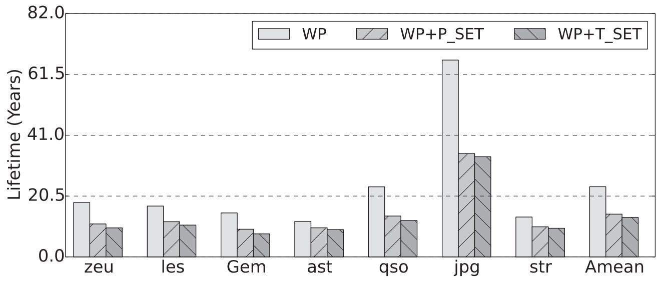


Fig. 10: The comparison of PCM lifetime.

Figure 10 compares the lifetime of MLC PCM based main memory when adopting different schemes. We followed the assumption in [20], [9], [23], [19] with good wear leveling designs [17], we expect our MLC PCM based memory subsystem has a lifetime over 9 years for all workloads except *Gem*. This meets the 4-year lifetime demand discussed in [19]. Comparing to `WP+P_SET`, `WP+T_SET`'s lifetime decreases by about 1 year (e.g., for *les*, lifetime is 11.9- and 10.7-year, before and after adopting TirState-SET). However, `WP+T_SET` finishes more instructions within the shorter lifetime. To compare the efficiency, we used the Instructions per Lifetime (IPL) metric [9], and found out that `WP+T_SET` and `WP+P_SET` shows almost the same IPL.

G. Sensitivity Studies

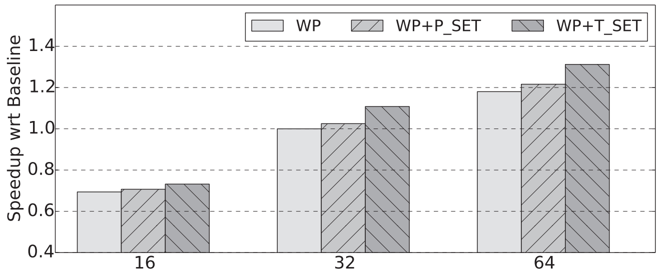


Fig. 11: The sensitivity on the number of banks.

1) Varying the Number of Banks: `WP+P_SET` and `WP+T_SET` rely on memory idle time to conduct proactive SET operations. Since memory idle time is closely related to bank level parallelism, we next evaluated the performance with different numbers of banks.

Figure 11 compares the system performance when we vary the number of PCM banks. Reducing the bank number from 32 (i.e., the number of banks in the baseline) to 16, we observed 30.6% and 29.3% performance degradation for `WP` and `WP+P_SET`, respectively. `WP+T_SET` only achieves 2.5%

performance improvement over WP+P_SET. As expected, we get better performance improvement when there are more memory banks.

2) Varying the Proactive SET Queue (PSQ) Size: Figure 12 compares the performance when varying the queue length from 32 to 512 entries. By default, we used a 128-entry PSQ for the above evaluation. The results show that the performance improvement is insensitive to PSQ length. Given a proactive SET may not be issued if PSQ is full, intuitively, a larger PSQ tends to have less contention and thus better performance. However, since proactive SET requests are issued only when the memory bank is idle and may be paused/cancelled by read operations, a large queue may not expose more opportunities.

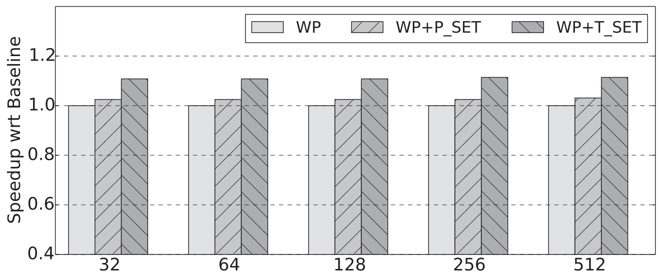


Fig. 12: The sensitivity on PSQ length.

VI. Conclusions

In this paper, we proposed TriState-SET, a proactive-SET based scheme to exploit the asymmetric write behavior of MLC PCM. By adopting 2-way P&V programming and 3-state MLC encoding, TriState-SET effectively reduces the write latency to 0.3× at write-back time. This helps to alleviate the blocking effect on read operations, and thus improves system performance. The experimental results shows that the performance improvement is 11% on average (up to 18.4%), with 6.7% (up to 12.2%) lower system energy and up to 25% (14.1%, on average) EDP enhancement.

References

[1] A. R. Alameldeen and D. A. Wood. Adaptive cache compression for high-performance processors. In *ISCA*, 2004.

[2] K. Albayraktaroglu, A. Jaleel, et al. Biobench: A benchmark suite of bioinformatics applications. In *ISPASS*, 2005.

[3] F. Bedeschi, R. Fackenthal, et al. A bipolar-selected phase change memory featuring multi-level cell stroage. In *JSSC*, 2009.

[4] K. Fang, L. Chen, et al. Memory architecture for integrating emerging memory technologies. In *PACT*, 2011.

[5] J. E. Fritts, F. W. Steilling, et al. Mediabench ii video: Expediting the next generation of video systems research. In *MICPRO*, 2009.

[6] M. R. Guthaus, J. S. Ringenberg, et al. Mibench: A free, commercially representative embedded benchmark suite. In *IISWC*, 2001.

[7] J. D. McCalpin. Stream: Sustainable memory bandwidth in high performance computers. http://www.cs.virginia.edu/stream/.

[8] T. Happ, M. Breitwisch, et al. Novel one-mask self-heating pillar phase change memory. In *VLSIT*, 2006.

[9] M. Hoseinzadeh, M. Arjomand, et al. Reducing access latency of MLC PCMs through line striping. In *ISCA*, 2014.

[10] A. N. Jacobvitz, R. Calderbank, et al. Coset coding to extend the lifetime of memory. In *HPCA*, 2013.

[11] L. Jiang, Y. Zhang, et al. ER: Elastic RESET for low power and long endurance MLC based phase change memory. In *ISLPED*, 2012.

[12] L. Jiang, B. Zhao, et al. Improving write operations in MLC phase change memory. In *HPCA*, 2012.

[13] M. Joshi, W. Zhang, et al. Mercury: A fast and energy-efficient multi-level cell based phase change memory system. In *HPCA*, 2011.

[14] D.-H. Kang, J.-H. Lee, et al. Two-bit cell operation in diode-switch phase change memory cells with 90nm technology. In *VLSIT*, 2008.

[15] T. Nirschl, J. Phipp, T. Happ, et al. Write strategies for 2 and 4-bit multi-level phase-change memory. In *IEDM*, 2007.

[16] M. K. Qureshi, M. M. Franceschini, et al. Morphable memory system: A robust architecture for exploiting multi-level phase change memories. In *ISCA*, 2010.

[17] M. K. Qureshi, M. M. Franceschini, et al. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *MICRO*, 2009.

[18] M. K. Qureshi, M. M. Franceschini, et al. Improving read performance of phase change memories via write cancellation and write pausing. In *HPCA*, 2010.

[19] M. K. Qureshi, M. M. Franceschini, et al. PreSET: Improving performance of phase change memories by exploiting asymmetry in write times. In *ISCA*, 2012.

[20] M. K. Qureshi, V. Srinivasan, et al. Scalable high performance main memory system using phase-change memory technology. In *ISCA*, 2009.

[21] N. H. Seong, S. Yeo, and H. S. Lee, Tri-level-cell phase change memory: toward an efficient and reliable memory system. In *ISCA*, 2009.

[22] J. Wang, X. Dong, et al. Energy-efficient multi-level cell phase-change memory system with data encoding. In *ICCD*, 2011.

[23] H. Yoon, J. Meza, et al. Efficient data mapping and buffering techniques for multi-level cell phase-change memories. *ACM TACO*, 2014.

[24] X. Zhang, L. Jiang, et al. WoM-SET: Low power proactive-SET-based PCM write using WoM code. In *ISLPED*, 2013.

[25] P. Zhou, B. Zhao, et al. A durable and energy efficient main memory using phase change memory technology. In *ISCA*, 2009.