Reducing Read Latency in MLC PCM

Jianhui Yue
Dept. of Computer Science & Software Engineering
Miami University
Oxford, OH 45056
yuej2@miamioh.edu

Yifeng Zhu
Dept. of Elec. & Computer
University of Maine
Orono, ME 04469
yifeng.zhu@maine.edu

Abstract—**Multi-level-cell (MLC) phase-change memory (PCM) provides higher storage density at the cost of slower reads and writes. Since reads are latency critical, this paper propose a simple and effective bit mapping scheme, called Mapping Critical Word to MSBs (MCWM), to address slow reads in MLC. MCWM takes advantage of fast read speed of most-significant-bits (MSBs) of MLC cells and strips a cache line among MLC cells at the bit level. Taking 2-bit MLC as an example, MCWM stores the first half of each cache line at most-significant-bits (MSBs) of MLC cells, and the second half at least-significant-bits (LSBs). This design leverages the observation that most critical words are located within the first half of a cache line. Upon a cache miss, the critical word can be fetched at the same speed as single-level cell (SLC) PCM, thus reducing processor stall time. Experimental results under 4-cores SPEC CPU 2006 workloads show that MCWM can reduce memory read latency by 27.5% and IPC by 13.7% on average, compared with conventional PCM. In addition, MCWM outperforms recently proposed Striped PCM (SPCM) by 12.5% in latency and 6.1% in IPC on average. Additionally, MCWM is complementary to write optimizations. MCWM can reduce read latency of Write Pause by 25% and increase IPC by 11.7% on average.**

I. Introduction

Better process scalability, none-volatility, and less leakage power make Phase-Change Memory (PCM) a promising alternative or supplement to DRAM. Applying varying levels of currents to phase-change material, such as Ge2Sb2Te5(GST), can transform the material into either crystalline and amorphous states that have dramatically different electrical resistance. In a single-level cell (SLC), the resistance level of the cell is used to represent a binary bit. Low resistance and high resistance represent 1 and 0 respectively. This resistance-based memory is proved to be very scalable. PCM consumes much less leakage current and requires no refresh operations due to its non-volatile property.

In order to increase storage density, multi-level cell (MLC) PCM stores two or more binary bits in a MLC cell. It trades the read and write performance for higher bit density. When data bits are written to a PCM cell, a repetitive program-and-verify mechanism is performed in order to precisely set the resistance of the target cell to some desired value. In addition, n iterations of sensing have to be carried out during reading a cell that stores n binary bits. One bit is read out from a MLC cell in each iteration. Many research works have focused on the slow write issue of MLC [1]–[3]. However, less attention has been drawn to the issue of slow read.

This paper proposes a new and simple bit mapping scheme, called Mapping Critical Word to MSBs (MCWM), to reduce the read latency of MLC. Our key idea is to store critical word, i.e., requested words during a cache miss, in the most significant bits (MSB) of MLC cells. Since MSB bits of MLC cells can be read as fast as SLC cells, this idea can speed up a widely deployed cache optimization technique named critical-word-first. We find a simple and effective way to implement our proposed idea. Taking 2-bit MLC as an example, we strip a cache line at the bit level among all MLC cells in a MLC line. As a result, the first half of a cache line is stored at MSB bits of MLC cells, and the second half is stored at LSB bits.

Our idea is motivated from two important observations.

1) The read latency of a MLC cell is almost proportional to the number bits stored in the cell. If a MLC cell holds two binary bits, the latency of reading the first bit (i.e. MSB) is nearly half of the latency of reading the second bit (i.e. LSB). The speed of reading only MSB bits of MLC cells is almost the same as the speed of reading SLC cells.
2) The position of critical words within a cache line has shown strong patterns and regularity. Our experimental results show that 87% of critical words are in the first half of a cache line. Since a critical word is more urgently needed by the processor than the remaining words in the cache line, transferring the critical word first can reduce the CPU stall time.

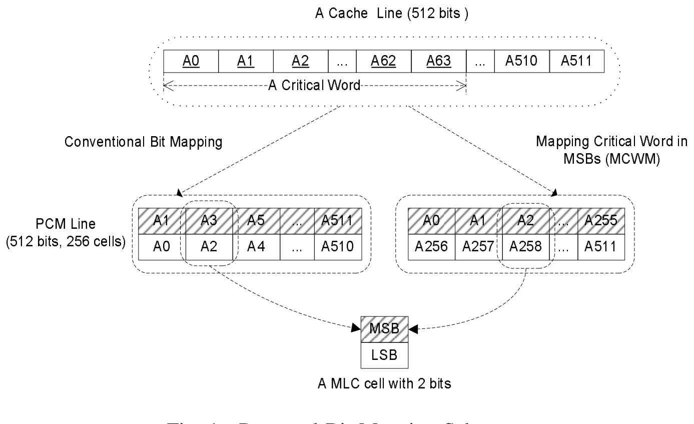

Fig. 1. Proposed Bit Mapping Scheme

Figure 1 illustrates the key idea of MCWM. Assume that a cache line has a size of 64 bytes, i.e. 512 bits, and a MLC cell stores two bits, including the most significant bit (MSB) and the least significant bit (LSB). In conventional PCM, all bits of a cache line is placed to MLC cells sequentially. For example, bits 0 and 1 of the cache line are stored in the first cell, bits 2 and 3 in the second cell, and so on. In MCWM, all bits of the cache line are striped among MLC cells in a PCM line. In this specific example, bit 0 is stored as MSB of the first cell, and bit 1 as MSB of the second cell, and so on. If word 0 (bit 0 - 63) is the critical word of this cache line, all bits of this critical word are stored as MSB bits of MLC cells. Since bits 0 - 255 of the cache line can be read concurrently with a fast speed, the critical word can be fetched first to the processor to quickly continue execution.

We evaluate our proposed MCWM by simulating a multi-core system under SPEC CPU 2006 benchmark suite. Experimental results under 10 multi-programmed workloads show that MCWM reduces the read latency of the standard PCM baseline by 27.5% on average These read latency reductions translate to an IPC improvement of 13.7%. In addition, MCWM is superior to Striped PCM, a recently proposed MLC PCM read optimization strategy, with 12.5% latency reduction and 6.1% IPC improvement. Finally, our proposed design can enhance Write Pause scheme by reducing read latency by 25% and increasing IPC by 11.7% on average.

The rest of this paper is organized as follows. Section II introduces the background of reading MLC cells, critical word regularity and the motivation of this research. Section III presents the design of our MCWM and section IV discusses the experimental results. Related work is summarized in Section V and conclusions are given in Section VI.

II. Background and Motivation

A. Reading MLC PCM

A PCM cell has one transistor and one resistor (1T1R), while a DRAM cell has one transistor and one capacitor (1T1C). PCM exploits remarkably different properties of phase-change material of a memory cell to store data. For example, phase change material Ge2Sb2Te5(GST) has two phases: an amorphous phase with a high resistance of $M\Omega$s and a crystalline phase with a low resistance of $K\Omega$s. Reading data is to sense electric current flowing through the cell. Writing data is to heat up the GST material and change its phase. By exploiting the resistance difference, a single-level cell (SLC) can store one binary bit, and a multi-level cell (MLC) can store two or more bits. MLC uses a narrower resistance band than SLC to represent a bit.

Reading a multiple-level cell (MLC) is slower than reading a single-level cell (SLC). When reading a SLC cell, the resistance of this cell is compared with only one reference cell whose resistance is at the middle of resistance range. However, reading a MLC cell takes multiple iterations of sensing and comparison. More specifically, MLC often uses a binary search strategy to determine the narrow resistance band associated with the cell. As shown in Figure 2, the read circuit first compares the resistance of a MLC cell with a reference cell that has 148 $K\Omega$ [4]. A resistance smaller than 148 $K\Omega$

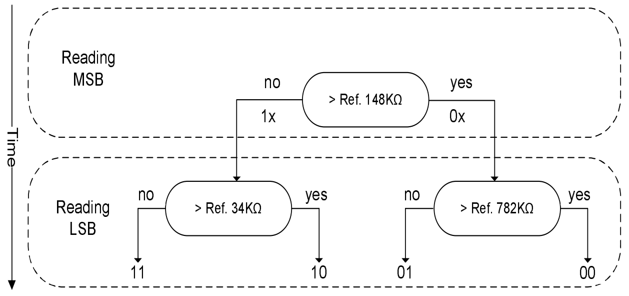

Fig. 2. Reading a MLC cell that stores two binary bits

represents that the most significant bit (MSB) stored in this MLC cell is 1. Otherwise MSB is 0. Then the read circuit reads the value of the next bit stored in the MLC by comparing this cell with a reference cell with 34 $K\Omega$ if MSB is 1 or a reference cell with 782 $K\Omega$ if MSB is 0. This process repeats until all bits of a MLC cell have been read out. In general, reading a n-bit MLC cell takes n iterations of comparison. It is impractical to compare a MLC against all possible reference cells simultaneously. This is because parallel reading requires $2^n - 1$ duplications of sensing and comparison circuits. In order to reduce the hardware overhead, PCM uses one set of sensing and comparison circuits that read iteratively all bits stored in a MLC cell.

Low read latency for MSBs has great potential to reduce the overall read latency. However, the conventional PCM conservatively assumes reading a MLC cell takes as much time as reading its LSB. In this paper, we will present a new bit mapping scheme to exploit the low latency of reading the MSB of a MLC cell.

B. Regularity of Critical Word Accesses

On a cache miss, the requested word is more critical than the other words in that missed cache line. Transferring the missed word (or called critical word) to the processor first allows the execution continue, thus reducing CPU stall time. Ref. [5] observes that words at some specific positions of cache lines are more likely to be critical ones than words at the other positions. Such regularity of critical words has been exploited to improve cache performance and energy efficiency [5]. Ref. [6] finds that the first word (i.e., word 0) of a cache line is a critical word in most cases in SPEC CPU 2006 benchmark, OpenMP NAS parallel benchmark, and STREAM benchmark. It is shown that applications with either stream access patterns or stride access patterns have majority of word 0 as the critical word. On the other hand, the programs with chasing pointers demonstrate less regularity regarding the position of critical words.

C. Motivation

Our experimental results confirm critical word regularity (see Section IV). Assume the size of a cache line size 8 words, Figure 3 shows the average distribution and cumulated distribution of critical words for our SPEC CPU 2006 benchmark studied in this paper (see Section IV for details). On average, 68% of critical words of all read requests are located at word 0 of a cache line. Furthermore, 87% of them are located in the first half of a cache line (i.e., the first four words).

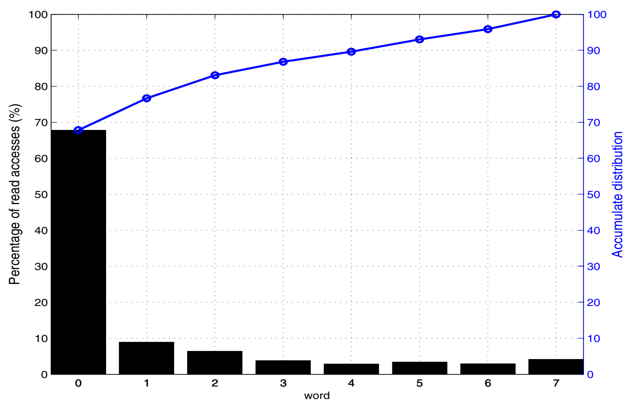

Fig. 3. Distribution of critical words in a cache line with a size of 8 words
.

This observation motivates the design of our MCWM. Reading critical words can be speeded up if they are stored in low-latency MSBs of MLC cells. Since the other non-critical words in the cache line are not accessed urgently, they can be stored in high-latency LSBs of MLC cells without degrading the overall performance. Based on this observation, we propose to place the first half line in MSBs and the second half line in LSBs.

III. Mapping Critical Word in MSBs

We propose a new data mapping scheme, called Mapping Critical Word in MSBs (MCWM), for multi-level cell (MLC) phase-change memory to improve its read performance. Our key idea is to store a critical word at the most significant bits (MSBs) of MLC cells to reduce CPU stall time.

This data mapping scheme is based on the following three important observations. First, reading the MSB bit of a MLC cell takes less time than reading the other bits of the cell. It takes multiple iterations to read a MLC cell, and the MSB bit is the first bit sensed [2], [7]–[9]. Second, conventional MLC PCM uses a simple strategy that places immediate-neighbor bits of a cache line in a MLC cell, without considering the latency difference of reading individual bits from the MLC, as shown in Figure 4. It conservatively assumes that reading each bit in a MLC cell takes the same amount of time as required to read its least-significant bit (LSB). Third, critical words in a missed cache line are often located within the first half of the cache line. For example, in SPEC CPU 2006 libquantum, more than 90% of critical words are located at the first word of missed cache lines [6].

Taking 2-bit MLC as an example, MCWM divides each cache line into two halves, stores all bits of the first half at MSBs of MLC cells, and stores the other half at LSBs, as shown in Figure 4. Upon a cache miss, the memory controller checks the location of the critical word in the missed cache line. If the critical word is in the first half of the missed cache line, the memory controller sends the critical word to processor immediately after reading MSBs from MLC cells. The remaining non-critical words are fetched to the processor after reading LSBs. Otherwise, similar to conventional scheme, the memory controller sends data after reading all bits stored in MLC cells. Since reading MSBs is faster than reading LSBs, our proposed bit mapping scheme takes shorter time to read critical word than the conventional scheme.

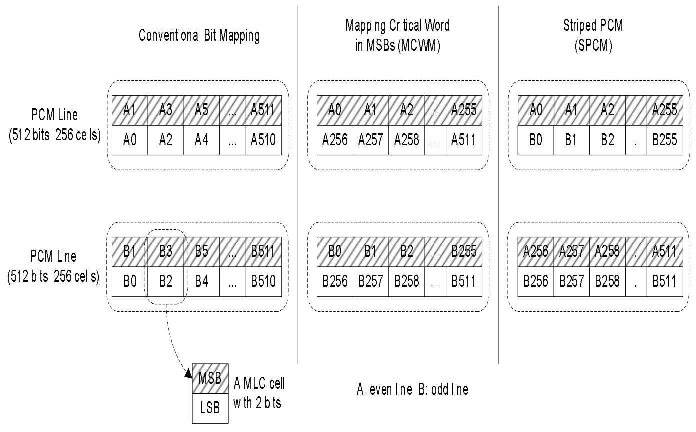

Fig. 4. MLC PCM bit mapping schemes in 2-bit MLC

Our MCWM differs from recently proposed SPCM in the following aspects.

- First, SPCM strips data among MSBs and LSBs at the granularity of cache lines. It stores a cache line with an even address at MSBs, and a cache line with an odd address at LSBs. In contrast, striping in MCWM is performed at the bit level within a cache line. While bits stored in a MLC cell in SPCM belongs to different cache lines, bits stored in a MLC cell in MCWM are part of the same cache line.
- Second, SPCM has much larger silicon area overhead than our proposed MCWM. In SPCM, reading a line involves two PCM lines and requires twice as many sense/amplifiers (SA) as our design. Since PCM's current S/As have larger area overhead than voltage ones used in DRAM, a PCM chip has a limited number of current S/As.
- Third, while SPCM mostly reduce the read latency for lines with even address, our MCWM can accelerate reads for all cache lines.
- Fourth, SPCM not only consumes more energy but also reduces the reliability of PCM. SPCM can reduce the lifetime of PCM by half in the worst case. When writing a cache line, SPCM always update two paring PCM lines. On the contrary, writing a cache line in our proposed MCWM only needs to update one PCM line.

IV. Experiment Results

We evaluate our design by using the execution-driven processor simulator GEM5 [10] and the cycle-level memory simu-

Parameter	Value
System	4-core CMP, 4 GHz
Execution Core	Alpha-like out-order processor
L1 Cache	32KB I-cache, 32KB D-cache
L2 Cache	Latency 20ns, 1MB, 4-way, 64B cache line
L3 Cache	Latency 50ns, 32MB, 8-way, 64B cache line
Memory Controller	28 read/write queue entries, read priority scheduling (unless $WRQ > 80\% full$)
Memory Organization	1 ranks, 8 bank/rank
PCM read latency	MSB: 125ns, LSB: 250ns
PCM write latency	SET: 125ns, RESET: 250ns; Write model: F1 = 0.375, F2 = 0.625, i = 2 for '01'; F1 = 0.425, F2 = 0.675, i = 2 for '10'; fixed 1 iteration for '00', fixed 2 iterations for '11'

TABLE I
SIMULATION PARAMETERS

Workload	Description	RPKI	WPKI
MIX1	astar, astar, astar, astar	2.23	1.01
MIX2	astar, cactusADM, libquantum, soplex	0.72	0.04
MIX3	cactusADM, cactusADM, gobmk, gobmk	1.88	0.81
MIX4	soplex, sooplex, sjeng, sjeng	0.4	0.01
MIX5	bwaves, soplex, sphinx3, cactusADM	1.13	0.12
MIX6	sphinx3, cactusADM, gromacs, omnetpp	0.46	0.03
MIX7	mcf, xalancbmk, GemsFDTD, lbm	1.03	0.22
MIX8	mcf, GemsFDTD, povray, perlbench	0.21	0
MIX9	mcf, GemsFDTD, perlbench, gcc	0.39	0.02
MIX10	GemsFDTD, lbm, povray, namd	0.24	0

TABLE II
CHARACTERISTICS OF 10 WORKLOADS FOR A FOUR-CORE SYSTEM

lator NVMain [11]. Table I shows the parameters of simulated processor and PCM. The processor simulated has four out-order cores with 32 MB L3 cache. The read latency of PCM MSB and MLB is set as 125ns and 250ns, respectively [2]. We also apply multiple program-and-very iterations [1]–[3], [12] to model writing. In addition, due to slow writing, reads are prioritized over writes until the occupancy of write requests in the memory I/O queue reaches 80%, which has been recommended by other researchers [1]–[3], [12].

We construct 10 multi-programmed workloads by combining multiple applications selected from the SPEC CPU 2006 benchmark suit. Most of these combined applications have a relatively small amount of memory writes since our study focuses on the impact of the memory read latency on the overall system performance. In addition, we also include memory write intensive workload $MIX1$. All applications in each workload run in parallel. Each application is fast-forwarded 15 billion instructions and then 300 million instructions are simulated.

Table II summarizes key characteristics of these workloads, including memory Read Per Kilo Instructions (RPKI), memory Write Per Kilo Instructions(WPKI). While our L3 cache has 32MB and is smaller than the one used in other studies [1], [2], the intensity of memory accesses measured in our workloads is very close to theirs and thus we believe a larger L3 cache is unnecessary.

We compare MCWM with the baseline scheme that does not incorporate any read or write optimization. We also compare MCWM with two recently proposed techniques, including Write Pause (WP) [1] and Striped PCM (SPCM) [7]. Since WP takes advantages of iterative write algorithms to suspend writes, WP is complementary to MCWM and SPCM, which focus on optimizing reads directly. Therefore, we will combine MCWM and SPCM with WP, which is refereed to as WP+MCWM and WP+SPCM, respectively.

A. MSB Hit Rate

The MSB hit rate (MHR) is defined as the percentage of memory read accesses whose critical word is stored in MSBs of MLC cells. A higher MHR means more read requests can benefit from the low latency of MSBs. Figure 5 compares the MHR of SPCM and MCWM. As expected, SPCM places the even lines at MSBs and hence 50% of read requests are served by MSBs. In contrast, MCWM stores the first half of a cache line to the MSBs and the first half is very likely to contain critical words of the corresponding cache line as shown in Figure 3. MCWM's MHR is significantly larger than 50%. For example, workload $MIX3$ has very strong critical word regularity and its MHR is close to 100%. The average MHR of all workloads studied in this paper reaches 85%, which is consistent with critical word distribution presented in Figure 3.

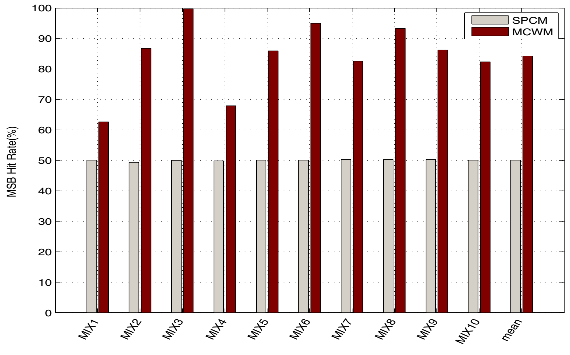

Fig. 5. MSB Hit Rate Comparison for SPCM and MCWM

B. Read Latency

Figure 6 presents the reduction of read latency of MCWM over the baseline, and compares it with the other optimization schemes. We have the following observations.

First, MCWM achieves more read latency reduction than SPCM. For example, on average MCWM reduces read latency by 27.5% compared with the baseline, and it outperforms SPCM by 12.5%. This is because MCWM's MHR is higher than SPCM and accordingly more read requests benefit from low-latency MSBs.

Second, while WP can effectively reduce read latency for write-intensive workloads, such as $MIX1$, it has less chance to reduce read latency for non-write-intensive workloads, such as $MIX4$. In those read intensive workloads, both MCWM

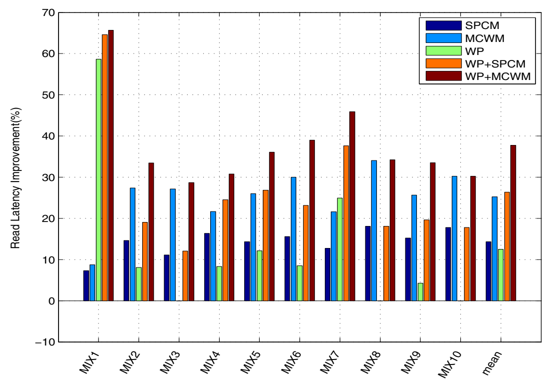

Fig. 6. Read Latency Reduction

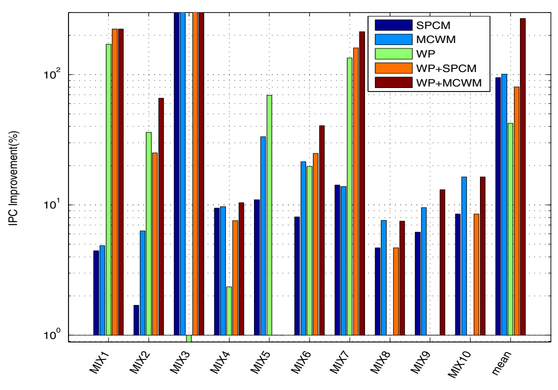

Fig. 7. IPC Improvement (Log scale)

and SPCM can more effectively reduce read latency. For example, in workload $MIX9$, MCWM and SPCM can reduce read latency 21% and 13% more than WP respectively. On average, MCWM and SPCM achieve 15% and 3% more read latency reduction respectively.

Third, MCWM and SPCM can be effectively combined with WP. The formers directly optimize reads while the latter indirectly optimizes reads by pausing writes. After WP successfully mitigates the negative impact of slow writes on performance, slow MLC reads become a new performance limiter. Our experimental results show that MCMW+WP and SPCM+WP achieve latency reduction 25% and 14% more than WP on average.

C. IPC

Figure 7 shows the IPC improvement over the baseline in log scale. The experiment results confirm that MCWM's reduced read latency is directly translated into performance gain. Since workload $MIX3$ is both read and write intensive, the memory slows down the overall performance significantly. Under such workload, MCWM and SPCM successfully improve the IPC with 882.2% and 884.2% respectively. Excluding $MIX3$, on average MCWM and SPCM outperform baseline with 13.7% and 7.6% respectively. In addition, MCWM and SPCM outperform WP by 11.7% and -5% when they are combined with WP. It is noted that $MIX5$ and $MIX9$ are very heterogeneous in terms of memory intensity. If IPC is improved, more memory requests are generated under these two workloads, which block the execution of other applications that have less memory requests. Therefore, the average IPC for four cores is decreased even thought read latency is reduced under some workloads.

V. Related Work

The following summarizes closely related work from two aspects: optimizing read and write on PCM, and critical word first.

Several research projects have aimed to hide the long write latency of PCM. Ref. [13] adds a buffer to each PCM bank and exploits the data locality to mitigate the slow write. They further propose a partial write strategy for a cache line to reduce the amount of data written to PCM. Flip-N-Write [14] and RBW/DI [15] use a simple read-modify-write technique to write either flipped or unflipped data to reduce write time. Write cancellation and write pausing [1] are proposed to indirectly improve the PCM read performance. Write cancellation aborts an on-going write for a newly-arriving read request targeted to the same bank if the write operation is not close to completion. Recently, write truncation and form switch [2] are proposed to improve write performance for the multi-level-cell PCM. Based on the observation that not all bits for a block of data need the same number of write iterations, the write truncation early terminates the write iteration when most bits have been successfully written and then recover the data with extra error correction code during reading. In addition, PCM needs high levels of electrical current to write to a PCM cell and thermally change its state. Delivering such high levels of power is a challenge for both PCM chips and the system. Ref. [3], [12], [16]–[19] work on write power management to address write throughput loss due to limited power budget inside PCM chip. After PCM write optimizations are deployed, the high read latency become an emerging performance limiter in MLC PCM.

High read latency of MLC PCM has drawn important attentions. SPCM [7] first addresses this issue by exploiting the MSBs low read latency. It strips a line across two PCM data blocks and maps an even line and an odd line to MSBs and LSBs inside two paired blocks respectively. Writing one line to PCM, SPCM needs to write two paired PCM blocks, which not only consume more energy but also reduces PCM reliability. Our design has no such limitations.

Transferring critical word before other words can reduce processor stall time since the other words are not needed immediately [20], [21]. Ref. [5] observes critical word access regularity and exploits this insight to optimize L2 cache performance and energy efficiency. Recently, Ref. [6] further confirms that the first word (word 0) of cache lines accounts for a significant number of cache line fetches. They propose to store word-0 in the lower access latency DRAM and other words in the other DRAM with higher access latency to improve heterogeneous DRAMs performance. In this paper, we propose to store words in the first half of a cache line at MSBs of MLC cells to reduce PCM read latency.

VI. Conclusion

Multi-level-cell (MLC) PCM achieves higher storage density than single-level-cell (SLC) at the cost of inferior write and read performance. Slow read latency can become a performance limiter in MLC, particularly under read-intensive workloads. This paper presents and evaluates a new and simple bit mapping scheme, called MCWM, which takes advantage of the latency difference between reading most-significant-bits (MSBs) and least-significant-bits (LSBs) of a MLC cell to improve read performance. At the bit level, MCWM strips all bits of a cache line among MLC cells of a PCM line. Taking 2-bit MLC for example, MCWM maps the first half of a cache line to MSBs of MLC cells, and maps the second half to LSBs. This simple mapping strategy fully leverages the distribution regularity of critical words. For SPEC CPU'06 benchmark studied in this paper, on average 87% of critical words are located within the first half of a cache line. On a cache miss, MCWM allows the critical word of the missed cache line to be fetched from multi-level-cell (MLC) at a speed as fast as single-level-cell (SLC), thus reducing the processor stall time.

Experiment results of 10 multi-programmed SPEC CPU 2006 workloads on a four-core out-order system with MLC PCM show that MCWM successfully reduce the read latency of standard MLC PCM baseline by 27.5%, and improve IPC by 17.3% on average (excluding workload $MIX3$). Workload $MIX3$ is both read and write intensive, the performance gain of MCWM is even more significant and the IPC is almost 8 times better than the baseline. On average, MCWM reduces read latency 12% and 15% more than SPCM and WP respectively. Lastly, MCWM can further reduce read latency by 25% on average when working with WP.

MCWM can be easily implemented at the memory controller level It does not introduce hardware overhead since no extra hardware components is required. It also does not incur software overhead since the bit mapping is very simple. No memory space overhead will be generated since the proposed mapping scheme is fixed and no status bits needs to be recorded.

A. Acknowledgement

We would like thank our shepherd Daniel Sorin, anonymous reviewers, and the program committee. We also thank Ali Shareef and Vincent Weaver for their proofreading. This work was supported by National Science Foundation under grants IIS 091663, CNS 1117032, EAR 1027809, CCF 0937988, CCF 0621493, and EPS 0904155.

References

[1] K. Q. Moinuddin, M. F. Michele, and L. A. Lastras-Monta, "Improving read performance of phase change memories via write cancellation and write pausing," in *Proceedings of HPCA*. IEEE Computer Society, 2010.

[2] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in mlc phase change memory," in *Proceedings of HPCA*, 2012, pp. 201–210.

[3] L. Jiang, Y. Zhang, B. R. Childers, and J. Yang, "Fpb: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory," in *Proceedings of Micro*, 2012.

[4] W. Zhang and T. Li, "Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system," in *DSN'11*, 2011, pp. 197–208.

[5] E. J.Gieske, "Critical words cache memeory: Expploiting criticality with primary cache miss streams," 2008.

[6] N. Chatterjee, M. Shevgoor, R. Balasubramonian, and etc, "Leveraging heterogeneity in dram main memories to accelerate critical word access," in *Proceedings of MICRO*, 2012.

[7] M. Hoseinzadeh, M. Arjomand, and H. Sarbazi-Azad, "Reducing access latency of mlc pcms through line striping," in *Proceedings of ISCA*, 2014.

[8] M. K. Qureshi, M. Franceschini, L. Lastras, and etc, "Morphable memory system: A robust architecture for exploiting multi-level phase change memories," in *Proceedings of ISCA*, 2012.

[9] F. Bedeschi, R. Fackenthal, C. Resta, and etc, "A bipolar-selected phase change memory featuring multi-level cell storage," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, Jan. 2009.

[10] N. Binkert, B. Beckmann, G. Black, Reinhardt, and etc, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[11] M. Poremba and Y. Xie, "Nvmain: An architectural-level main memory simulator for emerging non-volatile memories," in *Proceedings of the 2012 IEEE Computer Society Annual Symposium on VLSI*, ser. ISVLSI '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 392–397.

[12] L. Jiang, B. Zhao, J. Yang, and Y. Zhang, "A low power and reliable charge pump design for phase change memories," in *Proceedings of ISCA*, 2014.

[13] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *Proceedings of the 36th annual international symposium on Computer architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 2–13.

[14] S. Cho and H. Lee, "Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 347–357.

[15] Y. Joo, D. Niu, X. Dong, G. Sun, N. Chang, and Y. Xie, "Energy- and endurance-aware design of phase change memory caches," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '10, 2010, pp. 136–141.

[16] A. Hay, K. Strauss, T. Sherwood, G. H. Loh, and D. Burger, "Preventing pcm banks from seizing too much power," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44 '11. New York, NY, USA: ACM, 2011, pp. 186–195.

[17] J. Yue and Y. Zhu, "Accelerating write by exploiting pcm asymmetries," in *Proceedings of 2013 IEEE 19th International Symposium on High Performance Computer Architecture*, 2013.

[18] ——, "Exploiting subarrays inside a bank to improve phase change memory performance," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013.

[19] ——, "Making write less blocking for read accesses in phase change memory," in *Proceedings of 2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2012.

[20] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach 4th ed.* Elsevier Press, 2007.

[21] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems - Cache, DRAM,Disk*. Elsevier Press, 2008.