SPCM: The Striped Phase Change Memory

MORTEZA HOSEINZADEH and MOHAMMAD ARJOMAND, Sharif University of Technology
HAMID SARBAZI-AZAD, Sharif University of Technology & Institute for Research in Fundamental Sciences (IPM)

Phase Change Memory (PCM) devices are one of the known promising technologies to take the place of DRAM devices with the aim of overcoming the obstacles of reducing feature size and stopping ever growing amounts of leakage power. In exchange for providing high capacity, high density, and nonvolatility, PCM Multilevel Cells (MLCs) impose high write energy and long latency. Many techniques have been proposed to resolve these side effects. However, read performance issues are usually left behind the great importance of write latency, energy, and lifetime. In this article, we focus on read performance and improve the critical path latency of the main memory system. To this end, we exploit striping scheme by which multiple lines are grouped and lie on a single MLC line array. In order to achieve more performance gain, an adaptive ordering mechanism is used to sort lines in a group based on their read frequency. This scheme imposes large energy and lifetime overheads due to its intensive demand for higher write bandwidth. Thus, we equipped our design with a grouping/pairing write queue to synchronize write-back requests such that all updates to an MLC array occur at once. The design is also augmented by a directional write scheme that takes benefits of the uniformity of accesses to the PCM device—caused by the large DRAM cache—to determine the writing mode (striped or nonstriped). This adaptation to write operations relaxes the energy and lifetime overheads. We improve the read latency of a 2-bit MLC PCM memory by more than 24% (and Instructions Per Cycle (IPC) by about 9%) and energy-delay product by about 20% for a small lifetime degradation of 8%, on average.

CCS Concepts: • **Information systems** → **Phase change memory**; • **Hardware** → **Memory and dense storage**

Additional Key Words and Phrases: Phase change memory, multilevel cell memory, line striping

ACM Reference Format:

Morteza Hoseinzadeh, Mohammad Arjomand, and Hamid Sarbazi-Azad. 2015. SPCM: The striped phase change memory. *ACM Trans. Archit. Code Optim.* 12, 4, Article 38 (November 2015), 25 pages.
DOI: <http://dx.doi.org/10.1145/2829951>

1. INTRODUCTION

The need for a high density and low latency main memory becomes a major issue in future Chip Multi-Processors (CMPs) to meet the capacity and performance requirements for running multiple threads or applications. DRAM has successfully kept pace with these demands by providing roughly $2\times$ density every 2 years and stretching to twice its working frequency every 3–4 years [Kilbok 2007; Stuecheli et al. 2010]. Entering a deep nanometer regime where leakage power and process variation are dominant factors, however, large DRAM-based arrays confront serious power, scalability, and reliability limitations [Condit et al. 2009; Lee et al. 2009; Qureshi et al. 2009b; Zhou

---

Authors' addresses: M. Hoseinzadeh, M. Arjomand, and H. Sarbazi-Azad, HPCAN Lab, Computer Engineering Department, Sharif University of Technology, Tehran, Iran; emails: {mhoseinzadeh, arjomand}@ce.sharif.edu, azad@sharif.edu; H. Sarbazi-Azad, School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran; email: azad@ipm.ir.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2015 ACM 1544-3566/2015/11-ART38 \$15.00

DOI: <http://dx.doi.org/10.1145/2829951>

et al. 2009]. Therefore, memory technologies that can provide more scalability have become attractive for designing future memory systems. Among the various resistive memories, Phase Change Memory (PCM) is the most promising candidate that has a read latency close to that of DRAM [Condit et al. 2009; Lee et al. 2009; Qureshi et al. 2009b; Zhou et al. 2009].

PCM exploits the ability of chalcogenide alloy (e.g., $\text{Ge}_2\text{Sb}_2\text{Te}_5$, GST) to switch between two structural states with significantly different resistances (i.e., high resistance amorphous state and low resistance crystalline state). There is a resistance difference of three to four orders of magnitude between the crystalline (i.e., SET) and amorphous (i.e., RESET) states. This wide resistance gap is sufficient to enable good and safe memory readability in Single-Level Cell (SLC) PCMs. Nevertheless, Multilevel Cell (MLC) operation mode is used to achieve high density PCM devices. To store more bits in a single cell, MLC uses fine-grained resistance partitioning, which can be stabilized by adjusting amplitude or duration of programming pulses. MLCs mainly rely on an iterative sensing technique for read and a repetitive program-and-verify technique for write operation. Unfortunately, these two schemes can significantly increase the effective latency of the memory and need consideration when employing MLC PCM as main memory. The MLC write is a well documented problem and has fueled several recent studies that attempt reducing the number of write iterations or using write buffers and intelligent scheduling [Cho and Lee 2009; Joshi et al. 2011; Lee et al. 2009; Qureshi et al. 2009b, 2010a; Zhang et al. 2012; Zhou et al. 2009]. However, less attention has been paid to the problem of high read latency in MLCs. In this article, we propose a scheme that obtains higher read performance without incurring significant hardware overhead and complexity.

A key insight that enables our solution is that read latency of different bits of a cell in MLC mode is not equal. As stated in Son et al. [2013], sensing takes about 47% of memory access time (tAC) [Choi et al. 2012]. On the other hand, in PCM devices, because of the nature of nonvolatility, there is no need to store data back to the row after read operations. Then, 48% of cycle time (tRC) that previously was taken by restoring can be void. Consequently, sensing has even more contribution in tRC [Close et al. 2013]. In a 2-bit MLC PCM device, tRCD consists of two iterations for reading two bits. When a read request is scheduled for a PCM line, data bits stored in Most Significant Bits (MSBs) of the cells are read in the first iteration, while the remaining bits (in the Least Significant Bits (LSBs), of the same cells) cannot be read until the MSBs are determined. Thus, MLC read latency is much higher than that of SLC's (almost $2\times$ for 2-bit MLC). Note that, read accesses, unlike writes, are on the main memory's critical path and can significantly impact system performance.

In 2-bit MLC,¹ we could reduce the latency of a memory line by $2\times$ if its bits are stored in MSBs of the cells. We exploit this key insight and propose *Striped PCM (SPCM)*, a memory architecture that leverages the read asymmetry of MLCs to access a memory line with low latency. Indeed, unlike the traditional MLC PCM in which data bits of a single line are coupled and aligned vertically in cells, SPCM couples data bits of two lines and interleaves them over the cells of a memory row. When a read request reaches the main memory, the memory line is read quickly, if it is stored in MSBs of the cells in a row. Otherwise, the memory line is stored in LSBs and its read should be preceded by accessing the partner line stored in MSBs. Thus, read operations are faster for half of the lines (stored in MSBs of the cells) and not slower than a conventional MLC for the other half (stored in LSBs of the cells). Since SPCM only rearranges block addressing and data alignment, it imposes no latency overheads to the main memory's critical path.

¹The proposed technique in this article is general and can be applied to any MLC density. For simplicity, we describe our methodology for 2-bit MLC.

In addition to rearrangement policy, we develop an ordering mechanism by which the most frequently read line of a group is placed on the most significant bits of the cells in a cell array. The level of bits in cells storing a memory line decreases with respect to its read frequency. Consequently, the ordering mechanism raises the chance of reading a block from MSBs, which requires fewer read sequences.

While SPCM improves performance significantly, it increases the number of times that each cell is written in the same program run. In fact, any update to a memory line should be accompanied with reading and rewriting its partner lines. This is undesirable as it incurs power overheads and reduces system lifetime. To limit such overheads while retaining the potential performance benefits, we rely on a key insight that when a line is written back to the main memory, the corresponding partner lines are probably in the write buffer or dirty in the Last-Level Cache (LLC) and will be written back to the main memory later. Therefore, write requests can be rescheduled to enable pair writes of multiple memory lines sharing the same MLC array. More accurately, if all partner lines are in the write buffer, they are given the highest priority for write service. Otherwise, if a partner line is in the LLC and it is dirty, it may be written again before getting evicted from the LLC. Then, it is better to give it the lowest priority in the write queue to increase the chance of pair writes. Following this scheduling policy, we manage to keep the write rate in the SPCM almost equal to that in the conventional PCM. This can prevent energy overhead and lifetime degradation while improving the performance significantly.

Although such a priority write queue prevents undue writes on partner lines, our observation shows that rewritten blocks are most likely to be written again. So, we can switch the write operation to nonstriped mode whenever the line is considered as write intensive. Based on this observation, we propose an adaptive morphology for SPCM using directional write operation (*diwrite*), which means that writing to a group of lines can be either in striped mode (when it comes from an I/O device via DMA), or nonstriped mode (when it is requested by LLC or master port for multiple times). When a group of lines is written in nonstriped mode, subsequent read accesses to them will be similar to the conventional MLC PCM. As a result, it reduces the energy consumption and prolongs lifetime at the cost of performance overhead.

We discuss the extensions to memory controller in order to equip SPCM with write-reordering scheme and *diwrite* operation. Our evaluations reveal that the SPCM with the proposed write-aware supplements shows a performance near to a pure SPCM but significantly reduces power and lifetime overheads. Overall, the ultimate design can reduce the effective read latency of the baseline MLC PCM by more than 24% and improve IPC by about 9%, on average. This scheme also improves the energy-delay product of the system by about 20% with a lifetime overhead of 8%, on average.

The rest of this article is organized as follows. Section 2 describes a brief background on MLC PCM and its read and write mechanisms. Section 3 explains our motivation followed by Section 4 that presents details of the proposed striping mechanism, preliminary results, and its shortcomings. Supplementary designs are presented in the rest of the section to address the mentioned shortcomings. Section 5 represents the evaluation results. Cost, performance, and energy analysis of the proposed memory architecture is presented in Section 6. Section 7 discusses related works, and finally, Section 8 concludes the article.

2. BACKGROUND ON MLC PCM

The wide resistance difference between the amorphous and crystalline states enables the concept of MLC PCM technology. Currently, 2-bit MLC prototypes can be found, but International Technology Roadmap for Semiconductors (ITRS) predicts higher densities (3 and 4) being achievable near to 2020 [Allan et al. 2002]. In MLC, the

number of resistance levels increases exponentially with the number of bits stored in a cell, implying that the resistance band assigned to each data value must be narrowed accordingly. Therefore, MLC read/write operations need dedicated control mechanisms, which are described in this section.

MLC Write. In MLC, a specified resistance band is assigned to represent a specific data value and the write process must be accurate enough to program the cell. To this end, MLC PCM controllers widely rely on a repetitive program-and-verify technique [Bedeschi et al. 2009; Nirschl et al. 2007]. When the write circuit receives a request, it first injects a RESET pulse with large voltage magnitude to completely RESET the cell, and then injects a sequence of SET pulses with lower voltage magnitudes. After each SET pulse, the circuit reads the cell resistance value and verifies whether it falls within the target resistance range; if so, the write process is completed. Otherwise, the circuit recalculates the write parameters (either amplitude or duration of SET pulses) and repeats the steps until the target resistance is formed.

Due to process variation and composite fluctuation of nanoscale devices, nondeterminism arises in MLC PCM writes [Jiang et al. 2012a; Zhang and Li 2009]. The cells storing a memory line may take a variable number of iterations to finish. Furthermore, the same cell requires a different number of iterations when writing different data values. Thus, to write a memory line, the worst-case write latency is variable. This is a known problem for PCM and many previous studies proposed techniques to either speed up MLC write accesses [Jiang et al. 2012c] or to protect the processor from the negative impacts of slow writes [Zhou et al. 2009].

Besides inferior performance, PCM writes bring two main challenges: (1) limited cell lifetime, and (2) considerable energy consumption per write access. These challenges are even exacerbated in MLCs because of the iterative pulses required during write operation. Limited cell lifetime can be dealt with by using combinations of three strategies: (1) reducing the total number of cell updates by differential write [Cho and Lee 2009; Lee et al. 2009; Zhou et al. 2009], (2) spreading cell wear out [Qureshi et al. 2009a; Seong et al. 2010a; Zhou et al. 2009], and (3) tolerating cell failure [Condit et al. 2009; Schechter et al. 2010; Seong et al. 2010b; Yoon et al. 2011]. The most promising strategies are those in the first category because, in addition to reducing cells' wear out, they reduce the required energy for write operation. Although all known wear management and error correction schemes are fully applicable to the SPCM architecture, our baseline system assumes differential writes [Zhou et al. 2009], as it can potentially reduce both wear-out rate and energy.

MLC Read. Reading a PCM cell involves sensing the resistance level and mapping it to its corresponding data value. Read operation usually applies binary search to determine MLC content bit by bit (Figure 1).

At the first step, the circuit compares the resistance to a reference cell. The reference cell's resistance is chosen as roughly the middle of the resistance window. Depending on the comparison outcome, (1) the MSB is determined (a larger resistance indicates the stored bit is "0"; otherwise, it is "1") and (2) the circuit chooses the next reference resistance for the next bit. This process iteratively continues until all bits are read. In general, a read operation for an n -bit MLC requires n iterations to complete [Close et al. 2013], which is not desirable as the processor performance highly depends on memory read latency. A straightforward solution to this problem is to perform all comparisons in parallel and complete the read in one step. Since such a parallel sensing requires $2^n - 1$ copies of the read circuits for an n -bit MLC PCM, it increases sensing current and poses hardware overhead in addition to fabrication constraints, which make it impractical. As such, a bit by bit read mechanism is widely used in current MLC PCM prototypes [Bedeschi et al. 2009; Qureshi et al. 2010b; Close et al. 2013].

Fig. 1. 2-bit MLC read mechanism. Reference resistance levels are driven from Zhang and Li [2011] for $2\times$ difference between adjacent regions in order to tolerate 1 minute drift.

Fig. 2. (a) Conventional vertically aligned bit mapping. (b) Striped bit mapping.

3. MOTIVATION

Considering a 2-bit MLC PCM, the MSB of a PCM cell can be determined in early cycles of a read access irrespective of the LSB. This asymmetry brings an opportunity to reduce read latency and energy consumption of MLC PCM. As a solution, we propose pairing two consecutive data blocks and rearranging them horizontally in an array of cells such that each cell stores one bit of the first line and one bit of the other. Thus, MSBs hold the data block with odd address and LSBs hold the block with even address, as shown in Figure 2. This reduces the average memory access latency since the block at odd address is retrieved in the first read step (as in SLC mode), while reading the other block takes the latency of 2-bit MLC read.

Based on our observation when a read request at an odd address arrives, the next line with even address is requested shortly most of the times as a result of spatial locality. Hence, if we buffer the most recently read blocks from the PCM array requested by the memory controller in a Read Buffer (RB), it might not be necessary to spend extra read iterations to extract the MSB line (as a reference to extract the LSB line). Even in the case of miss in RB, the LLC may have already cached the MSB block. We define Odd-Block Hit-Rate (OBHR) as the frequency of odd block hits in either RB or LLC, required to realize a read for an even-address block. Overall, a higher average OBHR means a smaller number of redundant accesses to the PCM array. Figure 3(a) gives the breakdown of OBHR. As shown in Figure 3(a), approximately 63% of requested odd blocks are found in RB. Intuitively, to read the requested data block with even address, the controller can first request the odd block from RB or LLC. If it fails, it reads this block from the PCM array. Notice that if the odd block is found in the LLC, it must be clean.

Fig. 3. Distribution of different memory accesses: (a) Breakdown of the OBHR in RB and LLC. The configuration reported in Section 5.1 augmented with a RB of 20 slots is used in our simulations, and (b) the percentage of one-time reads, odd and even rereads.

Fig. 4. Timing diagram for (a) all possible read requests (RD1: Read odd block; RD2: Read even block/buffered odd block; RD3: Read even block/NOT buffered odd block), (b) even block write requests (WR1: Write even block/NOT buffered odd block; WR2: Write even block/buffered odd block), and (c) odd block write requests (WR3: Write odd block/NOT buffered even block; WR4: Write odd block/buffered even block). We assume 1, 3, and 6 time units for buffer, one MLC read sequence, and write latencies respectively.

Figure 3(b) shows stacked bars representing the percentage of one-time reads (not more than one access to them), odd rereads (reading an odd block more than once), and even rereads (reading an even block more than once). Based on this observation, the rates of odd and even rereads are almost identical. It can be hypothesized that with a proper swapping mechanism, we can control the frequency of rereads, such that odd blocks are accessed more than even blocks. An ordering mechanism may provide a better utilization of SPCM by letting the most frequently read blocks place on the MSBs of cell arrays.

4. THE STRIPED PHASE CHANGE MEMORY (SPCM)

Again, consider a 2-bit MLC PCM. Based on the preceding motivation, SPCM seeks for a better performance by pairing two memory lines. For simplicity, we assume that the two paired blocks differ in the lower bit of the block address. This rearrangement requires some modifications in read and write operations.

Read. When a block with odd address is read, it is retrieved within the first iteration (RD1 in Figure 4). Given an even block address, the SPCM controller requires to know the MSBs of the cells to use as reference. So, the SPCM controller first examines the RB and LLC in parallel for the odd block. On a hit (either in RB or in the LLC when the odd block is clean) the requested even block can be retrieved in approximately one iteration (RD2). In the worst case, the odd block is not present in RB or LLC (or it is dirty in LLC), and the memory controller calls for reading the odd block as well; hence, the read process lasts as long as that in the baseline memory (RD3). Fortunately, the odd block usually exists in the RB due to spatial locality, enhancing the chance of performance improvement.

Write. Regarding line striping, all updates to a memory line necessitate writing both MSB and LSB blocks at once. Therefore, the memory controller first attempts to obtain the partner block from the LLC; if it fails, the SPCM is accessed by using the read mechanism described previously. In the case of updating an even block, the partner odd block must be written as well. If the odd block is not cached (WR1 in Figure 4), it should be retrieved in one iteration of MLC read operation. Otherwise, the controller obtains it by accessing the cache (WR2). Similarly, when an odd block is requested for write, its partner even block must be written back too. In the worst case, the even block is not cached (WR3) and the controller needs to extract the old odd block (as a reference for reading the even block) and the partner even block itself. However, when the even block is cached (WR4), it can be obtained by accessing the cache. Finally, buffered blocks are scheduled to be written to the main memory in next cycles (see Hudgens and Johnson [2004] for read/write-aware scheduling mechanisms).

4.1. Ordering Mechanism

According to Figure 3(b), reread accesses to odd and even lines are evenly distributed, which calls for an optimization. Thus, we propose a simple ordering scheme to move the most frequently read line toward the MSB. In 2-bit SPCM, only one swap operation is required after the number of read accesses to a line exceeds a fixed threshold. This upward movement incurs extra write operations, which harms the durability of the PCM device. Therefore, we must investigate best thresholds that enhance the ratio of odd reads per total reads while imposing less swap operations. Figure 5 shows the behavior of SPCM in one set (SET1) of benchmarks with our ordering (here swapping for 2-bit MLC) mechanism enabled. At the first column of this figure, the read ratio of odd blocks over total accesses is represented for different thresholds—value 0.5 indicates an equal number of read accesses to odd and even blocks. Obviously, since SPCM is more effective for read accesses to odd blocks, the greater value of odd ratio reveals more performance gain. As shown, for $T = 0$ (i.e., a swap occurs once the even block is accessed), SPCM works more efficiently. Although this speedup is considerable, it might impose a huge number of swaps, which is not desirable for the PCM device. The number of swaps is shown in the second column of Figure 5. This graph shows that the system will be saturated after a number of swaps and reaches a stable state in which many of the most frequently read lines are placed in MSBs and no further swap is required. To better show the impact of the threshold, the third column of the figure illustrates the proportion of reads from odd blocks over the number of swaps. The higher value of this ratio indicates a more efficient response to the swapping mechanism (higher rate of reading odd blocks and lower number of swaps). As a consequence, having $T = 0$ seems not beneficial according to this metric. However, the number of swaps is usually proportional to the total writes. The last column of Figure 5 depicts the ratio of swaps over writes. The saturation line in this graph is visible too. The steadily falling ratio of swaps/writes reveals that the system is reaching a steady state in which swaps are much less than writes (below 3% for all workloads). Here, having $T = 0$ is the best threshold for our swapping scheme. Note that this experiment is done for 2-bit MLC PCM. For higher densities, the cost of ordering should be significant, and thus, a larger threshold would be better. We also reiterated the experiment for different LLC sizes. Results show that while the upper bound values differ vastly, the trends do not change. However, larger LLC prevents many of the accesses to the SPCM, and consequently, much less number of swaps and writes occur.

The ordering mechanism requires $\lceil \log_2(n!) \rceil$ bits per n lines to keep the permutation of lines' ordering. Then, for SGB of n -bit SPCM comprising B -byte lines, $\frac{S}{n \times B} \times \lceil \log_2(n!) \rceil$ Gb metadata is required. For example, a 4GB 2-bit SPCM with 128B line size requires only

Fig. 5. The behavior of SPCM for MCF, BZIP2, OMNETPP, and XALANCBMK applications (SET1) with the swapping mechanism enabled. Other benchmarks (SET2, SET3, ..., SET9) exhibit nearly the same behavior.

2MB metadata for the ordering mechanism. This metadata is maintained in an external fast DRAM chip. For each access to the SPCM, the controller fetches the metadata from the external DRAM chip, to get the order of lines. It then retrieves data from SPCM using the ordering data. Of course, this mechanism incurs a negligible performance overhead, which is overwhelmed by the great amount of performance gain through the ordering mechanism.

4.2. Shortcomings

In the SPCM system, once a block is evicted from the LLC, it should be combined with its partner before writing in the associated cell array. For example, consider block *A* that becomes dirty at t_1 and is going to be written back to the PCM array at t_2 , but its partner block *B* is not dirty before $t_3 > t_2$. So, $A \cup B$ is written to the associated PCM cell array at t_2 . Subsequently, block *B* becomes dirty at t_3 and is evicted from the

Fig. 6. Abstract Concept of Waiting Room. R-marked people are ready to go, because the couple is in the room; The mates of exclamation-marked (!) people are not in the LLC (previously have gone), so they do not need to wait for anyone. Lastly, W-marked people must wait for their mate's arrival.

LLC at t_4 , requiring $A \cup B$ to be written on the same cell array. Then, the write-back traffic in the SPCM system is nearly twice the baseline. This is the main cause of an increase in the number of updated cells, which ultimately leads to shorter memory lifetime and more energy consumption. This energy overhead is higher than the performance gain achieved by the SPCM for most of the applications and must be handled through some additional write management schemes. We can conclude that using a pure SPCM is not worthwhile and we need to find a way to alleviate its energy/lifetime overheads.

We develop additional architectural innovations to the SPCM in order to prolong its lifetime and relieve its energy consumption. We employ a technique to carefully schedule write operations with the main goal of pairing write-back requests with odd and even addresses and send them to the SPCM memory at once as a pair write in order to prevent write-back traffic overhead. By this modification, we expect that the energy consumption and the number of writes per cell become close to those of the baseline system. Another technique used for relaxing write stress on the SPCM (to be discussed in Section 4.4) is adaptively transforming between *striped* mode and *nonstriped* mode, such that the write bandwidth for those lines encountering a high rate of writes becomes smaller with the aim of prolonging the lifetime of the system.

4.3. Pairing Write Queue: P-WRQ

In this section, we first present the abstract concept of our solution through an example. Then, we support our idea with some observations on the efficiency of our solution.

It was mentioned in Section 4.2 that any update in block A and/or B (which are paired) requires writing $A \cup B$. Therefore, both blocks must be written onto the SPCM when one of them is evicted from the LLC for write-back. In this example, we assume blocks A and B are evicted in t_2 and t_4 , respectively ($t_4 > t_2$). We suggest to postpone writing block A until t_4 . Thus, compared to the conventional design, SPCM gets a same-sized write update (occurred at t_4).

Waiting Room. Based on the preceding example, we must keep one block somewhere waiting for its partner block. We call this place “Waiting Room” to convey the abstract concept of waiting, as shown in Figure 6. Once a block is evicted from the LLC, it enters the waiting room to be scheduled for writing. In contrast with conventional write queues, the leaving priority is not in First-Ready First-Come-First-Serve (FR-FCFS) order. Instead, we define a SPCM-aware priority order as follows:

- (1) *Ready*: Both partner blocks are present in the room (marked with “R”). They are ready for pair-write operation.
- (2) *Confused*: One block is in the room (marked with “!”), and its partner block is not in the LLC. In this case, it is not required to wait for the partner block.

Fig. 7. (a) P-WRQ architecture for SPCM system, (b) Percentage of each priority in P-WRQ ejections.

- (3) *Waiting*: One block is in the room (marked with “W”) while its partner block is not, but it can be found in the LLC. Now, the block must wait for its partner block to become dirty and finally evicted from the cache.

We also consider FR-FCFS order as the minor precedence, which means that at each priority, the one that is ready, at first, or comes earlier can go first. Figure 6 clarifies this scheduling by exhibiting a waiting room in the society.

When the write buffer is full, two blocks are selected to leave. If both of them are in *ready* state, the scheduler sends them to the corresponding SPCM bank for writing. When one block is in *confused* state, the memory controller first accesses the SPCM bank to obtain the partner block and buffers it. Then, both blocks are sent for writing. Ultimately, if there is no *ready* or *confused* block, the scheduler sends the first *waiting* block inevitably (at cost of redundant cell updates).

Figure 7(a) illustrates the proposed architecture for a pairing write queue. In this design, we supply a pairing write queue to the SPCM controller on the dual in-line memory module (DIMM) with the eviction policy described before. An arbiter logic is required to determine which block can leave the queue. It requires knowing the state of all buffered blocks. First, it should figure out how many blocks are *ready* (counter “R” indicates the number of *ready* blocks). If there is any, it chooses the first couple to leave (pointers “P1” and “P2” contain the location of chosen *ready* blocks). But, if there is no *ready* block, it gets the number of *confused* blocks from counter “C,” and chooses the first one pointed by “P3.” The partners of *confused* blocks are retrieved from the SPCM and buffered. Finally, if there is no *confused* block either, the arbiter selects the head of the queue (which is *waiting*) and fetches its partner from the cache. Note that, in all transactions including buffer ejection and injection as well as the LLC updates, all counters and pointers are updated in the background.

Using this architecture, we expect that the energy and lifetime overheads incurred by the SPCM system can be recovered. This is achievable when almost all ejected blocks are in *ready* state. Figure 7(b) plots the priority distribution of the ejected blocks from write queue. As shown, about 72% of the ejected blocks are of *ready* type. Based on this observation, the number of writes in the SPCM system with P-WRQ gets close to that of the baseline system.

4.4. Adaptive Morphology

Figure 8(a) depicts quantity of Read After Write (RAW) and Write After Write (WAW) operations for different applications. As can be seen in the figure, they are evenly distributed revealing roughly identical probabilities of reading from and writing on an already written line. Since writing to an SPCM system is costly, we can switch the writing operation between *striped* and *nonstriped* modes. The negative implication of switching is represented in Figure 8(b) in the form of a RAW fraction of total Reads,

Fig. 8. Read/write after write percentages when running different applications: (a) percentage of Read after Write and Write after Write accesses against total access to the main memory, and (b) percentage of Read after Write accesses against total read accesses.

Fig. 9. Write and read intensities and types: (a) write intensity histogram (W x means x times write on a single line), (b) read intensities histogram (R x means x times read from a single line), (c) different types of read operation, and (d) different types of write operation (S: Striped, $\bar{S}$: Nonstriped, R: Read, R $^+$: Reread, horizontal axes in (c) and (d) represent thresholds, N: infinite).

which are requiring slow reads similar to conventional MLCs after switching to *non-striped* mode. Then, we should carefully switch the writing mode so that the majority of subsequent reads from written lines stay in striped mode.

Figure 9(a) represents the amount of write stress on a line in terms of distribution of write intensity. As can be seen in this figure, the vast majority of lines are not under write pressure. On the other hand, Figure 9(b) shows the number of read operations occurring upon a single line in terms of distribution of read intensity. As seen, in those applications exhibiting a large amount of write stress (all W_x where $x > 1$), reread operations (all R_x where $x > 1$) are more often. According to this observation, we propose an adaptive morphology based on the number of writes on a single line. When a write request arrives through DMA from an I/O device (e.g., secondary storage), it would be written in *striped* mode. As soon as the number of write-back requests from the LLC overshoots a predefined threshold, the whole group is changed to *nonstriped* mode. We call this type of writing mechanism directional write, or in short, *diwrite*. When a group or a pair of lines is diwritten and converted to *nonstriped* mode, the ordering mechanism and pair-write operations are no longer required and must be

disabled. Since *diwrite* operation is costly and changing to *nonstriped* mode steps back to slow read operations, we should compromise the switching mode threshold, such that its overheads become as minimal as possible. Figures 9(c) and 9(d) display the impact of changing threshold on the distribution of different access types for read and write operations, respectively. Our goal is to enlarge the portion of striped reads (SR and SR⁺ in Figure 9(c)) while limiting the number of striped writes (S in Figure 9(d)). It can be concluded from these figures that as the threshold increases, the portion of nonstriped reads ($\bar{S}R$ and $\bar{S}R^+$ in Figure 9(c)) dwindles, but on the other hand, the ratio of striped writes (S) grows. We choose threshold 1, as a moderate point, for *diwrite* operation to considerably increase the number of nonstriped writes while keeping most of the read accesses in striped mode. According to Figures 9(c) and 9(d), selecting threshold 1 results in slow reads for only 10% of read accesses, while averagely 73% of writes are performed in *nonstriped* mode.

The metadata required to implement adaptive morphology consist of a mode bit and a counter per each group/pair of lines. The size of the counter depends on the selected threshold. For threshold τ and an SGB n -bit MLC SPCM with B -byte memory lines, $\frac{S}{n \times B} \times \lceil \log_2(\tau + 1) + 1 \rceil$ Gb metadata is needed. For example, for a 4GB 2-bit MLC SPCM with 128B lines, 4MB metadata should be considered. The metadata of both ordering mechanism and adaptive morphology can be stored along with each other in a separate external DRAM chip named metaDRAM.

To write on a line, the SPCM controller first examines the source of the write request. If it is received from I/O for dumping information from secondary storage, the controller sets its mode bit and writes data in *striped* mode. Each write access to a group/pair of lines increments its counter. When the counter reaches the threshold, all lines of the group are retrieved from their most up to date places (SPCM, LLC, RB, or P-WRQ); then, the group is transformed into *nonstriped* lines (reset mode bit) and each line is written on its own cells. Before each read operation, the corresponding metadata must be available first. This extra information incurs a small delay (5ns for reading metadata from a 6MB metaDRAM) in all read operations. Also, rewriting data to reform a group imposes some extra writes. Nonetheless, the long-term impact of using *diwrites* on the SPCM's energy consumption and lifetime is way beneficial. The problem of extra latency can be resolved using the ordering mechanism described in Section 4.1. Overall, an exact trade-off is required to select best thresholds for both ordering mechanism and adaptive morphology. According to our observations, we select 0 and 1 as thresholds for ordering mechanism and adaptive morphology, respectively.

4.5. Modified Read/Write Operations

Enabling all solutions, the read and write operations require some modifications. On the SPCM DIMM, there are several PCM banks, the large DRAM cache, the metaDRAM, and the P-WRQ. PCM banks, which shape the main memory space, are placed in the back end of the access path. The metaDRAM is an auxiliary space preserving the status of each memory line, and should be accessed before the PCM banks. In the middle way, RB and P-WRQ may exclusively contain the most up to date cache line. In the front end of the access path, the DRAM cache is placed to provide the fastest access to the memory space.

The read data path is composed of three subpaths starting at the same time. On a read operation arrival, the SPCM examines all devices on the DIMM to look up the cache line, and also starts retrieving it from memory banks, through these subpaths. If data was found in DRAM cache (first subpath), other subpaths would be ceased, and the row buffer is flushed. Then, the cache line will be transferred from DRAM banks to the memory controller in the CPU side through data bus. If cache line was present

Table I. Simulation Platform Configuration

Processor and On-Chip Memories	
Cores	quad-core, SPARC-III, out-of-order, 2.5GHz, Solaris 10 OS
L1 caches	Split I and D, UCA 32kB private, 32B, LRU, write-back, three-cycle hit
L2 cache	NUCA 4MB shared, eight-way, 64B, write-back, 50-cycle hit
Memory Controller	20 slot read/write/read-response queue buffers
PCM Main Memory	
SPCM Controller [†]	2MB SRAM, 16K slots P-WRQ (6ns, 22mm ²); 2KB SRAM, 20 slots RB (0.7ns, 0.65mm ²), 5.3K two-input NAND gates control logic
metaDRAM [†]	8MB SDRAM, 4Banks $\times$ 1M $\times$ 16b, 5ns
DIMM	667MHz Command rate, DDR3-1333 C6, 64-bit data channel, dual-rank
L3 cache (LLC)	off-chip 128MB DRAM shared, eight-way, 128B, write-back, 63-cycle hit (25ns)
MLC PCM	48ns/40uA/1.6V/5.5nJ read, 40ns/300uA/1.6V RESET, 150ns/150uA/1.2V SET, 15ns/100–996uA/up to 32 iterations partial set, 531.8nJ for all line updates

[†]SPCM Controller and metaDRAM are not used in the baseline system.

in RB or P-WRQ (second subpath), the same thing would happen. Ultimately, the PCM banks (third subpath) will be accessed for acquiring the cache line.

All write requests coming from the memory controller will be redirected to the DRAM cache, the as DRAM main memories. The role of SPCM unfolds when a cache line is evicted from the DRAM cache. As stated in Section 4.3, the evicted cache lines are moved to the P-WRQ to be scheduled for writing on memory banks.

5. EVALUATION RESULTS

Throughout evaluation, we use the same configuration parameters for the baseline system used in similar studies [Qureshi et al. 2009a, 2010a; Jiang et al. 2012c]. Results are presented for different combinations of proposed techniques: P-WRQ refers to the SPCM architecture using a pairing write queue; SWP is representative of the architecture using an ordering/swapping mechanism; and DW shows that *diwrite* operation is considered.

5.1. Evaluation Settings

Infrastructure. We perform microarchitectural level, execution-driven simulation of a processor model with UltraSPARCIII ISA using GEMS [Martin et al. 2005] and Simics toolset [Magnusson et al. 2002]. The simulated CMP runs a Solaris 10 operating system at 2.5GHz. We use CACTI 6.5 [Muralimanohar et al. 2007] to obtain timing, area, and energy estimations for the main memory and caches. Note that we used the approach given in Lee et al. [2009] to adapt CACTI for PCM. For all the components except for the PCM main memory that uses a *Low-Operating-Power* (LOP) process, we use 32nm ITRS models, with a *High-Performance* (HP) process.

System. We model a four-core CMP detailed in Table I. The system has three levels of caches: separated L1 instruction and data caches that are private for each core, an L2 cache that is logically shared among all the cores while physically structured as static NUCA, and an off-chip DRAM cache.

SPCM Controller. SPCM requires a controller to handle read and write operations. We use a 2KB RB within the SPCM controller. To prevent traffic overloading on the memory bus, we assume both the LLC and SPCM controller are placed in the DIMM side. Since the memory bus is physically wrapped between the memory controller (on CPU side) and LLC, the SPCM will not pose unnecessary traffic on the bus.

PCM Main Memory. Our baseline architecture of a 2-bit PCM memory subsystem is shown in Figure 10. Similar to DRAM, PCM is based on a two-sided DIMM with

Fig. 10. Breaking down of the SPCM dual sided DIMM.

two ranks and eight PCM chips per rank. A large DRAM shared cache is used in the form of eight DRAM chips to enervate the problems of long write latency and limited endurance of PCM devices. The DRAM last-level cache has a default size of 128MB. Due to the nondeterminism of MLC PCM write latency, we adopt the *universal memory interface* proposed by Fang et al. [Fang et al. 2011].

We use write pausing with an adaptive write-cancellation policy proposed by Qureshi et al. [2010a] that can pause an ongoing write in order to service a pending read request. Furthermore, we rely on a randomized Start-Gap algorithm [Qureshi et al. 2009a] for low-overhead wear leveling.

For further lifetime and write energy efficiency, a differential write mechanism is enabled. In this scheme, before a PCM block is written, the old value is read and compared to the new data, and only cells that need to change are then programmed [Cho and Lee 2009; Zhou et al. 2009].

Considering the baseline, we choose a granularity of 64B for the write circuit [Hay et al. 2011] to amortize the write driver's area. As such, the baseline needs two rounds to write a line (128B). In the SPCM with 128B data blocks, we use the same hardware and hence need four rounds while writing. There is no area overhead but the energy/latency costs should be accounted. In contrast, we should double up the number of read circuits to prevent latency overheads which impose $2\times$ read energy and area overheads. To avoid these overheads, both ranks on the DIMM should be involved while accessing the SPCM for read/write operations. Both SPCM and baseline use a two-rank DIMM (each rank consists of 8 PCM banks) with the same number of read circuits. However, SPCM considers the two ranks as a single rank of 16 PCM banks, while in the baseline they are completely separated. Therefore, there would be only a negligible hardware overhead for handling chip selection and multiplexing data bus in comparison with the baseline (i.e., the number of read/write circuits in both SPCM and baseline systems are equal).

Organization. Figure 10 represents the organization of the SPCM abstract implementation on a two-sided DIMM considering common form factors. The DIMM consists of two MLC-PCM ranks for both baseline and SPCM systems. Each rank comprises 8 PCM chips that are organized in eight banks. A bank contains $16k \text{ rows} \times 1k \text{ columns} \times 8 \text{ cells} \times 2 \text{ bits}$ providing 32MB memory space. In contrast with DRAM in which the output channels of banks are 8-bit wide, in 2-bit MLC PCM it must be 16-bit wide, because each bank cell has eight PCM cells (for SLC, an 8-bit channel is enough).

However, it is not necessary to double up read circuits, and this change occurs only upon row buffer size and banks output channels. Notably, a PCM chip has a 16-to-1 multiplexer near its output pads such that each two input ports are fed from the same bank (distinct high and low bytes). As shown in Figure 10, a cache line in the baseline system is distributed among eight chips in one rank, while in the SPCM system, two ranks are merged and a cache line is laid on 16 chips on both ranks.

MLC PCM model. Table I summarizes the timing and power characteristics of the modeled 2-bit MLC PCM. We evaluated the MLC design with a resistance distribution that can tolerate resistance drift of at most 1 minute (Figure 1), after which a refresh command is issued. This provides a readout current of 40uA and RESET and SET currents of 300uA and 150uA, respectively. These values are obtained using a model by Kang et al. [2003] that calculates the minimum programming current required for a successful write operation. Following this model, experiments show that program-and-verify takes at most 32 iterations to complete.

Timing Parameters. In nonvolatile main memories, the memory controller does not require restoring data after reading a row. However, the delay between an array read and buffer read/write command (tRCD) in SLC PCM (48ns read and 7.5ns row decode latencies) is much longer than that of DRAM (13.5ns) [Lee et al. 2009]. In MLC PCM, we assume this latency (48ns) is multiplied by the storage density. In other words, in the MLC PCM system, each read iteration takes 48ns, which is reflected in tRCD. Also, tRP, the delay between an array write and a following array read, is assumed to be 150ns [Lee et al. 2009]. Other cell-technology independent latency components such as tCL, tWL, tCCD, tWTR, and tBURST are considered the same as DRAM. For burst transferring of 128B data with tCL (aka tCAS) of 6ns to the memory controller via 64-bit data bus, 16 transfers are required (9ns, 14.25ns, and 20.25ns for first, eighth, and 16th words, respectively). Additionally, we consider a tMETA of 5ns for reading metadata from metaDRAM upon every access. Altogether, when the requested memory line is not cached or buffered, reading a memory line would take 79ns for odd blocks, and 127ns (same as baseline) for even blocks.

Workloads: We used parallel programs in PARSEC-2 suite [Bienia and Li 2009] as multithreaded workloads and a set of programs in SPECCPU2006 benchmarks [Spradling 2007] for multiprogram applications. Particular applications are chosen for their memory intensity and we did not consider a benchmark if its main memory access rate is low in order to better analyze the impact of the memory system on the overall system performance. Also, the selected set of workloads may not stress pathological patterns in write. Regarding input sets, we use *Large* set for PARSEC-2 applications and *Sim-large* for SPECCPU2006 workloads.

We classify benchmarks based on their LLC's *Misses Per Thousand Instructions* (MPKI) when running alone in a four-core system detailed in Table I: each workload is either high miss (H) if MPKI is greater than 10, medium miss (M) if MPKI is between 3 and 10, or low miss (L) if MPKI is less than 3. Table II characterizes the evaluated workloads.

5.2. Energy Evaluation

Figure 11 shows the impact on energy consumption of an SPCM system with and without enabling P-WRQ, SWP, and DW mechanisms. The SPCM system with P-WRQ has only 0.7% energy degradation, on average (1.3% improvement in MP applications and 4.6% degradation in MT programs). One can observe that in workloads where *ready* blocks are dominant, the SPCM with P-WRQ dissipates much less energy compared to the system without P-WRQ (see Figure 7(b)). Highly memory intensive applications

Table II. Characteristics of the Evaluated Workloads

Workload	MPKI	OBHR [†]	Lifetime (years)
SPEC CPU, 2006, Four-Application Multiprogrammed (MP)			
SET1 (mp-hi1): mcf-bzip2-omnetpp-xalancbmk	10.99 (H [‡])	0.78	14.89
SET2 (mp-hi2): milc-leslie3d-gemsfddt-lbm	19.28 (H)	0.93	20.74
SET3 (mp-hi3): mcf-xalancbmk-gemsfddt-lbm	10.77 (H)	0.77	6.7
SET4 (mp-md1): mcf-gemsfddt-povray-perlbenc	7.36 (M [‡])	0.89	4.88
SET5 (mp-md2): mcf-xalancbmk-perlbenc-gcc	7.61 (M)	0.84	7.65
SET6 (mp-md3): gemsfddt-lbm-povray-namd	5.15 (M)	0.72	8.64
SET7 (mp-lo1): gromacs-namd-dealii-povray	0.51 (L [‡])	0.86	35.56
SET8 (mp-lo2): perlbench-gcc-dealii-povray	2.11 (L)	0.97	20.02
SET9 (mp-lo3): namd-povray-perlbenc-gcc	1.01 (L)	0.98	29.84
PARSEC-2, 2009, Multithreaded (MT)			
caneal	6.86 (M)	0.6	5.86
dedup	1.27 (L)	0.99	25.8
facesim	4.91 (M)	0.23	12.7
ferret	3.53 (M)	0.94	13.59
vips	1.63 (L)	0.98	24.03
For all benchmarks			
Min	0.51	0.23	4.88
Max	19.28	0.99	35.56
Average	5.93	0.82	16.49

[†]Odd-Block Hit-Rate (OBHR); average OBHR for all benchmarks is 82% (63% in RB, and 19% exclusively in the LLC).

[‡]L: below 3.0; M: between 3.0 and 10.0; H: above 10.0.

Fig. 11. Impact on energy consumption.

such as *mp-hi1* running on the proposed architecture do not consume more energy than the baseline system, but they achieve improvement mainly because of data holding in P-WRQ. On the other hand, applications like *caneal* (10%) and *ferret* (13%), that have a great number of *confused* blocks, are less influenced by P-WRQ. Overall, combining P-WRQ with the pure SPCM system can save energy. However, enabling SWP worsens energy consumption (by 10%) for imposing extra writes due to data swapping. Since write operations occurring in striped mode demand larger bandwidth, and SWP imposes even more writes, the design is empowered with adaptive morphology to relax extra write energy by constricting the write bandwidth using *diwrite* operations. As shown in Figure 11, the energy consumption of the SPCM system with P-WRQ and DW is dramatically reduced (by 21%). Eventually, the ultimate design with all techniques enabled improves energy consumption by 6.9%.

Fig. 12. Impact of using SPCM on memory access latency.

Fig. 13. IPC improvement using the SPCM system.

5.3. Impact on Performance

P-WRQ has a negligible effect on the memory access latency since all write operations are off-path and pausing them is not costly. Thus, we expect nearly no change in memory access latency, as is shown in Figure 12. In this figure, the SPCM system with P-WRQ achieves an average improvement of 31%, similar to the system without P-WRQ. Also, it can be seen that the SPCM system with P-WRQ and SWP remarkably raises the latency improvement (by about 40%) by enhancing the chance of the most frequently read block to be placed in MSBs. However, toward energy issues, swap mechanism cannot be used without *diwrite* (DW) operations. But then, the sole use of DW causes many of the previously fast reads (single read sequence) to be slow (dual read sequences), and consequently, lowers the latency improvement from 33% in SPCM+PWRQ down to 19% in SPCM+PWRQ+DW, on average. The final design has a latency gain of 25% for memory accesses.

Figure 13 also shows the overall system performance improvement. We can observe that using the proposed techniques does not harm performance improvement while saving energy. Although using DW negatively influences the IPC gain (from 11% in SPCM+PWRQ+SWP to 6.5% in SPCM+PWRQ+DW), the overall speedup is still considerable (9%). Generally, MPKI plays a key role in IPC improvement. Applications with higher values of MPKI would take more benefits from SPCM in terms of performance. For example, benchmarks *mp-hi1* (MPKI=10.99) and *mp-hi2* (MPKI=19.28) exhibit more than 20% IPC improvement. This is while the performance gain in applications with smaller values of MPKI like *mp-lo1* (MPKI=0.51) and *mp-lo3* (MPKI=1.01) shows less than 5% improvement.

Fig. 14. The impact of using SPCM on EDP.

5.4. EDP Results

With the energy saving and performance gain of the SPCM system, we expect improvements in total Energy-Delay Product (EDP). As illustrated in Figure 14, the SPCM system achieves up to 65% reduction in EDP and this improvement is more noticeable on benchmarks with more memory accesses. The main reason is that the performance gain in these benchmarks is large enough to hide the negative impact of energy loss. For instance, while *dedup* shows an increased energy consumption of about 20%, effective memory access latency improvement of about 34% results in 48% enhancement of EDP. Finally, benchmarks for which both energy and performance are improved (such as *mp-hi2*) demonstrate considerable EDP gains. The great impact of SWP and DW is completely clear in Figure 14. In addition to the large amount of performance gain of SWP, DW reduces the energy consumption causing more reduction in EDP. As shown in this figure, the SPCM system reduces EDP by about 20%.

5.5. Impact on Lifetime

In addition to energy consumption, lifetime is another important concern in PCMs. Unfortunately, pairing two memory lines in a 2-bit MLC cell array has an adverse impact on the overall PCM lifetime. On the other hand, increasing system performance causes higher write bandwidth even though both have an identical number of writes. Lifetime is inversely proportional to the write bandwidth. In our experiments when an ideal wear leveling (e.g., those in Qureshi et al. [2009a] and Seong et al. [2010a]) is used, we observed that the SPCM system shortens the average life span by about 24%. On the other hand, the SPCM system improves the performance without decreasing the number of writes leading to a higher number of writes per time unit. Moreover, each write operation on the SPCM system occurs at double the number of MLCs in comparison with conventional systems. These cause a sudden growth of write bandwidth, which shortens the PCM's lifetime. To remove this overhead, P-WRQ manages write-back requests such that the former problem is almost resolved. Additionally, an adaptive morphology mechanism is set to push the writing operation back to nonstriped mode to overcome the latter problem.

Figure 15 shows the impact of SPCM on the system lifetime. The baseline system is presumed to have an ideal lifetime and the impact of using each approach is normalized to it. As is presented, the SPCM system worsens the system lifetime by almost 50%. Nonetheless, P-WRQ makes the lifetime overhead much shorter (76% of ideal lifetime). The SWP mechanism significantly improves memory access latency, though it imposes extra write operations (for swapping), and as a result, shows more reduction in lifetime (62%) in comparison with the sole use of P-WRQ. In contrast, DW makes the write bandwidth smaller with the aim of prolonging SPCM lifetime, costing some

Fig. 15. The impact of using SPCM on lifetime.

performance loss. Since the lifetime and performance in this design mutually affect each other, DW also degrades IPC to extend the life span (up to 95% of its ideal lifetime). Therefore, it can be concluded that using a SPCM memory system with all supplementary techniques is more performance and energy efficient than the baseline while degrading the nominal lifetime by only 8%.

6. SCALABILITY ANALYSIS

The SPCM is a general approach and can be also applied to higher bit storage level PCMs. This means that more than two blocks can be grouped together and stored in one MLC array in a striped manner. In this section, we utilize different models to observe the impact of using an N -bit SPCM system, equipped with the P-WRQ unit, on different metrics and costs.

6.1. Read Latency Improvement

We develop an analytical model to investigate the impact of storage density on the average memory latency of the SPCM, considering a normal distribution for all possible data patterns. Foremost, a recursive relation can be given as $L(1) = P \times L_B + (1 - P) \times L_M$ to calculate the latency for extracting the first bit of the cell (the MSB), and $L(n) = P \times L_B + (1 - P) \times (L_M + L(n - 1))$ for lower bits, where n is the bit number in the cell (from the next more significant bit to the LSB). The first term declares that when a desired block is in the RB or the LLC (with a probability of P), it can be retrieved in L_B cycles (for simplicity we assumed a fixed L_B , but it depends on the latencies of LLC and RB). The second term states that if it is not cached or buffered, we need to spend L_M cycles in order to read the block from the SPCM, and in the case of not being the MSB, $L(n - 1)$ additional cycles are also required for extracting the upper $n - 1$ bits (as the reference for read circuit). Then, for simplification, we describe the average of expected values for our model as

$$L_{spcm} = \frac{1}{N} \sum_{i=1}^N \left(\frac{1 - (1 - P)^i}{P} \times (P \times L_B + (1 - P) \times L_M) \right), \quad (1)$$

where N is the cell density level, L_B is the average latency of accessing the buffer and cache, L_M is the latency of one iteration of reading from MLC PCM (120 cycles), and P is the probability of a block being buffered or cached. On the other hand, it is clear that the average latency of accessing the memory in the baseline system can be given by

$$L_{base} = P \times L_B + (1 - P) \times L_M \times N, \quad (2)$$

which means that if the block was cached or buffered with a probability of P , it can be obtained in L_B cycles; otherwise, $N \times L_M$ cycles would be required. Figure 16(a) plots

Fig. 16. (a) Analytical estimation for memory access latency improvement and (b) energy overhead in 2-, 3-, and 4-bit MLC striped PCM with P-WRQ.

curves of the expected memory access latency ratio ($1 - L_{spcm}/L_{base}$) for different cell density levels (i.e., $N = 2, 3$, and 4).

When $P = 0$, half of the blocks (those at odd addresses) can be extracted within 50% of the total required latency for a 2-bit MLC, which means 25% overall latency improvement as shown in Figure 16(a). As P increases, we achieve more improvement in access latency until $P < 0.9$. However, when $0.9 < P < 1.0$, most of read requests are serviced in the LLC and RB, and therefore, we have less latency gain by using SPCM.

6.2. Write Energy Overhead

In the case of using more than two bits per cell, the ejection priority order in P-WRQ should be modified. In this case, the most prior group to be written is the one whose partners are all in the queue and the next priorities are those that have less number of buffered partners. This, of course, imposes latency overhead in the arbiter logic that cannot be mathematically modeled and requires synthesis evaluations to be obtained. For the energy overhead, however, we can write

$$E_{spcm} = P \times E_B + (1 - P) \times \sum_{i=1}^N \binom{N}{i} P^{N-i} (1 - P)^i E_M, \quad (3)$$

where N is the storage level; E_B is the energy for writing a block inside the buffer (P-WRQ), which is about 100.3pJ; E_M is the required energy for writing an N -bit MLC array consisting of N cells, which is 272.5pJ per cell; and P is the probability of a block being in the queue. The first term shows that in the case of the block being buffered (with a probability of P), the write operation is redirected to P-WRQ, which dissipates E_B joules. The second term states that if the block was not buffered, for each partner of the group that is not in the buffer, one write operation is required (E_M joules). Based on this model, if none of the partner blocks is in the P-WRQ ($P = 0$), $N \times E_M$ joules would be consumed to write them back separately.

For comparison, the energy consumption in the baseline system can be estimated as

$$E_{base} = P \times E_B + (1 - P) \times \frac{E_M}{N}, \quad (4)$$

Fig. 17. The result of sensitivity analysis over the P-WRQ size versus LLC capacity in (a) 3D and (b) 2D forms. (c) Latency/Energy costs for different P-WRQ sizes. Selected design corners are shown in (b) as P_x where x is the size of LLC.

where the first term indicates that if the block is in the buffer, it would dissipate E_B joules. The second term states if it is not buffered, E_M/N joules would be consumed. Note that in the baseline, an S -bit block lies on S/N number of MLCs. Figure 16(b) plots the expected memory access energy ratio curves (E_{spcm}/E_{base}) as a function of P .

As we discussed before, when $P = 0$, the energy consumption is exactly $N^2 \times$ higher than the baseline since the number of updated cells in the striped PCM system is $N \times$ greater than that of the baseline system and each partner block is written back in N different. Obviously, if $P = 1$, we have no energy overhead because all write-backs occur at once. However, we probably have some improvement when $0.75 < P < 1.0$ because not only most of the partners that are ejected from the queue have been written in time, but also some writes are handled in the buffer. Figure 11 shows that for benchmarks with medium rate of P (see Table II for OBHRs) we have some energy improvement.

7. RELATED WORK

In this section, we focus on important related studies concerning innovative techniques to favorably affect overall system performance and PCM lifetime.

To reduce the latency of the system's critical path at the architectural level, especially for read operation, certain topical microarchitectures tend to take advantage of MLCs high density along with immediate reaction to read requests. Among them, two classes can be categorized: first, based on the resume capability of the iterative program-and-verify process, the Write Pausing and Write Cancellation schemes are proposed in Qureshi et al. [2010a], which try to pause/cancel an ongoing write in support of prompt response to a superior read request directed to the same memory bank; second, aggregating benefits of both SLC and MLC in a storage system, strategies in Dong and Xie [2011] and Qureshi et al. [2010b] try to dynamically transmute between these two states. Dong and Xie [2011] proposed AdaMS, an adaptive MLC/SLC PCM design for file storage, which focuses on exploiting fast SLC response speed and high MLC density with the consciousness of workload characteristic and lifetime requirement. MMS, a Morphable Memory System [Qureshi et al. 2010b], also decides on bit-storage level of the physical memory page and subsequent recent transactions. To this end, a monitoring scheme determines whether to operate at lower cell bit capacity in order to obtain faster response time during a phase of low memory usage, or at high-density MLC when the program working set is large.

More recently, Yoon et al. [2014] proposed a data mapping and buffering technique for MLC PCM, which similarly decouples two bits of a cell and assigns each of them to either fast-read/slow-write or slow-read/fast-write regions, through a hardware/software technique. There are also some circuit-level approaches such as early/turbo reads [Nair et al. 2015] that are originally proposed to reduce sensing

latency of SLC PCMs by modifying read circuitry of PCM subarrays. Although they target two-level PCM cells, their approach can also be applied to four-level cells. SPCM can gain even more speedup in system performance, providing that it uses early and turbo reads.

There are some researchers that pursue architectural techniques to reduce program-and-verify iterations. For example, Jiang et al. [2012c] and Qureshi et al. [2010a] discriminate between the way of programming based on data pattern and previously stored data values. The experiments confirm that a proper decision on the write scheme selection can lead to a striking drop in program iterations, which sequentially causes enhancements in overall latency, energy, and write endurance of the nonvolatile memories.

In the case of lifetime improvement, Qureshi et al. proposed Start-Gap [Qureshi et al. 2009a], a simple but efficient analytical wear leveling technique to prevent too many write repetitions on a particular memory region.

Taking advantage of asymmetric read latencies in multibit storage systems is another insight to improve system performance. Jiang et al. [2012b] proposed a large and fast MLC STT-MRAM-based cache for embedded systems where two physical cache lines are combined and rearranged to construct one read-fast-write-slow (RFWS) line and one read-slow-write-fast (RSWF) line. Then, a swapping mechanism is applied for mapping write-intensive data blocks to RSWF lines and read-intensive blocks to RFWS lines. Striping was also used in NAND-FLASH storage systems in different forms (vertical, horizontal, and two-dimensional) for page alignment and combining physical pages from several dies to enlarge the granularity of flash arrays [Grupp et al. 2012]. However, while multilevel STT-MRAM and Flash cells require modifications on fabrication process for different bit-storage densities, the structure of MLC PCM would not be changed. Yoon et al. [2013] published a technical report presenting a new data mapping scheme for MLC PCM similar to what we proposed in this work to exploit the read latency asymmetry. Although the durability is one of the most important PCM concerns, they did not address the problem of doubling up the write bandwidth. Another complexity of the technique reported in Yoon et al. [2013] is that the OS must be aware of hardware for mapping policy.

8. CONCLUSION

Among nonvolatile memory systems, PCM is a promising candidate as an alternative to DRAM for increasing main memory capacity especially in MLC mode. However, one of the main challenges in an MLC PCM system is the linear increase in read access latency with respect to the cell storage level. In this article, we took benefit of asymmetry in the read mechanism and proposed a striping scheme for data alignment in a 2-bit MLC PCM (SPCM). Additionally, a swapping mechanism was proposed to order data lines in a group with respect to their read frequency, such that more frequently read lines are placed in more significant bits stored in cells. Then, we augmented our design with some supplementary microarchitectures to prevent unnecessary write pressure on SPCM. We also evaluated SPCM and a priority write queue (to reduce the write stress) with different configurations. Our experiments showed significant improvements in system performance (by more than 24% of read latency and 9% of IPC) and energy-delay product (by about 20%) for a small lifetime degradation of 8%, on average.

REFERENCES

- Alan Allan, Don Edenfeld, William H. Joyner, Jr., Andrew B. Kahng, Mike Rodgers, and Yervant Zorian. 2002. 2001 technology roadmap for semiconductors. *Computer* 35, 1 (2002), 42–53.

- Ferdinando Bedeschi, Rich Fackenthal, Claudio Resta, Enzo Michele Donze, Meenatchi Jagasivamani, Egidio Cassiodoro Buda, Fabio Pellizzer, David W. Chow, Alessandro Cabrini, G. Calvi, R. Faravelli, A. Fantini, G. Torelli, D. Mills, R. Gastaldi, and G. Casagrande. 2009. A bipolar-selected phase change memory featuring multi-level cell storage. *IEEE Journal of Solid-State Circuits* 44, 1 (Jan. 2009), 217–227.
- Christian Bienia and Kai Li. 2009. PARSEC 2.0: A new benchmark suite for chip-multiprocessors. In *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*.
- Sangyeun Cho and Hyunjin Lee. 2009. 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 (MICRO'09)*. 347–357.
- Youngdon Choi, Ickhyun Song, Mu-Hui Park, Hoeju Chung, Sanghoan Chang, Beakhyoung Cho, Jinyoung Kim, Younghoon Oh, Duckmin Kwon, Jung Sunwoo, and others. 2012. A 20nm 1.8 V 8Gb PRAM with 40MB/s program bandwidth. In *Proceedings of the 2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC'12)*. IEEE, 46–48.
- Gael F. Close, Urs Frey, Jack Morrish, Ramiro Jordan, Simon John Geoffrey Lewis, Tom Maffitt, M. BrightSky, Christoph Hagleitner, Chris Lam, and Evangelos Eleftheriou. 2013. A 256-Mcell phase-change memory CHIP operating at bit/cell. *IEEE Transactions on Circuits and Systems I: Regular Papers* 60, 6 (2013), 1521–1533.
- Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. 2009. Better I/O through byte-addressable, persistent memory. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP'09)*. 133–146.
- Xiangyu Dong and Yuan Xie. 2011. AdaMS: Adaptive MLC/SLC phase-change memory design for file storage. In *Proceeding of the 2011 16th Asia and South Pacific Design Automation Conference (ASP-DAC'11)*. 31–36.
- Kun Fang, Long Chen, Zhao Zhang, and Zhichun Zhu. 2011. Memory architecture for integrating emerging memory technologies. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT'11)*. 403–412.
- Laura M. Grupp, John D. Davis, and Steven Swanson. 2012. The bleak future of NAND flash memory. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*. USENIX Association, 2–2.
- Andrew Hay, Karin Strauss, Timothy Sherwood, Gabriel H. Loh, and Doug Burger. 2011. Preventing PCM banks from seizing too much power. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'11)*. 186–195.
- Stephen Hudgens and Brian Johnson. 2004. Overview of phase-change chalcogenide nonvolatile memory technology. *MRS Bulletin* 29 (Oct. 2004), 829–832.
- Lei Jiang, Youtao Zhang, Bruce R. Childers, and Jun Yang. 2012a. FPB: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'12)*. 1–12.
- Lei Jiang, Bo Zhao, Youtao Zhang, and Jun Yang. 2012b. Constructing large and fast multi-level cell STT-MRAM based cache for embedded processors. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference (DAC'12)*. IEEE, 907–912.
- Lei Jiang, Bo Zhao, Youtao Zhang, Jun Yang, and B. R. Childers. 2012c. Improving write operations in MLC phase change memory. In *Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture (HPCA'12)*. 1–10.
- M. Joshi, Wangyuan Zhang, and Tao Li. 2011. Mercury: A fast and energy-efficient multi-level cell based Phase Change Memory system. In *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA'11)*. 345–356.
- Dae-Hwan Kang, Dong-Ho Ahn, Ki-Bum Kim, J. F. Webb, and Kyung-Woo Yi. 2003. One-dimensional heat conduction model for an electrical phase change random access memory device with an 8F2 memory cell (F=0.15um). *Applied Physics* 94, 5 (Sept. 2003), 3536–3542.
- K. Kilbok. 2007. Main memory technology direction. In *WinHec*.
- Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting phase change memory as a scalable DRAM alternative. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*. 2–13.
- Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hallberg, Johan Hogberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. 2002. Simics: A full system simulation platform. *Computer* 35, 2 (Feb. 2002), 50–58.
- Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. 2005. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *ACM SIGARCH Computer Architecture News* 33, 4 (Nov. 2005), 92–99.

- Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. 2007. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'07)*. 3–14.
- Prashant J. Nair, Chiachen Chou, Bipin Rajendran, and Moinuddin K. Qureshi. 2015. Reducing read latency of phase change memory via early read and Turbo Read. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 309–319.
- T. Nirschl, J. B. Phipp, T. D. Happ, G. W. Burr, B. Rajendran, M.-H. Lee, A. Schrott, M. Yang, M. Breitwisch, C. F. Chen, E. Joseph, M. Lamorey, R. Cheek, S. H. Chen, S. Zaidi, S. Raoux, Y. C. Chen, Y. Zhu, R. Bergmann, H. L. Lung, and C. Lam. 2007. Write strategies for 2 and 4-bit multi-level phase-change memory. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM'07)*. 461–464.
- Moinuddin K. Qureshi, Michele M. Franceschini, and Luis Lastras-Monta. 2010a. Improving read performance of Phase Change Memories via Write Cancellation and Write Pausing. In *Proceedings of the 2010 IEEE 16th International Symposium on High Performance Computer Architecture (HPCA'10)*. 1–11.
- Moinuddin K. Qureshi, Michele M. Franceschini, Luis A. Lastras-Montaño, and John P. Karidis. 2010b. Morphable memory system: A robust architecture for exploiting multi-level phase change memories. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. 153–162.
- Moinuddin K. Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. 2009a. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'09)*. 14–23.
- Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, and Jude A. Rivers. 2009b. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*. 24–33.
- Stuart Schechter, Gabriel H. Loh, Karin Straus, and Doug Burger. 2010. Use ECP, not ECC, for hard failures in resistive memories. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. 141–152.
- Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. 2010a. Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA'10)*. 383–394.
- Nak Hee Seong, Dong Hyuk Woo, Vijayalakshmi Srinivasan, Jude A. Rivers, and Hsien-Hsin S. Lee. 2010b. SAFER: Stuck-at-fault error recovery for memories. In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'10)*. 115–124.
- Young Hoon Son, O. Seongil, Yuhwan Ro, Jae W. Lee, and Jung Ho Ahn. 2013. Reducing memory access latency with asymmetric DRAM bank organizations. *ACM SIGARCH Computer Architecture News* 41, 3 (2013), 380–391.
- Cloyce D. Spradling. 2007. SPEC CPU2006 benchmark tools. *ACM SIGARCH Computer Architecture News* 35, 1 (Mar. 2007), 130–134.
- Jeffrey Stuecheli, Dimitris Kaseridis, Hillery C. Hunter, and Lizy K. John. 2010. Elastic refresh: Techniques to mitigate refresh penalties in high density memory. In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'10)*. 375–384.
- Doe Hyun Yoon, Naveen Muralimanohar, Jichuan Chang, Parthasarathy Ranganathan, Norman P. Jouppi, and Mattan Erez. 2011. FREE-p: Protecting non-volatile memory against both hard and soft errors. In *IEEE 17th International Symposium on High Performance Computer Architecture (HPCA'11)*. 466–477.
- Hanbin Yoon, Justin Meza, Naveen Muralimanohar, Norman P. Jouppi, and Onur Mutlu. 2014. Efficient data mapping and buffering techniques for multilevel cell phase-change memories. *ACM Transactions on Architecture and Code Optimization (TACO)* 11, 4 (2014), 40.
- HanBin Yoon, Naveen Muralimanohar, Justin Meza, Onur Mutlu, and Norman P. Jouppi. 2013. *Techniques for Data Mapping and Buffering to Exploit Asymmetry in Multi-Level Cell (Phase Change) Memory*. SAFARI Technical Report No. 2013-002. Carnegie Mellon University, Pittsburgh, Pennsylvania.
- Gang Zhang, Zhe Wu, Jeung hyun Jeong, Doo Seok Jeong, Won Jong Yoo, and Byung ki Cheong. 2012. Modified write-and-verify scheme for improving the endurance of multi-level cell phase-change memory using Ge-doped SbTe. *Solid-State Electronics* 76, (2012), 67–70.
- Wangyuan Zhang and Tao Li. 2009. Characterizing and mitigating the impact of process variations on phase change based memory systems. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'09)*. 2–13.

- Wangyuan Zhang and Tao Li. 2011. Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system. In *Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems and Networks*. 197–208.
- Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*. 14–23.

Received April 2015; revised September 2015; accepted September 2015