Exploiting Subarrays Inside a Bank to Improve Phase Change Memory Performance

Jianhui Yue, Yifeng Zhu

Electrical and Computer Engineering, University of Maine
jianhui.yue, yifeng.zhu@maine.edu

***Abstract*—Enabling subarrays reduces memory latency by allowing concurrent accesses to different subarrays within the same bank in the DRAM system. However, this technology has great challenges in the PCM system since an on-going write cannot overlap with other accesses due to large electric current draw for writes. This paper proposes two new mechanisms (PASAK and WAVAK) that leverage subarray-level parallelism to enable a bank to serve a write and multiple reads in parallel without violating power constraints. PASAK exploits the electric current difference between writing a bit 0 and a bit 1, and provides a new power allocation strategy that better utilizes the power budget to mitigate the performance degradation due to bank conflicts. WAVAK adds a simple coding method that inverts all bits to be written if there are more zeros than ones, with a goal to reduce electric current for writes and create larger power surplus to serve more reads if there is no subarray conflict. Experimental results under 4-cores SPEC CPU 2006 workloads show that our proposed mechanisms can reduce memory latency by 68.7% and running time by 34.8% on average, comparing with the standard PCM system. In addition, our mechanisms outperform Flip-N-Write 14.6% in latency and 8.5% in running time on average.**

I. Introduction

Better process scalability and less leakage power make Phase-Change Memory (PCM) a promising alternative or supplement to DRAM. In PCM, writes and reads are very different: writes are much slower than reads and take more electric current. When a PCM storage cell is written, large electric current is drawn to heat up the cell's GST material to change its phase and resistance. Compared with reads that simply measure the resistance of cells, writes require a larger electric current pulse for a much longer duration. However, the instantaneous supply of electric current to each PCM bank is restricted in order to control signal noise ratio on bit lines. As a result, the write bandwidth of a PCM bank is limited.

A high-density memory bank usually consists of multiple subarrays with private hardware components that can independently decode memory address and access memory cells. However, during a bank conflict when multiple requests access the same bank, these requests are served one after another. Very recently, Ref. [1] demonstrated that in DRAM allowing concurrent accesses to different subarrays within the same DRAM bank can significantly improve the memory performance in a very cost-effective way.

A PCM cell array also includes multiple subarrays. However, optimization via subarrays in PCM is more challenging than DRAM. First of all, writes in PCM are much slower and takes more electric current than writes in DRAM. Second, the instantaneous current supply to a PCM bank is limited due to the constraints of signal noise ratio on bit lines. Therefore the maximal number of bits that can be written in parallel is limited to 32 or 64 typically [2]. Compound by large write electric current and limited power supply in a bank, a PCM bank has no power surplus to serve other memory accesses when a write is being performed in this bank. When multiple requests with no subarray conflict go to the same bank, PCM often cannot simultaneously serve them, thus sacrificing subarray-level parallelism.

In order to leverage subarray-level parallelism, we propose two new power management techniques for the PCM system, called PASAK and WAVAK, which fully overlap a write operation with read operations that access different subarrays within the same bank, and accordingly reduce negative performance impact of slow writes on time-critical reads in PCM. We note that during a write, conventional memory controller conservatively estimates the required electric current by assuming that each bit to be written takes the same amount of electric current as writing a bit of zero. In fact, writing a zero bit takes much more electric current than writing a one bit. This leaves some instantaneous power budget unused, causing key hardware components underutilized. In PASAK, exploiting current difference between writing bit 0 and bit 1, the memory controller obtains an accurate-and-tight requirement of consumed electric current depending on the number of zeros and ones in the data block. Therefore a bank has power budget surplus to serve other requests that have no subarray conflicts. In addition, before the write completes, PASAK dynamically grants reads according to available power balance as long as there is no subarray conflict in a bank.

Our second technique (WAVAK) is based on the observation that a write data block often contains more bits of zeros than ones. WAVAK adds a simple bit-inverse scheme to reduce the number of zero bits that are more power-consuming to write. WAVAK simply inverts all bits of a data block to be written if there are more zero bits in this block. In this way, it has a larger power balance to serve more read requests in parallel when a bank is executing a slow write, achieving a larger degree of subarray-level parallelism than PASAK.

We evaluate our proposed techniques by simulating a multi-core system under the SPEC CPU 2006 benchmark suite. Experimental results under 13 multi-programmed workloads show that PASAK and WAVAK reduce the read latency by 59% and 68.7% on average over a standard PCM baseline, respectively. These read latency reductions translate to a running time decrease of 31% and 34.8% respectively. In addition, WAVAK is superior to Flip-N-Write [14], a recently proposed PCM write optimization strategy, with 14.6% latency reduction and 8.5% running time reduction.

The rest of paper is organized as follows. Section II introduces the background of PCM and subarrays in a bank. Section III and Section IV present our PASAK and WAVAK schemes respectively. Section V discusses the experimental results. Related work is summarized in Section VI and conclusions are given in Section VII.

II. Background

A. Phase Change Memory

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.

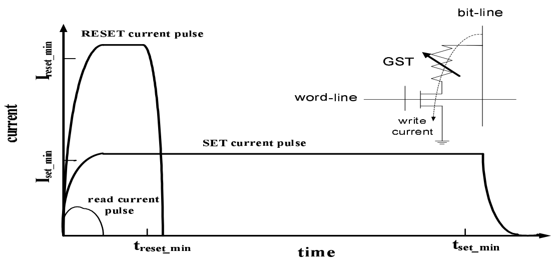

Fig. 1. Writing to a PCM Cell

Comparing with a bit of one, writing a bit of zero requires a much larger electric current but takes a much shorter amount of time, as show in Fig. 1. When bit 0 is written, a large current has to be applied to the target cell for a short duration in order to heat the GST abruptly and move it to the amorphous phase. On the other hand, when bit 1 is written, a relatively smaller current is applied to the cell for a longer duration to slowly heat up the GST and leave it in the crystalline phase. For example, ratio of the electric current to write bit 0 to current to write bit 1 is 2 in Ref. [3], 3 in Ref. [4], and 2 in Ref. [5]. By contrast, reading data does not need to melt the phase change material, and thus required electric current is much smaller than writing.

Meanwhile, conventional PCM controller also conservatively assumes a bit to be written takes the same amount of electric current as writing a bit 0. As a result, it is frequent that a less number of bits are written in parallel, resulting in underutilization of hardware resource.

B. PCM Power Delivery Constrains

Write performance is also limited by the electronic circuit's constraints in a PCM chip. As discussed previously, writing a 0 requires a large current to heat GST. The Dickson charge pump is widely used in PCM chips to provide electric current to the write driver with low power efficiency [6]. The noise on the power line prevents the charge pump from providing a high level of instantaneous current needed for write [6]. In addition, the noise constraints of charge pump also limit the number of cells that can be concurrently written to 2, 4, and 8 typically in a chip [6].

As discussed earlier, conventional write scheme often overestimate the electric current requirement by assuming all bits to be written are zero. This conservative scheme amplifies the negative impact of power delivery constraints.

C. Subarray and Subarray-level Parallelism in DRAM

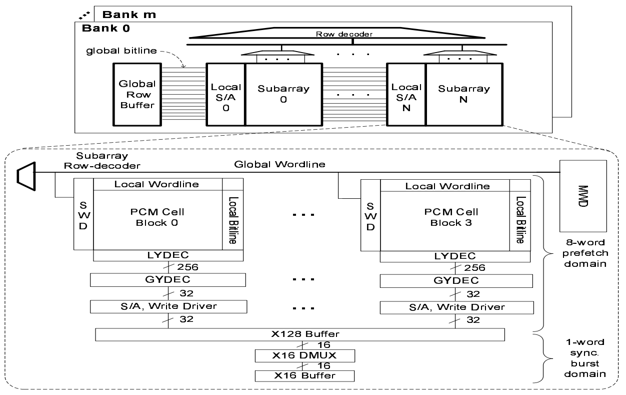

Fig. 2. Subarray in a Bank

A cell array in a DRAM bank includes a very large number of rows and columns. For example, a bank can have 32k rows, with 8k cells in each row. During a memory access, the row address and column address choose a specified word line and bit lines in a cell array. Long bit lines are required in a high-density bank. However, the large parasitic capacitance associated with long bit lines not only causes unreliable data sensing but also increases the latency for activation and precharge command. In order to circumvent these drawbacks, all cells in a bank are organized into a two-dimensional array of cell blocks as shown in Figure 2. Each block has its local bit lines, local word lines, local sense amplifiers(S/A), and write drivers. In addition, each block has a local row column decoder(LYDEC), a sub-wordline driver (SWD), and a local row decoder. Multiple blocks constitute a subarray and a bank consists of several subarrays. The global row buffer relies the signals from local S/A to I/O drivers through global bit lines, which can improve these signal quality.

Since each subarray has its own local data path and sense amplifiers to access data cells, a recent research [1] enhances DRAM banks by adding more data path control gates and latches for local and global address information, and introduces extra commands to specify the working subarray in order to exploit the subarray-level parallelism inside a bank. With the above enhancement, a bank can overlap activation of one subarray with precharging, write-recovery and activation on other subarrays inside the same bank.

Similar to DRAM, a high storage density PCM array also has multiple subarrays for two important reasons [2], [6].

First, without subarrays, bit lines and word lines become too long and then associated parasitic resistances are then too large. This not only reduces the cell sensing speed due to large RC delay, but also decreases the heating efficiency to PCM cell during write. Second, sharing SWD, S/A, W/D and other resource with adjacent subarray makes chip layout more efficient and simpler.

III. Power Allocation Scheme for PCM Subarrays

We propose a new power allocation scheme for a PCM bank to better leverage subarray-level parallelism without violation of power budget. This power allocation scheme is based on the following three insights about PCM. First, writing bit 1 consumes less current than writing bit 0 in PCM. It is reported that the current required to write bit 1 is half of the current to write bit 0 [7]. Second, reading PCM cells requires much less current than writing. In some devices, the current to read PCM is only 1/15 of the current to write bit 0 [7]. Third, conventional memory controller uses a static strategy that tends to overestimate the current requirement when writing a data block. It conservatively assumes each bit in the write request consumes the same current as writing bit 0, even though not all bits are 0 typically.

In our power allocation scheme, the memory controller enforces a dynamic and accurate power allocation for each memory access. Before writing a data block, the memory controller counts the number of zero bits and one bits in the target data block and computes the current requirement based on pre-measured current requirement for writing a 0 and a 1, which are given parameters by PCM manufactures. Since not all bits in a write request are 0, this current estimation usually is much smaller than the conventional scheme, which assumes that each bit to be written is 0. We call this power allocation scheme for subarray-enabled banks PASAK. Suppose the current for writing a bit 0 is I_{reset}, the current for writing a bit 1 is I_{set}, and the write block has a total of N bits, including N_0 bits of zero, the total current required to write the data in the conventional scheme and PASAK can be calculated as follows.

$$I_{conventional} = N \cdot I_{reset} \quad (1)$$
$$I_{PASAK} = N_0 \cdot I_{reset} + (N - N_0) \cdot I_{set} \quad (2)$$

When a write operation is performed, our allocation scheme uses electric current surplus of a PCM bank to serve performance-critical reads as long as there is no subarray conflict. This allocation scheme is safe since it does not violate the power budget for each PCM bank.

The implementation of PASAK is simple. The memory controller maintains a current balance (CB) for each bank. Initially, each bank's CB is set as the maximal current provision for a bank. Before issuing a memory access to a bank, the memory controller checks whether its CB is sufficiently large and decides whether to grant the access or postpone it. After an access is issued, the memory controller subtracts the access's current requirement from the corresponding CB. When an access completes, the memory controller adds this access's current requirement back to the CB.

PASAK's bit counter circuits is simpler than the similar one used in the Flip-N-Write, which computes the difference (hamming distance) between the old data and the new data on writing data. In addition, compared with Flip-N-Write, PASAK has no data inversion part. Therefore, PASAK has smaller latency and less power consumption than Flip-N-Write.

Figure 3 gives a simple example to compare PASAK and MASA, which was proposed recently [1]. Assume a write request and a read request arrive at the time instant t_0 and t_1, respectively. At t_1, MASA blocks the read request since the memory controller has already allocated the whole power budget of the target bank to the previous write. This leads to zero CB since the memory controller does not examine of the content of the target write data and simply assumes all bits to be written are 0s. However, our PASAK examines each bit in the write data and generates an accurate estimation of the required electrical current for the write request. This makes this bank's CB large enough to serve the read request issued at t_1 without any delay.

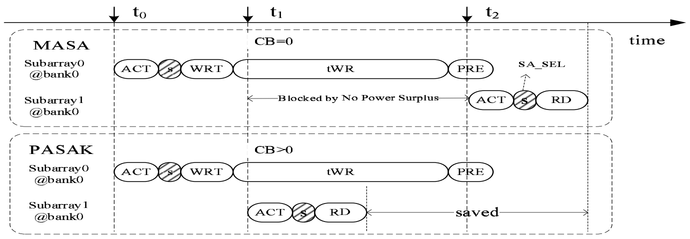

Fig. 3. Timeline of two requests accessing different subarrays of the same bank

PASAK is similar to the power token scheme [8] but has the following differences. First, while the power token scheme only manages the power consumption for writes in PCM, our scheme manages not only writes but also reads. Second, the power token scheme estimates the requirement of electric current for each bit without differentiating bit 0 and 1. Our scheme considers their difference, and accurately computes the electric current required for a given data block to be written. Third, while the power token scheme only considers the aggregate power budget at the system level, our scheme considers the power limitation specified in real PCM chips [2], [6] and enforces the power budget at the individual bank level, instead of the system level. This finer granularity helps us ensure the operation safety of each PCM bank.

IV. Write Data Inversion for PCM Subarrays

In the previous section, we present PASAK that leverages the current difference between writing a bit 1 and a bit 0 to perform accurate-and-tight current budgeting, which allows us to use surplus current budget to serve reads when a write is served. In this section, we propose a new write data coding scheme, called WAVAK, to reduce the current demand when writing a data block. WAVAK allows more reads to be served in parallel with a write, thus increasing subarray-level parallelism.

We note that the number of 0s and 1s in a data block are frequently unequal. Especially, it is observed that there are often more 0s than 1s. Previous research work [9] has observed this skewed distribution and used it to optimize PCM performance. As mentioned earlier, writing bit 0 takes more current than writing bit 1. Based on these two key observations, we propose a coding scheme to reduce the current consumption of a write.

The key idea of WAVAK is to invert all bits of the target data if there are more 0s than 1s in a write request. A flag bit is added and stored in PCM to indicate whether its corresponding data block has been inverted or not. When reading a data block from PCM, the memory controller inverts this block if its flag bit is set. The block size is set as the size of a cache line, which typically is 64 bytes, i.e., 512 bits. Thus the space overhead due to flag bits is very small. In addition, since write operations are not in the critical path and accordingly the performance overhead generated by data inversion is almost negligible. Compared with Flip-N-Write, a recently proposed technique for PCM, WAVAK has much less performance and energy overhead since WAVAK does not need to read original data from PCM during a write. For each write request, Flip-N-Write always reads original data out from PCM, and compare the new and old data to decide flip or not.

The WAVAK's writing current requirement is listed as follows.

$$I_{WAVAK} = \begin{cases} N_0 \cdot I_{reset} + (N - N_0) \cdot I_{set} & \text{if } N_0 < N/2 \\ (N - N_0) \cdot I_{reset} + N_0 \cdot I_{set} & \text{otherwise} \end{cases}$$

WAVAK differs from Flip-N-Write in the following aspects. First, Flip-N-Write inverts data in order to reduce the number of bits to write whereas WAVAK inverts data in order to reduce the number of power-hungry zero bits to be written. Second, while Flip-N-Write inverts data when the number of different bits between the old data and new data is over a half of the total data size, WAVAK inverts data when the number of zero bits is larger than a half of data size. Third, the storage overhead for inversion flag bits in WAVAK is much smaller than Flip-N-Write. Flip-N-Write has one flag bit for each 8-byte block, but we have a flag bit of for a cache line (64 bytes in this study).

V. Experiment Results

We evaluate our design by using the execution-driven processor simulator M5 [10] and the cycle-level memory simulator DRAMsim [11]. Table I shows the parameters of simulated processor and product grade PCM [2]. The simulated processor has four out-order cores with 32 MB L3 cache. The PCM read and write latency is set as 57*ns* and 430*ns*, respectively [2]. We set the electric currents to read a bit and write bit 1 to be 40 μA and 300 μA respectively according to Ref. [7]. A bank has eight subarrays according to Ref. [2].

Our experiments consider realistic PCM chip power constrains and set the PCM write unit to be 8 bytes. In conventional writing scheme, writing a cache line of 64 bytes to PCM takes $8 \times 430ns = 3440ns$ while reading a cache line needs less than $100ns$ due to the chip-level prefetch. Since writing a zero bit takes 600 μA current [2] and a write unit has 8 bytes, the power budget for a bank is $8 \times 8 \times 600$ μA = 38.4 *mA*. To tolerate slow PCM write, we add a large off-chip L3 DRAM cache. In addition, since the memory write back is not in the performance critical path, we choose to use the Read Instruction and Fetch First (RIFF) scheduling algorithm, which has been recommended by other researchers [12].

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 20*ns*, 2MB, 4-way, 64B cache line
L3 Cache	Latency 50*ns*, 32MB, 8-way, 64B cache line
Memory Controller	RIFF request scheduling algorithm
Width of data bus	64 bits
Memory Organization	2 ranks, 16 bank/bank, 8 subarrays/bank
Time to access PCM	read: 53*ns*, write: 430*ns*
Electric Current Requirement	600 μA for writing bit 0 (I_{reset}), 300 μA for writing bit 1 (I_{set}), 40 μA for reading a bit
Bank Power Budget	$8 \times 8 \times 600$ μA = 38.4 *mA*
PCM write unit size	8 bytes

TABLE I
Simulation Parameters

We construct 13 multi-programmed workloads with intensive memory accesses by combining multiple applications selected from the SPEC CPU 2006 benchmark suit. All applications in each workload run in parallel. Each application is fast-forwarded 5 billion instructions first and then 1.25 billion instructions are simulated for each application. 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 [12], [13], 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 our design with the baseline scheme that uses the same parameters as listed in Table I but does not have any PCM optimization. We also compare ours against two recently proposed PCM write optimization techniques including Write Cancellation (WC) [12] and Flip-N-Write (FNW) [14].

We also conduct experiments for the subarray-enabled bank without bank power limitation and the subarray-enabled bank with conventional power allocation scheme. Compared with these two schemes, we can demonstrate how effectively our proposed power allocation schemes work and how much our schemes can be improved by using the yardstick of an ideal PCM system without power limitation.

A. PASAK

Figure 4 presents the read latency reduction of PASAK over the baseline that does not have any write optimization, and

Workload	Description	RPKI	WPKI
MIX1	astar, bzip2, milc, leslie3d	1.75	1.37
MIX2	astar, cactusADM, libquantum, soplex	6.46	4.42
MIX3	cactusADM, bzip2, gobmk, mcf	0.98	0.66
MIX4	cactusADM, cactusADM, gobmk, gobmk	5.57	3.46
MIX5	gobmk, gobmk, cactusADM, hmmer	1.01	0.54
MIX6	gobmk, leslie3d, mcf, libquantum	1.23	1.21
MIX7	gobmk, zeusmp, mcf, lbm	4.08	2.19
MIX8	leslie3d, bzip2, mcf, lbm	1.94	1.2
MIX9	leslie3d, gobmk, lbm, astar	7.99	6.09
MIX10	leslie3d, leslie3d, milc, milc	4.05	2.54
MIX11	leslie3d, soplex, bzip2, astar	1.92	1.43
MIX12	soplex, libquantum, astar, GemsFDTD	1.14	0.83
MIX13	soplex, soplex, sjeng, sjeng	0.97	0.76

TABLE II
CHARACTERISTICS OF 13 WORKLOADS FOR A FOUR-CORE SYSTEM

compares it with the other optimization schemes. We have the following observations.

First, the subarray-enabled bank without power limitation (subarray-No-limit) achieves the best performance, with an average of 82.7% read latency reduction over the baseline. On the other hand, the subarray-enabled bank with power limitation (subarray-PW-limit) fails to achieve any latency reduction over the baseline. This is not surprising. Simply applying the subarray into PCM does not provide any performance gain since a power-consuming PCM write operation cannot overlap with read operations due to the overestimation of electric current required for a write in the conventional system.

Second, for most of 13 workloads studied, PASAK can successfully reduce the read latency more than Write Cancellation (WC) and Flip-N-Write(FNW). On average, the read latency of PASAK is 59% smaller than the baseline, while Write Cancellation and Flip-N-Write(FNW) are only 9.53% and 54.1% smaller, respectively.

Third, PASAK has a smaller latency for workloads that are more write-intensive. For example, the latency reduction for workload MIX4, MIX6, MIX7 and MIX9 is 91%, 78.6%, 74.8% and 73.4%, respectively. This is because in write-intensive workloads writes are more likely to block reads that are more critical to performance. In addition, intensive writes create more optimization opportunities that allow reads to be served concurrently with a write at the subarray level. We also observe that there is a 23.7% gap in latency between the ideal subarray-enabled bank without power limitation and PASAK, implying that PASAK can be further improved potentially.

Figure 5 shows the reduction of the total running time of PASAK and other schemes when compared with the baseline. The results confirm that PASAK's reduced read latency is directly translated into performance gain. On average, PASAK provides 31% performance improvement over the baseline and its average running time is 4.7% and 24.18% shorter than Flip-N-Write and Write Cancellation, respectively.

B. *WAVAK*

Figure 4 compares the read latency reduction over the baseline for six schemes, including Flip-N-Write (FNW), Write Cancellation (WC), ideal subarray-enabled bank (subArray-No-limit), subarray-enabled bank with power budget (subArray-PW-limit), PASAK and WAVAK. We observe that WAVAK achieves more read latency reduction than PASAK. On average, the latency reduction for WASAK is 68.7%, while PASAK is 59%. Compared with PASAK, the read latency of WASAK is 9.7% smaller. WAVAK reduces the number of power-consuming zero bits in a write data block via inversion and hence consumes less electric current than PASAK when writing a cache line. Therefore, WAVAK has a larger current balance to allow more read requests to be served in parallel than PASAK. In other words, WAVAK achieves a larger subarray-level parallelism than PASAK under the same bank power budget, resulting in a smaller latency.

The running time reduction over the baseline is shown in Figure 5. Since WAVAK utilizes a new coding scheme to make more electric current available to read requests within a bank, it can significantly reduce the queuing time of reading a cache line. Under 13 workloads studied in this paper, WAVAK achieves 34.8% running time reduction over the baseline on average, and outperforms PASAK, Flip-N-Write and Write Cancellation by 3.8%, 8.5% and 27.98%, respectively.

VI. RELATED WORK

Several research projects have aimed to hide the long write latency of PCM. Ref. [7] 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 [12] 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 [13] are proposed to improve write performance for the multiple-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.

The 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. [8] proposes power tokens to manage the PCM writes and specifies tight power allocation to increase the number of parallel write bits under a given power budget. In this paper we follow the bank's power budget from real PCM chip prototypes and enforce power budget at the bank level. We manage power for both read requests and write requests in order to exploit the subarray-level parallelism inside a subarray-enabled PCM bank.

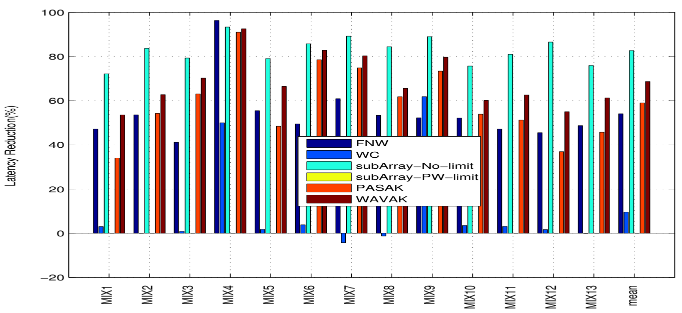

Fig. 4. Read Latency Reduction of WAVAK

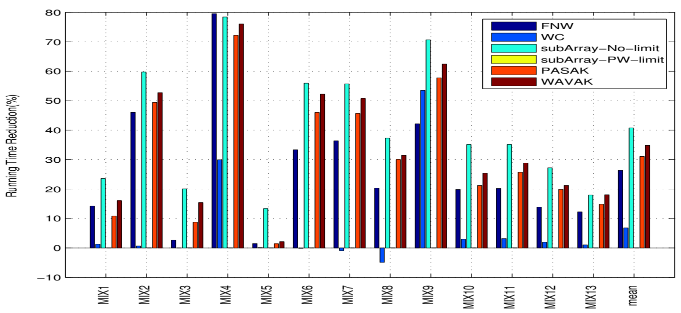

Fig. 5. Running Time Reduction of WAVAK

Many innovations of memory micro-architecture improve the memory performance. Among them, Ref. [1] is most closely related to our research. Although enabling the subarray-level parallelism works well for DRAM, the PCM's large writing power and its limited power delivery in bank prevents an executing slow PCM write request from overlapping with read requests in different subarrays in the same bank, resulting in under-utilization of subarrays.

VII. Conclusion

A subarray-enabled bank has been used in DRAM to mitigate the performance degradation due to bank conflicts. While PCM is emerging as a promising memory, this paper finds that the subarray-enabling technology cannot be directly applied to PCM due to large electric current draw for writes in PCM.

This paper presents and evaluates two power optimizations, called PASAK and WAVAK, that take advantage of the electric current difference of writing a zero bit and a one bit in PCM to judiciously enable subarray-level concurrent accesses. In PASAK, the memory controller accurately estimates the electric current required depending on the total number of zero bits and one bits in the data block to be written, and then dynamically schedule memory accesses with no subarray confliction according to the available power budget at each bank.

WAVAK further improves PASAK by reducing the power requirement of writing a data block. Based on the observation that writing bit 0 takes more electric current than write bit 1, all bits of a data block to be written are inverted if there are more zeros than ones in this block. An extra flag bit is added to record the inversion. In this way, WAVAK provides a larger power surplus than PASAK and accordingly allows more reads to be served concurrently with a slow write, resulting in better utilization of subarray-level parallelism.

Experiment results of 13 multi-programmed SPEC CPU 2006 workloads on a four-core out-order system with PCM memory show that PASAK and WAVAK successfully reduce the read latency of the standard PCM baseline by 59% and 68.7% on average, and reduce the running time by 31% and 34.8% on average, respectively. WAVAK outperforms Flip-N-Write and Write Cancellation by 14.6% and 59.17% in latency, and 8.5% and 27.98% in running time on average.

References

[1] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (salp) in dram," in *Proceedings of the 39nd annual international symposium on Computer Architecture*, ser. ISCA '12, 2012.

[2] K.-J. Lee, B.-H. Cho, W.-Y. Cho, and etc, "A 90 nm 1.8 v 512 mb diode-switch pram with 266 mb/s read throughput," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 150 –162, Jan. 2008.

[3] F. Bedeschi, C. Resta, and etc, "An 8mb demonstrator for high-density 1.8v phase-change memories," in *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, june 2004, pp. 442 – 445.

[4] H. rok Oh, B. hyung Cho, and etc, "Enhanced write performance of a 64-mb phase-change random access memory," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 1, pp. 122 – 126, jan. 2006.

[5] S. Kang, W. Cho, and etc, "A 0.1/spl mu/m 1.8v 256mb 66mhz synchronous burst pram," in *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, feb. 2006, pp. 487 –496.

[6] S. Kang, W. Y. Cho, B.-H. Cho, and etc, "A 0.1-um 1.8-v 256-mb phase-change random access memory (pram) with 66-mhz synchronous burst-read operation," *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 1, pp. 210 –218, Jan. 2007.

[7] 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, 2009, pp. 2–13.

[8] 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.

[9] M. Arjomand, A. Jadidi, A. Shafiee, and H. Sarbazi-Azad, "A morphable phase change memory architecture considering frequent zero values," in *29th IEEE International Conference on Computer Design*, ser. ICCD'11. Amherst, MA, USA: IEEE Computer Society, 2011.

[10] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, 2006.

[11] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob, "Dramsim: a memory system simulator," *SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 100–107, 2005.

[12] M. T. Jacob, C. R. Das, and P. Bose, Eds., *16th International Conference on High-Performance Computer Architecture (HPCA-16 2010), 9-14 January 2010, Bangalore, India*. IEEE Computer Society, 2010.

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

[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, 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.