MaxPB: Accelerating PCM Write by Maximizing the Power Budget Utilization

ZHENG LI, FANG WANG, DAN FENG, YU HUA, JINGNING LIU, and WEI TONG,
Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology,
Huazhong University of Science and Technology

Phase Change Memory (PCM) is one of the promising memory technologies but suffers from some critical problems such as poor write performance and high write energy consumption. Due to the high write energy consumption and limited power supply, the size of concurrent bit-write is restricted inside one PCM chip. Typically, the size of concurrent bit-write is much less than the cache line size and it is normal that many serially executed write units are consumed to write down the data block to PCM when using it as the main memory. Existing state-of-the-art PCM write schemes, such as FNW (Flip-N-Write) and two-stage-write, address the problem of poor performance by improving the write parallelism under the power constraints. The parallelism is obtained via reducing the data amount and leveraging power as well as time asymmetries, respectively. However, due to the extremely pessimistic assumptions of current utilization (FNW) and optimistic assumptions of asymmetries (two-stage-write), these schemes fail to maximize the power supply utilization and hence improve the write parallelism.

In this article, we propose a novel PCM write scheme, called MaxPB (Maximize the Power Budget utilization) to maximize the power budget utilization with minimum changes about the circuits design. MaxPB is a "think before acting" method. The main idea of MaxPB is to monitor the actual power needs of all data units first and then effectively package them into the least number of write units under the power constraints. Experimental results show the efficiency and performance improvements on MaxPB. For example, four-core PARSEC and SPEC experimental results show that MaxPB gets 32.0% and 20.3% more read latency reduction, 26.5% and 16.1% more write latency reduction, 24.3% and 15.6% more running time decrease, 1.32× and 0.92× more speedup, as well as 30.6% and 18.4% more energy consumption reduction on average compared with the state-of-the-art FNW and two-stage-write write schemes, respectively.

CCS Concepts: • **Information systems → Phase change memory**; • **Computer systems organization → Architectures**; • **Hardware → Emerging technologies;**

Additional Key Words and Phrases: PCM, write scheme, power budget, write unit

---

New paper, not an extension of a conference paper.
This work was supported by the National High Technology Research and Development Program (863 Program) No. 2015AA015301, No. 2013AA013203, and No. 2015AA016701; National Key Research and Development Program of China under Grant 2016YFB1000202; NSFC No. 61303046, No. 61472153, and No. 61173043; State Key Laboratory of Computer Architecture, No. CARCH201505; Wuhan Applied Basic Research Project (No. 2015010101010004). This work was also supported by Key Laboratory of Information Storage System, Ministry of Education, China.
Authors' addresses: Z. Li, F. Wang (corresponding author), D. Feng, Y. Hua, J. Liu, and W. Tong (corresponding author) are with the Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology, Huazhong University of Science and Technology, Division of Data Storage System, Wuhan 430074, China; emails: {lizheng, wangfang, dfeng, csyhua, jnliu, Tongwei}@hust.edu.cn.

DOI: http://dx.doi.org/10.1145/3012007

ACM Reference Format:
Zheng Li, Fang Wang, Dan Feng, Yu Hua, Jingning Liu, and Wei Tong. 2016. MaxPB: Accelerating PCM write by maximizing the power budget utilization. ACM Trans. Archit. Code Optim. 13, 4, Article 46 (December 2016), 26 pages.
DOI: http://dx.doi.org/10.1145/3012007

1. INTRODUCTION

Phase Change Memory (PCM) has received extensive attention from institutes and companies for its better scalability and lower leakage-power consumption compared with traditional DRAM technology [Lee et al. 2010a, 2010c; Wong et al. 2010; Roadmap 2013; Raoux et al. 2008]. However, PCM has many shortcomings that need solving, such as high write latency [Lee et al. 2009; Zhou et al. 2009; Qureshi et al. 2009b; Jiang et al. 2012b; Li et al. 2016a, 2016b] and limited write endurance [Qureshi et al. 2009a; Seong et al. 2010; Huang et al. 2016; Jiang et al. 2013]. Power restriction is the key bottleneck of the write performance. Due to the high write energy consumption and huge power noise in narrow space, the size of write unit is strictly restricted. The size of write unit refers to the size of concurrent bit-write to the PCM array. The typical sizes of write units are strictly limited to 4, 8, and 16 bits, and even only 2 bits under the worst current supply circumstance [Kang et al. 2007; Lee et al. 2008; Cho and Lee 2009; Yue and Zhu 2012, 2013a, 2013b; Li et al. 2016a, 2016b]. If PCM serves as the main memory, the write limitations lead to long write latency. As a result, long write latency causes significant delays of more critical read requests [Zhao et al. 2014] and it also takes time for memory bus to do the read and write switching [Lee et al. 2010b]. If there are a number of read or write requests within a memory bank, the interference caused by write requests could be worse. In short, long write latency restricts service parallelism and causes huge performance degradation.

In order to improve the memory system's performance, Last-Level Cache (LLC) is widely used to cache time-hungry write requests [Du et al. 2013a]. However, the size of LLC is strictly restricted due to energy consumption and cost considerations. Write-backs to PCM-based main memory still delay the critical read requests and cause system performance degradation [Zhou et al. 2012; Wang et al. 2013]. Typically, write unit is much smaller than cache line. Due to the mismatching of the write unit's size and cache line's size, the write operations can only be finished with many successive operated write units, which results in high cache line write service time. For example, a 64-byte cache line block should be written back to the main memory in a typical memory system but only (16×4) bits can be written one time with four PCM chips making up the 64-bit data bus. Due to the restrictions caused by the power supply, it consumes $(64 \times 8)/(16 \times 4) = 8$ write units for writing the data of a cache line [Lee et al. 2008; Cho and Lee 2009; Yue and Zhu 2012, 2013a, 2013b; Li et al. 2016a, 2016b]. These almost serially executed write units cause huge long cache line service time and high write latency as well as overall system performance degradation. Moreover, the size of the LLC line is growing to 128 bytes or bigger in some of the latest computer systems [Warnock et al. 2011; Kalla et al. 2010], which leads to the worse write performance.

Existing state-of-the-art PCM write schemes, such as FNW (Flip-N-Write) or two-stage-write, aim to address the poor performance problem by improving the write parallelism under the power constraints. The parallelism is obtained via reducing the data amount with ingenious data encoding and leveraging both power and time asymmetries, respectively. However, we observe the following:

—FNW is quite simple and effective but faces low power budget utilization problems. In other words, only a small number of bits is changed and consumes power by "removing the redundant bit-write" method [Zhou et al. 2009], and the power is often

excessively supplied but underutilized. According to our experimental results of 16 PARSEC and SPEC workloads, the power budget utilization is only 27.4%/14.9% without/with power asymmetry on average under FNW scheme and the system environment is shown in Table I.

—two-stage-write is highly efficient but suffers from performance degradation when the asymmetries are not significant. According to our theoretical analysis shown in Figure 3, if SET is four times (or more) slower than RESET, two-stage-write gets significant write service-time reduction over FNW. Otherwise, the write performance improvement is not that significant. Moreover, two-stage-write needs extra control circuits and the modified write driver logics to support the separation of writing ones and zeros.

Based on these key observations, we propose a novel PCM write scheme named MaxPB (Maximize the Power Budget utilization) based on the SLC PCM for its better write performance and intuitive power budget model [Lee et al. 2008; Yue and Zhu 2013a]. Our design goals are to **maximize** the power budget utilization with **minimum** changes in the circuits design. MaxPB is a "think before acting" method. The main idea of MaxPB is to rearrange all data units according to their actual power needs and package all data units into the least number of write units under the power constraints. MaxPB can significantly improve the write parallelism and reduce the critical number of write units.

The main contributions of this article include the following:

—An effective PCM write algorithm called **MaxPB**. We observe that existing write schemes are not aware of low power budget utilization and experimental results of 10 multithreaded and six multiprogrammed workloads show that the utilization is only 27.4% without power asymmetry. With MaxPB, we can maximize power budget utilization by using the least number of write units.

—A variation of MaxPB named **MaxPB-asy**. We observe that the power budget utilization decreases to 14.9% considering the power asymmetry of SET and RESET. By leveraging power asymmetry, MaxPB-asy can obtain more performance improvement and energy consumption reduction compared with the original MaxPB scheme.

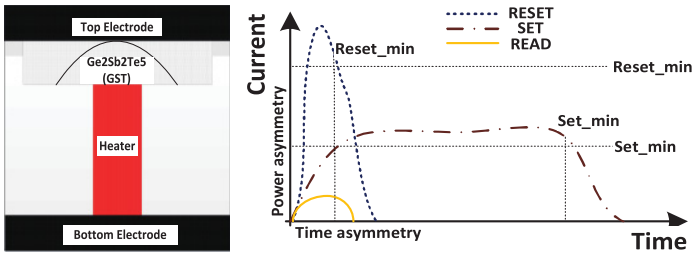
—Efficient **hardware circuits** design to support MaxPB and MaxPB-asy. The circuits are slightly altered with extremely low overhead compared with FNW and two-stage-write designs.

The remainder of this article is structured as follows. Section 2 describes the background, the details of existing write schemes, and motivations of our designs. Section 3 describes the implementation of circuit designs. Section 4 presents and analyzes the experimental results. Section 5 introduces the related work. Finally, Section 6 concludes our article.

2. BACKGROUND AND MOTIVATION

2.1. PCM Properties and Limitations

PCM takes advantage of the properties in resistance of the storage material (such as Ge2Sb2Te5, briefly called GST). The material shows great diversity and resistance gap when the material shows different states, that is, crystalline state and amorphous state. In general, amorphous-state material shows several orders of magnitude higher resistance value than crystalline-state material. If given the same voltage level, the current value adopted by the sense amplifier varies four or more orders of magnitude when the material shows opposite states. The striking differences in resistance and current status can be used for presenting binary information, that is, digital "0" and "1." A typical mushroom structure of a PCM cell is illustrated in Figure 1(a). A PCM



(a) Cell structure (b) Read and write processes

Fig. 1. Typical mushroom PCM cell architecture and operation mechanisms.

cell typically adopts mushroom architecture including electrodes, heater, and phase change material. Simplified read and write mechanisms and Current-Time sparkline of PCM are illustrated in Figure 1(b). PCM has obvious asymmetries [Xia et al. 2015]:

—Asymmetries between read and write. The power needs and the service time of read and write vary significantly. In general, read operation consumes less power with shorter service time [Qureshi et al. 2012; Lee et al. 2008].
—Asymmetries between reset and set. The power needs and the write time of reset and set vary significantly. In general, set operation needs higher current level but its service time is much shorter than set operation [Qureshi et al. 2012; Sun et al. 2012].

PCM write operation consumes a great deal of current [Lee et al. 2008; Cho and Lee 2009]. However, due to the power noise and limited charge pump area, the maximum current that can be provided for one chip is highly restricted. It is also difficult and undesirable to address this problem by enlarging the charge pump, which is the most area-hungry component inside a PCM chip (usually more than 20% of the chip area [Oh et al. 2006; Palumbo and Pappalardo 2010]). As a result, the parallelism inside one chip is constrained and only a small number of bits can be written concurrently under the power budget. The size of write unit, that is, the size of concurrent bit-write to the PCM array, is restricted and typical sizes of write unit are 4, 8, and 16 bits [Lee et al. 2008; Cho and Lee 2009; Yue and Zhu 2013a, 2013b; Li et al. 2016a, 2016b]. Power budget is the key obstacle of the poor PCM write performance. When PCM is adopted as the main memory, we get huge overall system performance degradation due to the size gap between write unit and cache line block. PCM's write operations can only be finished with many serially executed write units, which results in high write service time. The almost serially executed write units cause huge long write latency and overall system performance degradation.

2.2. State-of-the-Art Write Schemes

The conventional write scheme is under pessimistic assumptions about the power demand of each unit, regardless of the power and time asymmetries in PCM write. Typically, as shown in Equation (1), it strictly writes $S_{writeunit}$ bits per write unit until finishing the write service of S_{total} bits. As illustrated in Figure 2, the service is finished at T5 under the parameters presented in Table I.

$$T_{Conventional} = \frac{S_{total}}{S_{writeunit}} T_{set}. \tag{1}$$

In order to address the poor write performance problem, FNW [Cho and Lee 2009] uses a thin read-before-write scheme, encodes the data with an extra bit to reduce the

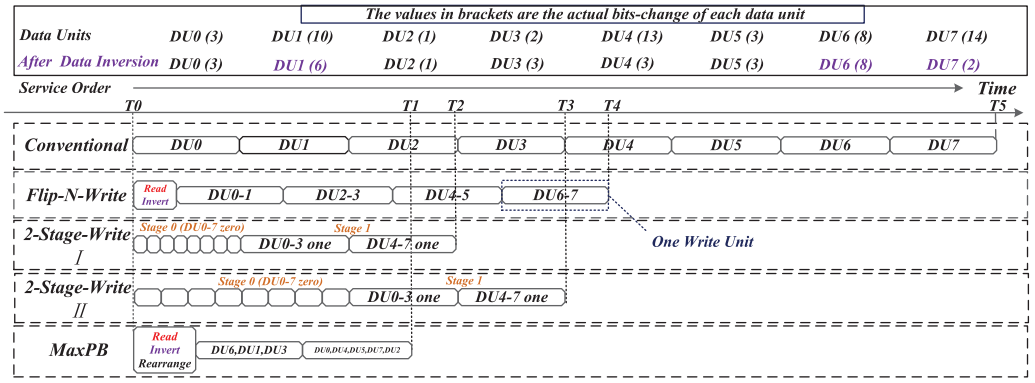


Fig. 2. An example for state-of-the-art PCM write schemes. Assuming 16 bits can be served at the same time and the values in brackets are the actual bits-change of each data unit.

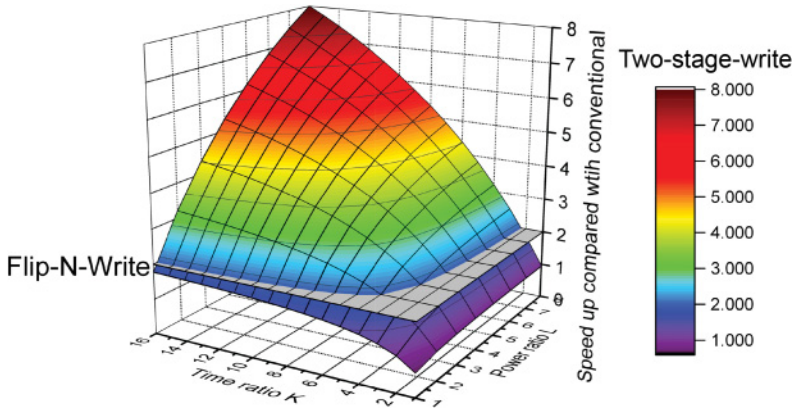


Fig. 3. Acceleration of FNW and two-stage-write over conventional scheme.

amount of written data, and improves the write parallelism by writing the different bits only. In general, the concurrent serviceability is doubled and the write service time is halved compared with the conventional write scheme, as shown in Equation (2).

$$T_{FNW} = T_{read} + \frac{S_{total}}{2 \times S_{writeunit}} T_{set}. \tag{2}$$

In our sample, the cache line service can be finished at T4 as illustrated in Figure 2.

Two-stage-write [Yue and Zhu 2013a] accelerates PCM write parallelism by leveraging the asymmetries of writing "0" and "1" (time and power demand) as shown in Figure 1(b). By adding carefully designed stage control circuits, the write is finished in two steps, that is, "Stage 0" and "Stage 1." Assuming the time ratio of writing "0" and "1" is $1/K$, the current ratio is L. The average service time of a cache line is concluded in Equation (3).

$$T_{two\text{-}stage\text{-}write} = \frac{S_{total}}{K \times S_{writeunit}} T_{set} + \frac{S_{total}}{2 \times S_{writeunit} \times L} T_{set}. \tag{3}$$

The write service time of two-stage-write is highly relative to the current ratio and time ratio, as illustrated in Figure 3. When the current ratio L is 2, the time ratio K is 8, two-stage-write finishes its cache line write service at T2 (two-stage-write I in

Figure 2). However, if the time gap between SET and RESET is not huge, the theoretical performance of two-stage-write is close to FNW. When the time ratio is 4, the write service is finished at T3 (two-stage-write II in Figure 2).

2.3. Observations

We give a new metric named **power budget utilization** to measure the efficiency of current supply utilization. The definition of power budget utilization is given in Equation (4) and the term $ActualPowerUse$ without/with power asymmetry is calculated by Equations (5) and (6), respectively. In general, power budget utilization means the rate of actual power use per write unit and the maximum power supply provided by the charge pump. Under the same dataset, higher power budget utilization means that more data bits are written in one write unit and higher chip-level write parallelism, which reduces the write service time of PCM-based main memory.

$$PowerBudgetUtilization = \frac{PowerUse}{PowerBudgetPerWriteUnit}, \tag{4}$$

$$PowerUse = Totalbits \times Power/bit, \tag{5}$$

$$PowerUseAsy = SETbits \times SETPower/bit + RESETbits \times RESETPower/bit. \tag{6}$$

On one hand, FNW is simple and effective but faces low power budget utilization problems. In general, only a small number of bits is changed compared with original data (old data) and consumes power, and the power is often excessively supplied but underutilized. Low power budget utilization limits parallelism upgrade and write performance improvement. According to our experimental results of multithreaded PARSEC and multiprogrammed SPEC 2006 workloads illustrated in Figure 4, the power budget utilization is only 27.4%/14.9% without/with power asymmetry on average with FNW scheme and the system environment and benchmark details are shown in Table I and Table II, respectively. On the other hand, two-stage-write is highly efficient but suffers from performance degradation when the asymmetries are not significant. According to our theoretical analysis shown in Figure 3, if SET is four times (or more) slower than RESET, two-stage-write gets huge enhancement over FNW. Otherwise, the performance improvement is not that significant. Moreover, two-stage-write needs extra control circuits and modified write driver logics to support the separation of writing ones and zeros.

To address these problems, we propose a novel PCM write scheme named MaxPB based on the insight that existing PCM write schemes are unaware of their low power budget utilization. Our design goals are to maximize the power budget utilization with minimum changes to the circuits design. MaxPB rearranges the executed sequence of all write units and maximizes the power budget utilization with the least number of write units. The main idea of MaxPB is "think before acting," that is, we rearrange all data units according to their actual power needs and package more data in one write unit to minimize the number of write units rather than writing down the data directly.

As shown in Figure 2, the values in brackets present the actual number of different bits of each unit compared with the old data stored in the chip. If the number of bits changed is more than half of the maximum size under the limitations of the power budget, the data will be flipped like FNW. After that, the power requirement of each data unit is no more than half of the power budget size. MaxPB then rearranges all data units in descending order according to their actual power requirement. After all units are rearranged, MaxPB writes data units in parallel according to the results of the

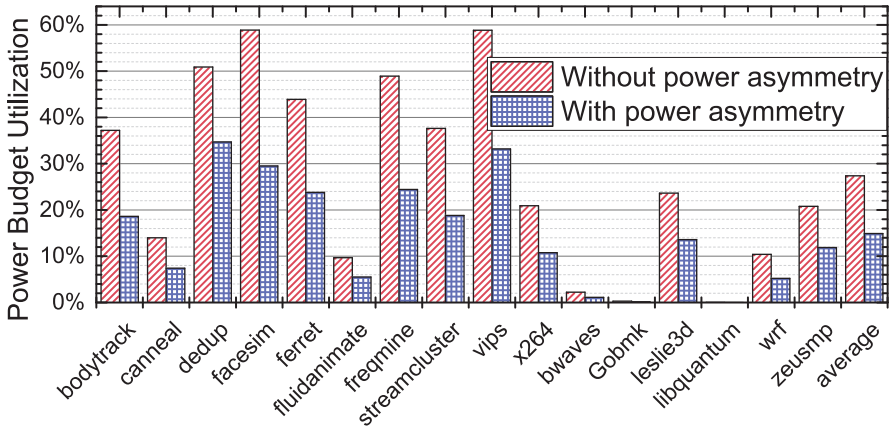


Fig. 4. Power budget utilization of FNW scheme.

MaxPB algorithm illustrated in Algorithm 1. After the "think before acting" process, DU6, DU1, and DU3 are chosen to execute concurrently for the number of bits needed to be written is less than the maximum number, that is, the total power requirement is smaller than the power limitation ($8 + 6 + 2 <= 16$). Similarly, all other data units can be written concurrently under the power constraints ($3 + 3 + 3 + 2 + 1 <= 16$). The cache line write service can be finished in T1, which is shorter than FNW (T4) and two-stage-write (T2), respectively. However, the overall overhead is higher than FNW and two-stage-write for the flow path of all units' rearrangement. According to our experimental results of designing MaxPB on Xilinx FPFA, the MaxPB write algorithm causes 25 cycles overhead on average to finish the packaging of all units. However, compared with the thousands of PCM write cycles, the overhead is extremely low and acceptable (within 1%). Moreover, MaxPB delivers no extra overhead on the critical read path, which has no bad influence on critical read performance.

3. IMPLEMENTATION AND OVERHEAD

3.1. Hardware Circuits Support for MaxPB

To meet the proposed design goals of MaxPB, we implement altered hardware circuits based on an industrial PCM prototype published by Samsung [Lee et al. 2008] and FNW write scheme [Cho and Lee 2009]. The data path of our design is shown in Figure 5. The original prototype supports an eight-word prefetch method with a big buffer to reduce the data transmission overhead.

To support the proposed MaxPB write scheme, we add a middle circuit design, that is, MaxPB write logic, between the write buffer and S/A write driver compared with the original design. It is worth noting that we do not increase the read operation overhead and the length of the read path is the same as the original and FNW designs. Moreover, the overall length of the write path is similar to FNW and two-stage-write designs. We extend the write buffer size to 160 bits and the write buffer is composed of eight data units. Actually, the data are flipped similar to the data conversion of FNW. Extra 8 flip bits are added to index the data have been flipped or not similar to FNW. To monitor the actual power consumption of eight data units, 24 Wup bits are used to store the power requirements of data units, that is, the number in the brackets as shown in Figure 2. We assume the size of the write unit is 16 and 3 Wup bits (2^3 equals to 8) are used for per data unit as the maximum number of bits changed is 8 after the inversion process. Our design does not need extra PCM array compared with FNW design as we

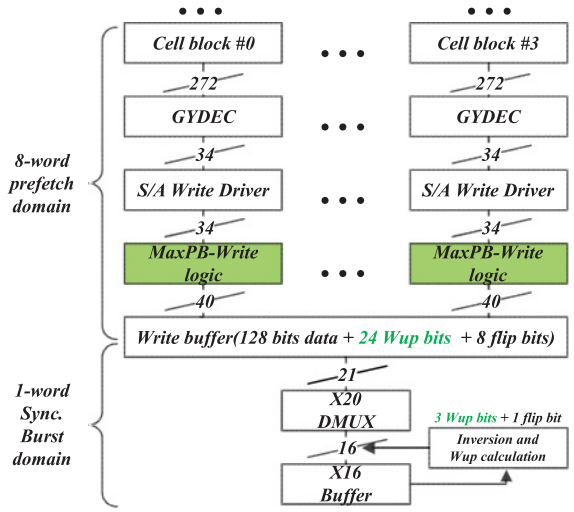


Fig. 5. Data path of MaxPB.

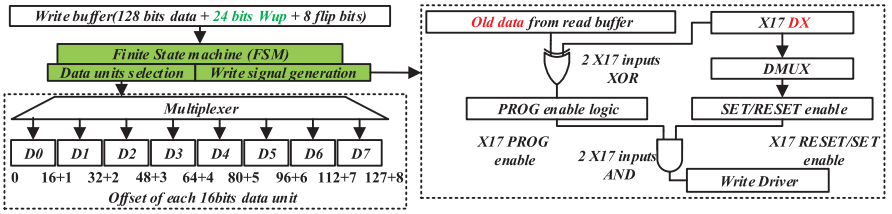


Fig. 6. MaxPB write logic.

just need to extend the write buffer size, which can be easily implemented in the PCM chip.

MaxPB write logic gets the data to be written together with flip bits and Wup bits, and then decides which data units should be packaged into one write unit. The write control logic is illustrated in Figure 6. The shared Finite State Machine (FSM) is the key component of our MaxPB write logic design. FSM consists of data units selection logic and write signal generation logic. In detail, FSM decides which data units should be packaged into one write unit to get the maximum power budget utilization with the minimum number of write units. In order to boost the accessing efficiency, all data units, that is, DX in Figure 6, are indexed by fixed offsets. Offset 0 corresponds to data unit 0 (D0) and D1–D7 are indexed by 17, 34, 51, 68, 85, 102, and 119 considering the flip bits, respectively. The units packaging and choosing processes deliver extra overhead when finishing a cache line write service.

The data units packaging algorithm is shown in Algorithm 1. The key idea behind the MaxPB algorithm is to decide which data units should be executed in parallel under power constraints to earn better power budget utilization and lower number of write units. MaxPB is a greedy algorithm and the key working processes are as follows:

(1) Calculating the actual power requirements based on the number of bits that have to be written, that is, the number of different bits between new and old data.
(2) Rearranging all data units in descending order according to their actual power requirement. Since there are only a few data units (eight data units in our study), the ordering and selection process is fast and efficient.

ALGORITHM 1: MaxPB Algorithm

Require:
The quantity of existing data units, Qdu;
The number of bits changed of each data unit, $Diffbits[Qdu]$;
The power consumption of each data unit, $Pcons[Qdu]$;
The actual power use of each write unit, $Pwu_p[Qdu]$;
Power budget limitation, $PBlimitmax$;
The power demand for writing one bit, PD_{bit};
Ensure:
The write order of all data units;
1: *Initial* $Pwu_p[Qdu] = \{0\}, Writeunitnum = 0, findtag = false$;
2: **for** each $i \in [1, Qdu]$ **do**
3: $Pcons[i] = Diffbits[Qdu] \times PD_{bit}$
4: **end for**
5: **for** $i = 1; i <= Qdu; i++$ **do**
6: Find unprocessed $Pcons[i]$ with highest value;
7: $findtag = false$;
8: **if** $Pcons[i] > \frac{PBlimitmax}{2}$ **then**
9: $findtag = true$;
10: $Writeunitnum++$;
11: $Pwu_p[Writeunitnum] += Pcons[i]$;
12: Marking $dataunit[i]$ is written in $writeunit[Writeunitnum]$;
break;
13: **end if**
14: **for** $j = 1; j <= Writeunitnum; j++$ **do**
15: **if** $Pcons[i] + Pwu_p[j] < PBlimitmax$ **then**
16: $findtag = true$;
17: $Pwu_p[j] += Pcons[i]$;
18: Marking $dataunit[i]$ is written in $writeunit\ j$;
break;
19: **end if**
20: **if** $findtag == false$ **then**
21: $Writeunitnum++$;
22: $Pwu_p[Writeunitnum] += Pcons[i]$;
23: Marking $dataunit[i]$ is written in $writeunit[Writeunitnum]$;
24: **end if**
25: **end for**
26: **end for**

(3) From the data unit with the highest power requirement to the data unit with the least power need, trying to put a current data unit into an existing write unit and recording the power budget residue of each existing write unit.
(4) If the remaining power supply of one write unit is enough, the data unit is marked that it should be written into this write unit. If all write units cannot satisfy the processing data unit, another write unit is enabled and the data unit is marked with the new write unit.

After the preceding process, we get the total number of write units and keep in mind which data units should be packaged in one write unit, respectively. According to the process results, FSM selectively sends the write control signals to each data unit's write driver using a simple multiplexer as illustrated in Figure 6.

The write driver logic is shown in Figure 6. Like FNW and two-stage-write design, an extra control signal, called PROG-enable, is used for individual bit control. Once FSM sends a writing signal to a data unit DX, the write driver logic compares the data

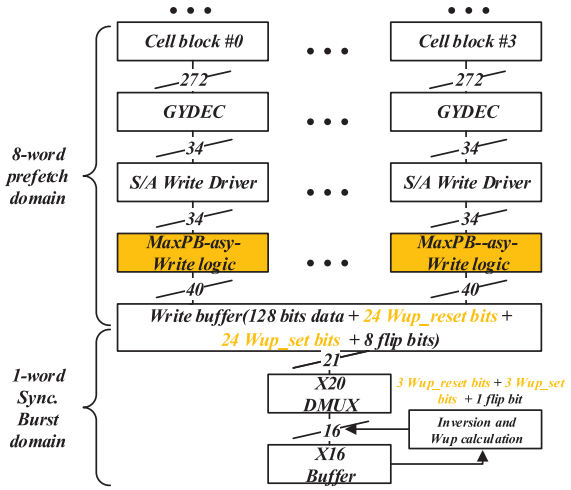


Fig. 7. Data path of MaxPB-asy.

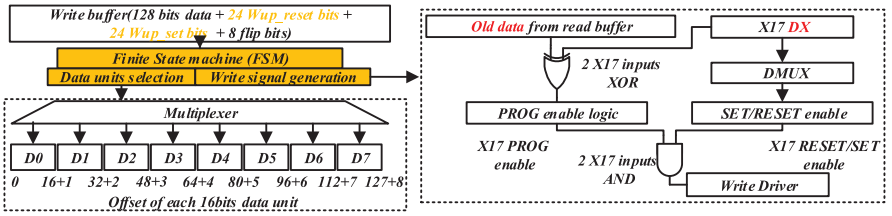


Fig. 8. MaxPB-asy write logic.

DX with the data already stored in the PCM array, that is, 16 bits old DX, by leveraging a simple XOR gate logic. A DMUX gate logic is used for deciding the stage of writing "0" or "1," that is, reset and set enable, respectively.

MaxPB circuit designs do not deliver any extra overhead on the critical read path, which is the key bottleneck of the system performance. Moreover, MaxPB can significantly reduce the number of the write units and improve the write performance with extremely low overhead.

3.2. MaxPB with PCM Asymmetries

Based on the PCM power asymmetry, we propose a new method called MaxPB-asy to further improve the write parallelism. The primary difference between MaxPB-asy and MaxPB is the statistical approach of "actual power use." There exists power asymmetry in PCM write that RESET operation consumes much more current than SET operation. The data path and write control logic of MaxPB-asy are illustrated in Figures 7 and 8. Compared with 24 Wup bits overhead in MaxPB, the buffer of MaxPB-asy needs 24 Wup_reset bits and 24 Wup_set bits to store the number of RESET and SET operations of all data units. Wup calculation module is modified to count the number of RESET and SET operations of all data units after data inversion. Only the buffer is extended compared with the original FNW design and it is not necessary to write down all Wup bits to PCM array. The write control logic of MaxPB-asy is virtually identical to MaxPB design and the only difference is the algorithm inside the finite state machine.

The algorithm of MaxPB-asy is shown in Algorithm 2. The algorithm input is extended for some necessary information such as the number of SET and RESET bits

ALGORITHM 2: MaxPB-asy Algorithm

Require:
The quantity of existing data units, Qdu;
The number of SET bits changed of each data unit, $Diffbits_{SET}[Qdu]$;
The number of RESET bits changed of each data unit, $Diffbits_{RESET}[Qdu]$;
The power consumption of each data unit, $Pcons[Qdu]$;
The actual power use of each write unit, $Pwu_p[Qdu]$;
Power budget limitation, $PBlimitmax$;
The power demand for SET one bit, PD_{SET};
The power demand for RESET one bit, PD_{RESET};
Ensure:
The write order of all data units;
1: Initial $Pwu_p[Qdu] = \{0\}; Writeunitnum = 0;\ findtag = false$;
2: **for** each $i \in [1, Qdu]$ **do**
3: $Pcons[i] = Diffbits_{RESET}[Qdu] \times PD_{RESET} + Diffbits_{SET}[Qdu] \times PD_{SET}$
4: **end for**
5: **The remaining content is the same as Algorithm 1.**

changed of each data unit and power demand for SET and RESET one bit. MaxPB-asy uses fine-grained power demand calculation and the power consumption of each data unit is counted by leveraging the power asymmetry of SET and RESET. After the fine-grained power consumption calculation, the dealing processes are the same with MaxPB.

3.3. Overhead Discussions

As we mentioned, MaxPB and MaxPB-asy modified the conventional circuits and introduce extra data inversion and rearrangement processes, which deliver extra time, space, area, and power overhead. According to our experimental results on Xilinx FPGA (xc7z045ffv900-1) [Xilinx 2015b], the overhead is about 25 cycles on average. However, compared with the thousands of PCM write cycles, the overhead is extremely low and acceptable (within 1%). Moreover, MaxPB does not deliver any extra overhead on the critical read path, which is the key bottleneck of the system performance [Qureshi et al. 2010a]. The implementation of FSM only modifies the write datapath without the changes of read datapath. In other words, our designs accelerate the write operations while remaining the short latency of read operations. The space overheads of MaxPB and MaxPB-asy are almost the same. For each 16-bit data unit, one flip bit is needed, and the storage is 6.25%. The other critical information, such as Wup bits in our design, only consumes additional buffer size. In other words, MaxPB and MaxPB-asy extend the size of write buffer only and it is easy to be implemented. Moreover, the added write control logic is not in the list of cost-sensitive designs. The write driver, as well as FSM, are less complicated than the complex charge pump design, which consumes up to 20% of the chip area [Oh et al. 2006; Palumbo and Pappalardo 2010]. The area and power overheads are low and acceptable according to our experimental results. The FSM of MaxPB and MaxPB-asy only takes 237 LUTs (Look-Up Tables) in Xilinx FPGA (xc7z045ffv900-1) whose available LUTs are 218,600 (less than 0.1%). Moreover, MaxPB adds an extra data comparison logic to generate "PROG enable" signals, which can be easily implemented with a XOR gate and a counter. Compared with the original write driver, "SET/RESET" signal for one bit is AND-gated with the "PROG enable" signal. According to the results of VIVADO 2015.2 tools [Xilinx 2015a], the total power overhead of MaxPB/MaxPB-asy circuits is near to 2mW. The voltage of write is 5V in the original PCM prototype [Lee et al. 2008], and the current level is about 25mA at the same time, where the write current level for RESET cells is from 0.6mA to 1mA,

Table I. Parameters of Simulation

Parameter	Value
CPU	Four-Core CMP, 2GHz, ALPHA-architecture processor
Memory Bus	64 bits, 800MT/s LPDDR
Memory Controller	FRFCFS-WQF scheduling, 32-entry R/W queues
Memory Organization	4GB, 64 bits data width, single rank with eight banks
PCM Organization	Four PCM chips, 8B write unit size
Cache Organization	64-byte line size, write back, LRU replacement
L1 Cache	32KB I-cache, 32KB D-cache, two cycles access latency
L2 Cache	Eight-way, 2MB, 20 cycles access latency
L3 Cache	16-way, 32MB, 50 cycles access latency
Memory Timing	Read 53ns, RESET 50ns, and SET 430ns

typically. The efficiency of current switch during the write operation is only about 20%, so we assume that one write should consume $5 \times 25 = 125$mW power. Compared with the original design, the power overhead delivered by our MaxPB and MaxPB-asy designs is relatively low.

4. EVALUATION

In this section, we evaluate the effectiveness and efficiency of MaxPB and MaxPB-asy write schemes and analyze the experimental results. MaxPB and MaxPB-asy can significantly improve the power budget utilization and reduce the number of write units, that is, significantly improve the write performance of PCM. The experimental results include power budget utilization, the number of write units, read latency, write latency, CPI speedup, running time, and energy consumption.

4.1. Experimental Environment

We first introduce the experimental environment, which is shown in Table I. We use the event-driven GEM5 simulator [Binkert et al. 2011] combined with NVmain [Poremba and Xie 2012] module and all parameters are from the prototype published by Samsung [Lee et al. 2008] and previous works [Cho and Lee 2009; Yue and Zhu 2013a; Li et al. 2015; Choi et al. 2012; Meza et al. 2012a; Nair et al. 2015]. Specifically, we adopt the read and write latency parameters from an enterprise-class prototype [Lee et al. 2008] to provide the same test environment with compared state-of-the-art PCM write schemes [Cho and Lee 2009; Yue and Zhu 2013a; Li et al. 2015]. We use 10 workloads from the multithreaded PARSEC benchmark suite [Bienia 2011] and six workloads from the multiprogrammed SPEC 2006 benchmark suite [Henning 2006] with different memory Read Per Kilo Instructions (RPKI) and memory Write Per Kilo Instructions (WPKI) rates, as shown in Table II and Table III. All of these benchmarks are collected from different areas and we use the DCW write scheme as the baseline scheme [Yang et al. 2007].

4.2. Power Budget Utilization

The experimental results of power budget utilization are shown in Figure 9. MaxPB and MaxPB-asy show much higher utilization than FNW. MaxPB earns 46.9% power budget utilization and MaxPB-asy shows more than 40.2% utilization on average. In comparison, FNW gets only 27.4%/14.9% power budget utilization without/with power asymmetry on average. One workload shows more than 80% utilization (facesim) when using the MaxPB method and five workloads indicate more than 70% utilization (dedup, ferret, freqmine, streamcluster, and vips). One workload exhibits more than 70% power budget utilization (facesim) when using MaxPB-asy write scheme. The reason is that

Table II. Multithreaded Benchmarks

Benchmark	Introduction	RPKI	WPKI
bodytrack	Animation physics	0.13	0.17
canneal	Chip design	1.89	1.31
dedup	Data deduplication	0.34	0.33
facesim	Computer vision	0.86	0.71
ferret	Computer server	0.22	0.15
fluidanimate	Animation physics	0.13	0.11
freqmine	Data mining	0.32	0.33
streamcluster	Video encoding	0.03	0.02
vips	Image processing	1.33	1.21
x264	Video encoding	0.18	0.20

Table III. Multiprogrammed Benchmarks

Benchmark	Introduction	RPKI	WPKI
bwaves	Four copies of bwaves	11.7	11.0
gobmk	Four copies of gobmk	1.63	1.49
leslie3d	Four copies of leslie3d	5.60	8.72
libquantum	Four copies of libquantum	6.80	6.94
wrf	Four copies of wrf	2.54	2.32
zeusmp	Four copies of zeusmp	10.62	6.86

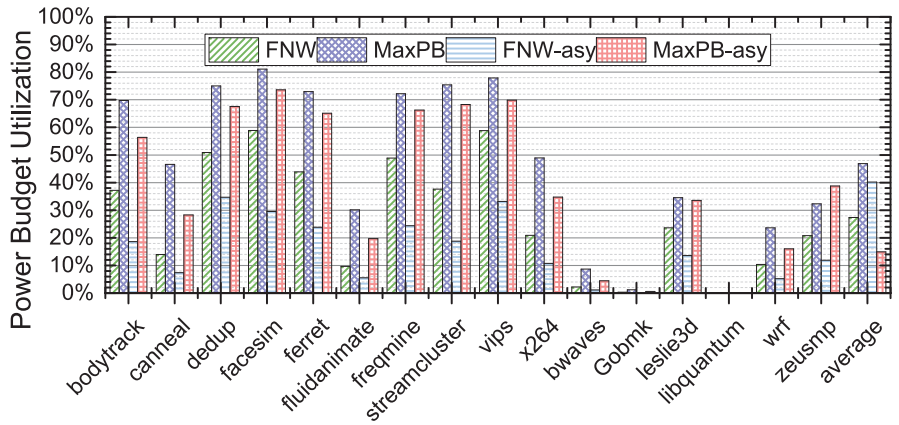


Fig. 9. Power budget utilization of compared write schemes.

our designs package more data units into one write unit and the power supply is fully used in these benchmarks. Under the same dataset, higher power budget utilization means more data bits are written in parallel with higher current use efficiency and write parallelism, which reduces the write service time of PCM-based main memory.

4.3. The Number of Write Units

The number of sequentially executed write units is one of the key metrics of PCM write latency and write performance, and less write units mean less write time consumption. We measure the average number of write units among 16 different workloads under the MaxPB and MaxPB-asy write schemes, as shown in Figure 10. In general, the number of write units, that is, the number of write-executed times to finish a cache line service, varies from 1.2 to 3.1 and is 2.0 on average under MaxPB write scheme. When

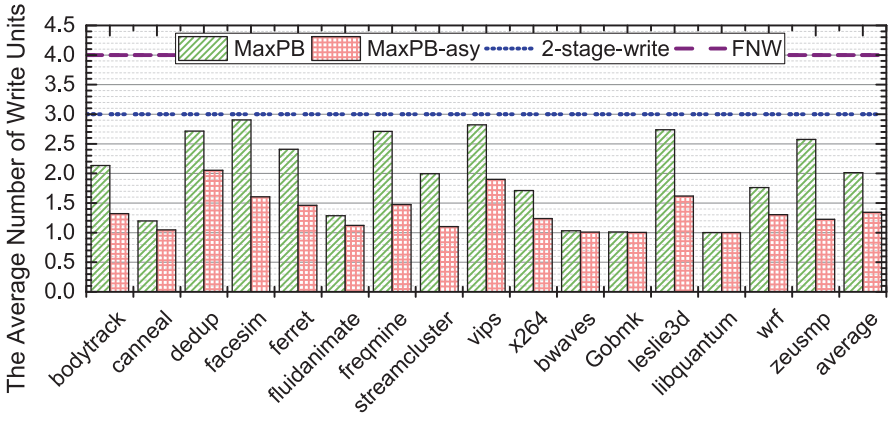


Fig. 10. The average number of write units.

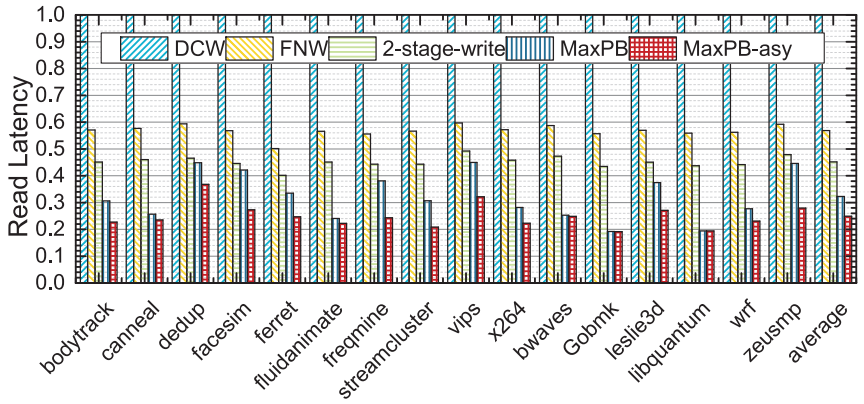


Fig. 11. Read latency.

adopting MaxPB-asy write scheme, the number of write units can be further reduced. The average number of write units is 1.4 when using MaxPB-asy write scheme. In comparison, the average number of write units when implementing FNW is almost 4 while two-stage-write's average number of write units is close to 3. MaxPB and MaxPB-asy can maximize the power budget utilization and minimize the number of write units, hence improving the overall system performance.

4.4. Read Latency

Read is on the critical path of the whole system performance and short read latency could deliver good system corresponding time. MaxPB and MaxPB-asy can minimize the number of write units and hence reduce the read latency under conventional memory scheduling algorithms, such as FCFRFS-WQF in our experimental environment [Hay et al. 2011; Jiang et al. 2012a]. Figure 11 shows the read latency results of 16 benchmarks. In general, MaxPB outperforms DCW, FNW, and two-stage-write in all benchmarks. However, the improvements of dedup, facesim, vips, and zeusmp are not significant because the power budget utilization is already high in those benchmarks as shown in Figure 4 and there is not much room for parallelism improvement. Overall, MaxPB can get 67.7% read latency reduction compared with the baseline DCW on average, and shows 24.6% and 12.9% more read latency reduction on average compared

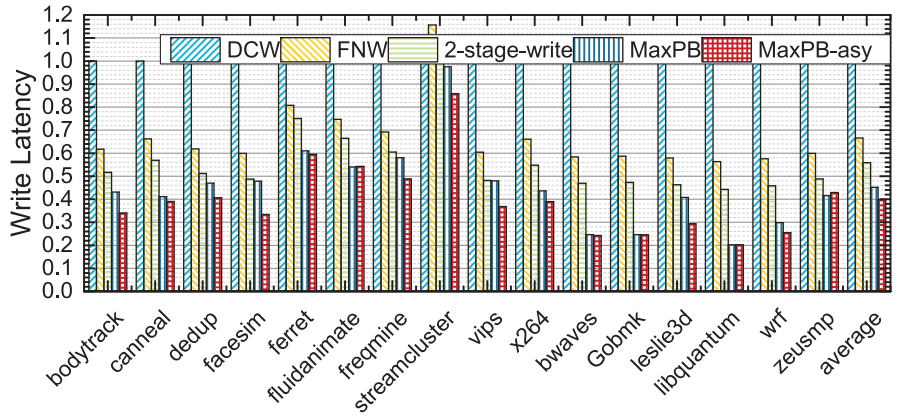


Fig. 12. Write latency.

with the FNW and two-stage-write, respectively. Moreover, by leveraging power asymmetry of SET and RESET, MaxPB-asy can get 32.0% and 20.3% more read latency reduction on average compared with the FNW and two-stage-write schemes.

4.5. Write Latency

MaxPB can significantly reduce the total number of write units and the overall write service time of a cache line can be reduced. The results of write latency of all workloads are illustrated in Figure 12. In general, MaxPB outperforms 44.9% more write latency reduction compared with conventional DCW on average. Moreover, MaxPB also shows 21.4% and 10.7% more write latency reduction compared with the state-of-the-art FNW and two-stage-write, respectively. MaxPB-asy shows 26.5% and 16.1% more write latency reduction on average compared with FNW and two-stage-write, respectively. One workload (streamcluster) gets little improvement or even performance degradation. There are two reasons that lead to this result. On one hand, streamcluster is not a memory-intensive workload; there are fewer memory read or write operations. There are only a small number of write requests in this workload. On the other hand, read requests are prior to write requests under the high-performance schedule algorithm. All these reasons result in the longer write latency compared with the baseline.

4.6. Speedup

We use the CPI (Cycles Per Instruction) to perform the system performance measurement. We use the DCW's CPI as the baseline and define variable *Speedup* as the following equation:

$$Speedup = \frac{CPI_{DCW}}{CPI}. \tag{7}$$

Compared with existing well-known PCM write schemes, MaxPB and MaxPB-asy show significant improvements in terms of CPI *Speedup*. As shown in Figure 13, similar to the results of read latency and write latency, MaxPB earns very good CPI improvement compared with state-of-the-art DCW, FNW, and two-stage-write write schemes. In general, MaxPB exhibits almost 2.6× CPI improvement and MaxPB-asy shows 2.9× CPI improvement compared with the baseline scheme. In comparison, FNW and two-stage-write have 1.6× and 1.9× speedup, respectively, compared with DCW on average. CPI improves due to the service time reduction of a cache line write. Benefiting from the read and write latency reduction, memory-access instructions finish faster than

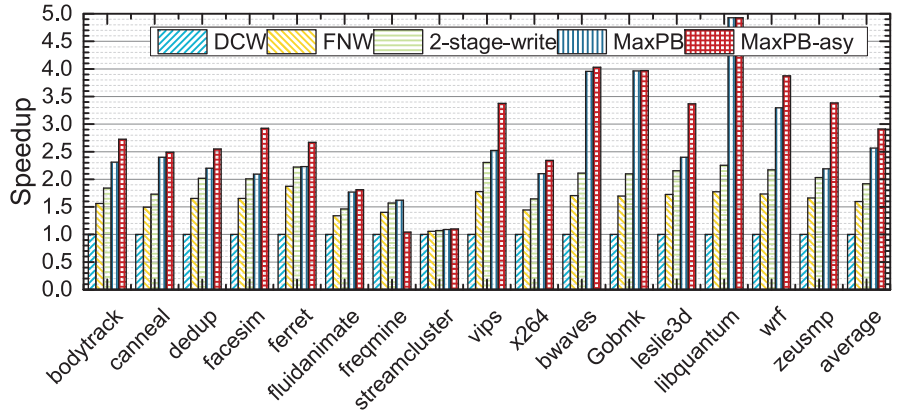


Fig. 13. Speedup.

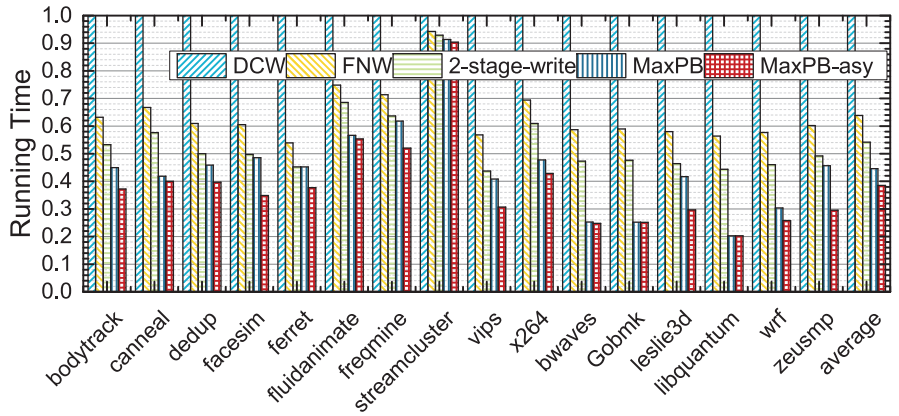


Fig. 14. Running time.

the baseline. Under the circumstances, more memory access instructions are sent and finished, and hence we can get better CPI improvement, especially in those workloads with heavy memory accesses (see all SPEC 2006 benchmarks used in this study).

4.7. Running Time

Workloads completion time is one of the key metrics to measure the whole system performance. As MaxPB and MaxPB-asy can significantly reduce the number of write units and shorten the write service time of a cache line under the power constraints, the workload running time can be also shortened. As shown in Figure 14, the experimental results show that MaxPB can get 45.5% running time reduction compared with DCW. More importantly, MaxPB outperforms the state-of-the-art schemes FNW and two-stage-write by 19.3% and 9.6% on average, respectively. By leveraging power asymmetry, MaxPB-asy can further shorten the running time. MaxPB-asy outperforms the state-of-the-art schemes FNW and two-stage-write by 24.3% and 15.6% on average, respectively.

4.8. Energy Consumption

Energy consumption is an important issue in both current big data centers and smart devices, such as smartphones, pads, etc.. In data centers, high-energy consumption

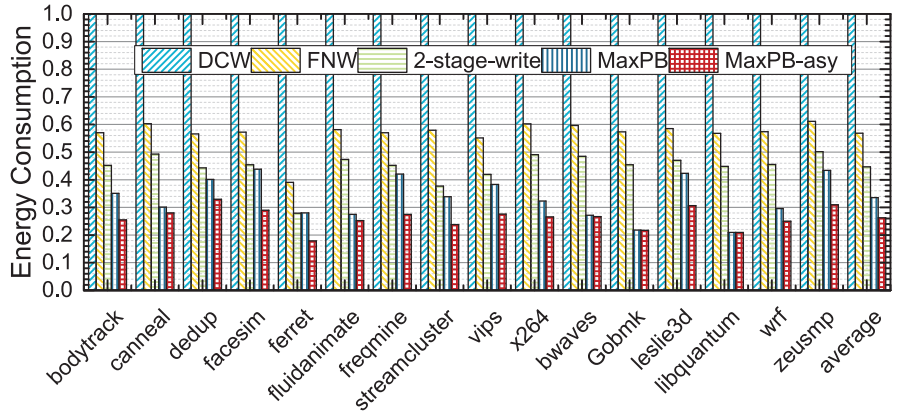


Fig. 15. Energy consumption.

put servers into security risks caused by heat, and numerous refrigerating devices are deployed to cool down the whole data center, which uses millions of watts of power per year. In smart devices, high-energy consumption leads to the reduction of use time and we have to charge the devices anytime anywhere, which leads to the decline of usability. Energy consumption reduction can bring significant benefits both to the environment and the economy. Even though MaxPB and MaxPB-asy do not reduce the amount of written data, that is, decrease the dynamic write energy consumption compared with FNW, our design can significantly reduce the cache line service time, the queueing time of each request, and hence reduce the standby power of the main memory system. As shown in Figure 15, MaxPB shows potential in energy consumption reduction. Compared with the DCW baseline, MaxPB shows 73.7% energy consumption reduction. Compared with the state-of-the-art FNW and two-stage-write schemes, MaxPB outperforms them by 23.3% and 11.1% on average, respectively. Moreover, MaxPB-asy outperforms the FNW and two-stage-write by 30.6% and 18.4% on average, respectively.

4.9. Sensitivity

To evaluate the effectiveness and efficiency of our designs under different memory configurations, we explored the design space of read latency, write latency, speedup, and running time based on a state-of-the-art server system. In general, modern systems contain four channels of memory, each channel containing two ranks of memory, each of which contains eight banks of memory, with a standard LPDDR memory bus (800MT/s).

In general, our designs still show much good read latency optimization compared with the baseline. On average, MaxPB shows 72.5% read latency reduction and MaxPB-asy presents 77.1% read latency reduction compared with the scheme without any optimizations, as shown in Figure 16. The result of the write latency is similar to that of the read latency. As shown in Figure 17, MaxPB gains 40.4% write latency reduction compared with the baseline. In comparison, MaxPB-asy gets 52.3% write latency decrease. The result of CPI improvement is shown in Figure 18. MaxPB shows 3.63× CPI improvements, while MaxPB-asy can earn 3.95× CPI improvements compared with the baseline. As for the running time of applications, the running time under MaxPB scheme is 27.5% that of the baseline as shown in Figure 19. On the other side, the running time under MaxPB-asy scheme is 25.2% that of the baseline. In short, our designs, including MaxPB and MaxPB-asy, are proven to be scalable. In other words, our proposed approaches are efficient and effective when memory scalability grows.

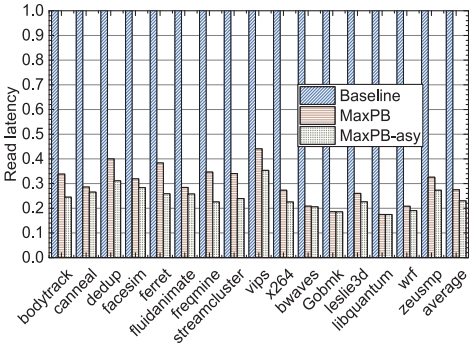


Fig. 16. Read latency sensitivity.

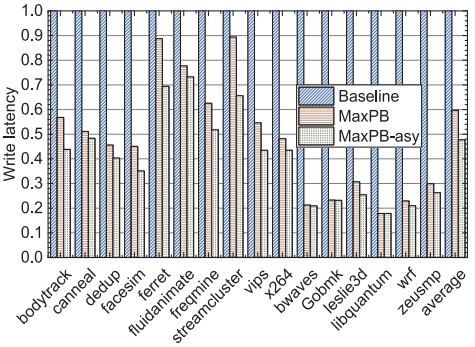


Fig. 17. Write latency sensitivity.

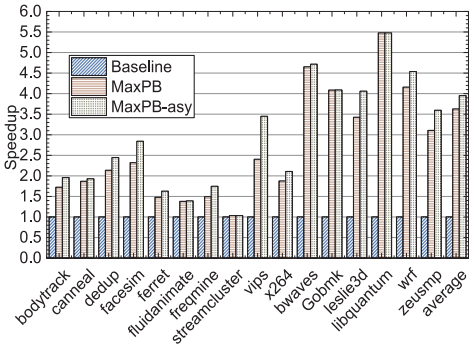


Fig. 18. IPC improvement sensitivity.

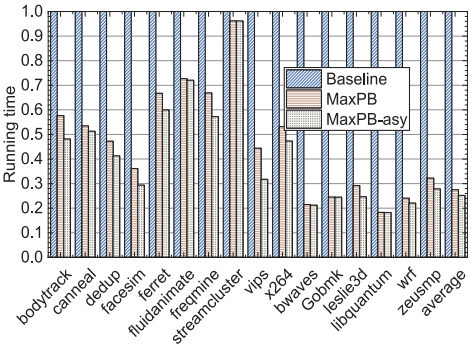


Fig. 19. Running time sensitivity.

5. RELATED WORK

5.1. Read and Write Optimizations

PCM faces many challenges such as poor endurance and unsatisfactory performance. Currently, numerous works focus on the read and write optimizations on PCM. Hoseinzadeh et al. [2014, 2016] try to reduce the access latency of MLC PCMs by leveraging the circumstance that the read latency of MSB (most significant bit) is almost the same to SLC's. By using striped bit mapping, consecutive data blocks are rearranged into different cell groups. The data blocks with odd address have lower read latency (the same to the latency of SLC) compared with data blocks with even address. As a result, the critical read latency of MLC PCM is highly optimized. Nair et al. [2015] proposed two mechanisms, that is, early read and turbo read, to reduce the read latency of PCM. Early read reduces the typical sensing time of target cell to reduce the read latency. However, it causes a potential data error problem. In order to address this problem, the authors implement a simple error correction method with the help of Berger Codes. Turbo read reduces the read latency by taking advantage of higher sensing voltage. Xia et al. [2014] proposed a write method named DWC (Dynamic Write Consolidation) based on the observation that only a small fraction of data are modified in a cache line. On one hand, DWC ignores the burst writes with unmodified data bits. On the other hand, DWC consolidates multiple burst write into one bus operation to reduce the write service time. Luo et al. [2015] migrated the DWC to MLC PCM considering the write asymmetries between different states. Qureshi et al. [2012]

proposed PreSET method to address the slow write problem of PCM by leveraging the asymmetries between RESET and SET operations. In general, RESET is much faster than SET. The key idea of the design is to send a PreSET request to the memory line if data is dirty in the last-level cache. The function of PreSET request is to SET all cells and thus the write latency decreases to T_{RESET} when written to this memory line next time. Zhang et al. [2015] proposed a TriState-SET on MLC PCM. Similar to PreSET, TriState-SET utilizes the asymmetries between different MLC states. There are also some works aimed at read and write boosted based on FPC (Frequent Data Compression) & BΔI coding [Palangappa and Mohanram 2016] and WoM (write once memory) coding [Zhang et al. 2013; Li and Mohanram 2014].

5.2. Parallelism Improvement

A lot of work tries to fix the slow write problem by improving the write parallelism. In general, these papers mainly focus on some critical factors related to the memory requests scheduling, such as the hierarchical architecture of memory, address mapping schemes, workload's access patterns and locality as well as the cache locality. Zhao et al. [2014] proposed a novel memory scheduler design for the persistent memory. In order to maintain the data persistence, a lot of methods are adopted to ensure the ordering of write requests. Persistent applications exhibit special access characteristics and introduce a number of read or write requests to contiguous memory regions, which results in low bank-level parallelism and low system throughout. The authors proposed FIRM to overcome this problem. The key ideas of FIRM are to distribute the president requests to many individual banks and reschedule the read and write requests to minimize the overhead caused by the write drain. Yue and Zhu [2013b] take advantage of subarray level parallelism to hide the long write latency caused by the existence of write units based on the insights that read can be executed with write if there are no subarray conflicts since the power need of read is much less than write's. Stuecheli et al. [2010] proposed Virtual Write Queue to address the bottleneck of memory bus utilization, especially in multicore architecture. By exploring the LLC and memory codesigned architecture, the memory characteristics are visible to the cache-level policies. In other words, the cache policies are memory-centric rather than memory-ignored. Lee et al. [2010b] provided a simple but effective LLC write back policy to relieve the performance degradation caused by write-caused interference. Write-caused interference is a phenomenon that (1) long write latency causes significant delays of more critical read requests and it also takes time for memory bus to do the read and write switching. By exploiting DRAM row buffer locality, more write requests hit in the row buffer and the service time is hence reduced. Yoon et al. [2015] presented a method to boost the performance of multilevel cell phase-change memories. By exploiting the asymmetries of MSB (Most Significant Bit) and LSB (Least Significant Bit) in MLC PCM, the memory is decoupled and divided into a FR (Fast Read) region and a FW (Fast Write) region. Moreover, a prediction technique is also proposed to map read-intensive pages to the FR region and write-intensive pages to the FW region. Zhou et al. [2016] proposed an effective requests scheduler based on the multipartition PCM architecture. The key idea of this design is to take advantage of the partition-level parallelism with some certain restrictions to improve the overall system performance and reduce application running time. Qureshi et al. [2010a] proposed write cancellation and write pausing methods to reduce the interference in read operation. MLC PCM uses multiple iterative operations and it provides an opportunity to pause and restart write operations within a physical cell. In short, write requests are paused if a read request arrives within the same memory bank, and the read request will be served first. After that, the paused write is restarted again. Qureshi et al. [2010b] proposed a morphable memory system based on the convertible property of MLC PCM that MLC cells can be treated as SLC cells with storing

fewer bits per cell. Observing that the memory capacity is often in excess of the requirement, more cells can be set to SLC-mode to get better memory performance. When the application is memory-hungry, it can be stored in MLC cells to get full memory capacity.

5.3. Related to Power Budget Utilization

There are many schemes focusing on the write performance improvement of PCM especially based on the power budget model. FNW [Cho and Lee 2009] is designed for write performance improvement on PCM. FNW uses a lightweight read-before-write scheme to reduce the amount of written data. FNW first reads the data to be written and encodes the data with an extra bit if the number of different bits is more than half compared with the original data. Under the same power limitation, two times the amount of data can be written in parallel, that is, the size of the write unit is doubled. Two-stage-write [Yue and Zhu 2013a] splits the PCM write process by taking into account the PCM write properties. The write is formed by two independent stages: stage "0" and stage "1." In stage "0," all zero bits in every unit are processed very quickly for RESET is much faster than SET. In stage "1," all ones are served with improved parallelism since the current need of SET a cell is only half of RESET a cell. To further enhance the parallelism of stage "1," new data are inverted if the number of "1" bits is more than half of bits to be served. Thus, the number of units that served in parallelism at stage "1" is doubled again under the same power constraint. There is no extra read operation overhead compared with the FNW scheme. Power-token-based method [Hay et al. 2011] improves the write concurrency by leveraging fine-grained power tokens management. Under the power limitations of one chip or a set of PCM chips, memory controllers can send more write commands to PCM by leveraging the bank-level parallelism, and the overall latency can be significantly reduced. To monitor the actual power consumption of each write back operation, the LLC is modified to record the count of different bits compared with data stored in PCM. FPB [Jiang et al. 2012a] tries to migrate the power-token-based write scheme to MLC PCM. By combining with the special program-and-verify iterations, the write parallelism can be significantly improved and FPB can improve the overall throughput of MLC PCM. Bit-Mapping [Du et al. 2013b] tries to make the data distribution among cell groups in a balanced way in order to get almost identical service time among different cell groups. Three-stage-write [Li et al. 2015] tries to combine FNW with two-stage-write to get further parallelism improvement as well as write latency reduction.

Comparisons of existing PCM write schemes are concluded in Table IV. In general, DCW, Flip-N-Write, two-stage-write, three-stage-write, power-token-based, MaxPB, and MaxPB-asy methods are designed for SLC PCM devices. FPB is designed for MLC PCM and the bit-mapping method can be used for SLC and MLC PCM. Power-token-based, FPB, and bit-mapping methods are designed at the memory controller level; they focus on write concurrency to pursue more write commands or requests be executed in parallel. However, other methods, like FNW, two-stage-write, MaxPB, and MaxPB-asy, are focusing on how to finish the chip-level write parallelism and how to write data from the on-chip buffer to PCM physical array quickly.

Unlike the state-of-the-art chip-level PCM write schemes, such as FNW, two-stage-write, and three-stage-write, MaxPB aims at different design goals. The differences are observed in Table IV. The key idea of FNW is to utilize the dissimilarity between stored and to-be-written data. If more than half of the total bits have to be written, the new data will be flipped with an extra bit to index it. FNW can double the write unit size and improve the write parallelism. Two-stage-write focuses on both the time and power asymmetry of writing zero and one to accelerate write. MaxPB exploits insights that the number of bits changed in each data unit is little and the power is often excessively supplied. MaxPB tries to write more data in one write unit to maximize the power

Table IV. Comparison of Existing PCM Write Schemes

	Design Level	Target	Main Ideas	Goals
DCW [Yang et al. 2007]	Chip	SLC	Write the different bits only	Reduce the energy consumption
Flip-N-Write [Cho and Lee 2009]	Chip	SLC	Add data inversion process and double the write unit size	Improve the write performance and reduce the energy consumption
Two-Stage-Write [Yue and Zhu 2013a]	Chip	SLC	Leveraging PCM asymmetries	Improve the write performance
Three-Stage-Write [Li et al. 2015]	Chip	SLC	Combining Flip-N-Write with Two-Stage-Write	Decreasing the write latency while reducing the power consumption
Power-token-based [Hay et al. 2011]	Controller	SLC	Manage power allocation at finer granularity	Improve the system performance and reduce the latency of memory requests
FPB [Jiang et al. 2012a]	Controller	MLC	Manage power allocation based on MLC PCM write properties	Improve write throughput and system performance
Bit-Mapping [Du et al. 2013b]	Controller	SLC or MLC	Data bits redistribution among cell groups in a balanced way	Balance the data amount and reduce latency of write request
MaxPB	Chip	SLC	Maximize the chip level pow er budget utilization	Reduce the chip-level write service time
MaxPB-asy	Chip	SLC	Maximize the chip level power budget utilization considering asymmetries	Reduce the chip-level write service time

budget utilization and hence reduces the total number of write units, which is the key performance bottleneck of PCM write.

5.4. Optimizations in Other NVMs

There are also some related works that provide ways to handle read or write problems in other NVMs such as STT-RAM and RRAM. Some issues can be migrated to PCM. Kültürsay et al. [2013] explored the designs of pure STT-RAM-based main memory. Experimental results show that it gets near performance and much lower energy consumption with DRAM-based memory with the help of some optimization technologies (partial write and row buffer write bypass). Smullen et al. [2011] presented a novel STT-RAM cache design to address the problems of high dynamic energy and long write time. By lowering the write current, the dynamic write energy is reduced. To address the potential problem raised by retention time reduction, a DRAM-like refresh method is proposed. Yoon et al. [2012] presented a novel hybrid memory design by leveraging both row buffer locality and data access patterns. Based on the observations that the access latency in PCM is as fast as DRAM if row buffer hits and is much slower if row buffer misses, the authors presented a row buffer locality-aware caching policy. It puts data most likely to have row buffer hits in PCM to reduce the performance degradation and improve energy efficiency. Meza et al. [2012b] explored the opportunities and challenges in adopting a row buffer in the NVMs architectural designs. With the increasing amount of data and improved parallel architecture, the data accesses locality becomes weaker and the energy consumption is increased due to the large row buffer area. The authors evaluate that a small row buffer in emerging NVM can achieve better energy efficiency without significant performance degradation. Zhang et al. [2016] proposed a novel requests scheduler based on the observation that the crossbar array of RRAM can be divided into many virtual regions with different access latency. By remapping memory requests into "fast" regions, the access performance of RRAM array can be significantly improved. Maddah et al. [2015] presented a data inversion method named CAFO for energy and endurance-asymmetric memories such as STT-RAM and PCM. By exploring the least cost of data inversion model, CAFO selectively selects the flip method of "lowest" energy or latency cost.

6. CONCLUSION

PCM is one of the most promising technologies for some outstanding characteristics (e.g., better technology scalability and low idle power consumption). However, it still suffers from a low write performance problem because the size of concurrent bit-writes are limited, which is caused by the high power requirement of write and results in many sequentially executed write units. In this article, we propose a novel PCM scheme named MaxPB, which tries to accelerate write by maximizing the power budget utilization with the least number of write units. MaxPB explores and exploits the key insights that existing state-of-the-art write schemes, such as FNW and two-stage-write, are unaware of low power budget utilization and still need many continuously executed write units to finish a cache line write back, which results in high write latency and overall system performance degradation. The main idea of MaxPB is to rearrange all original data units according to their actual power needs and write more data in one write unit. MaxPB can significantly reduce the number of write units in a cache line write service, which is the key bottleneck of the poor write performance of PCM. Experimental results of 16 PARSEC and SPEC workloads show that MaxPB gets 32.0% and 20.3% more read latency reduction, 26.5% and 16.1% more write latency reduction, 24.3% and 15.6% more running time decrease, 1.32× and 0.92× speedup, as well as 30.6% and 18.4% more energy consumption reduction on average compared with the state-of-the-art FNW and two-stage-write write schemes, respectively. Our future work is to

migrate MaxPB to the MLC PCM, which has quite different read and write properties compared with SLC PCM.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and editors for their positive and constructive comments and suggestions. We also thank anyone who helped us improve this article.

REFERENCES

Christian Bienia. 2011. *Benchmarking Modern Multiprocessors*. Ph.D. dissertation. Princeton University.

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, and others. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.

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 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 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*. IEEE, 46–48.

Yu Du, Miao Zhou, Bruce Childers, Rami Melhem, and Daniel Mossé. 2013a. Delta-compressed caching for overcoming the write bandwidth limitation of hybrid main memory. *ACM Transactions on Architecture and Code Optimization (TACO)* 9, 4 (2013), 55.

Yu Du, Miao Zhou, Bruce R. Childers, Daniel Mossé, and Rami Melhem. 2013b. Bit mapping for balanced PCM cell programming. *ACM SIGARCH Computer Architecture News* 41, 3 (2013), 428–439.

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 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 186–195.

John L. Henning. 2006. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News* 34, 4 (2006), 1–17.

Morteza Hoseinzadeh, Mohammad Arjomand, and Hamid Sarbazi-Azad. 2014. Reducing access latency of MLC PCMs through line striping. *ACM SIGARCH Computer Architecture News* 42, 3 (2014), 277–288.

Morteza Hoseinzadeh, Mohammad Arjomand, and Hamid Sarbazi-Azad. 2016. SPCM: The striped phase change memory. *ACM Transactions on Architecture and Code Optimization (TACO)* 12, 4 (2016), 38.

Fangting Huang, Dan Feng, Wen Xia, Wen Zhou, Yucheng Zhang, Min Fu, Chuntao Jiang, and Yukun Zhou. 2016. Security RBSG: Protecting phase change memory with security-level adjustable dynamic mapping. In *Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 1081–1090.

Lei Jiang, Yu Du, Bo Zhao, Youtao Zhang, Bruce R. Childers, and Jun Yang. 2013. Hardware-assisted cooperative integration of wear-leveling and salvaging for phase change memory. *ACM Transactions on Architecture and Code Optimization (TACO)* 10, 2 (2013), 7.

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*. IEEE, 1–12.

Lei Jiang, Bo Zhao, Youtao Zhang, Jun Yang, and Bruce R. Childers. 2012b. Improving write operations in MLC phase change memory. In *Proceedings of the 2012 IEEE 18th International Symposium on High Performance Computer Architecture (HPCA 2012)*. IEEE, 1–10.

Ron Kalla, Balaram Sinharoy, William Starke, and Michael Floyd. 2010. POWER7TM: IBM's next generation server processor. *IEEE Micro* 30, 2 (2010), 7–15.

Sangbeom Kang, Woo Yeong Cho, Beak-Hyung Cho, Kwang-Jin Lee, Chang-Soo Lee, Hyung-Rok Oh, Byung-Gil Choi, Qi Wang, Hye-Jin Kim, Mu-Hui Park, and others. 2007. A 0.1-μm 1.8-v 256-mb phase-change random access memory (pram) with 66-mhz synchronous burst-read operation. *IEEE Journal of Solid-State Circuits* 42, 1 (2007), 210–218.

Emre Kültürsay, Mahmut Kandemir, Anand Sivasubramaniam, and Onur Mutlu. 2013. Evaluating STT-RAM as an energy-efficient main memory alternative. In *Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 256–267.

Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting phase change memory as a scalable dram alternative. *ACM SIGARCH Computer Architecture News* 37, 3 (2009), 2–13.

Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2010a. Phase change memory architecture and the quest for scalability. *Communications of the ACM* 53, 7 (2010), 99–106.

Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger. 2010c. Phase-change technology and the future of main memory. *IEEE Micro* 30, 1 (2010), 143–143.

Chang Joo Lee, Veynu Narasiman, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt. 2010b. *DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems*. HPS Technical Report, TR-HPS-2010-002. Carnegie Mellon University.

Kwang-Jin Lee, Beak-Hyung Cho, Woo-Yeong Cho, Sangbeom Kang, Byung-Gil Choi, Hyung-Rok Oh, Chang-Soo Lee, Hye-Jin Kim, Joon-Min Park, Qi Wang, and others. 2008. A 90 nm 1.8 V 512 Mb diode-switch PRAM with 266 MB/s read throughput. *IEEE Journal of Solid-State Circuits* 43, 1 (2008), 150–162.

Jiayin Li and Kartik Mohanram. 2014. Write-once-memory-code phase change memory. In *Proceedings of the 2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6.

Yanbin Li, Xin Li, Lei Ju, and Zhiping Jia. 2015. A three-stage-write scheme with flip-bit for PCM main memory. In *Proceedings of the 2015 20th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 328–333.

Zheng Li, Fang Wang, Dan Feng, Yu Hua, Wei Tong, Jingning Liu, and Xiang Liu. 2016a. Tetris write: Exploring more write parallelism considering PCM asymmetries. In *Proceedings of the 2016 45th International Conference on Parallel Processing (ICPP)*. IEEE, 159–168.

Zheng Li, Fang Wang, Yu Hua, Wei Tong, Jingning Liu, Yu Chen, and Dan Feng. 2016b. Exploiting more parallelism from write operations on PCM. In *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 768–773.

Huizhang Luo, Liang Shi, Mengying Zhao, Qingfeng Zhuge, and Chun Jason Xue. 2015. Improving MLC PCM write throughput by write reconstruction. In *Proceedings of the 2015 IEEE Non-Volatile Memory System and Applications Symposium (NVMSA)*. IEEE, 1–6.

Rakan Maddah, Seyed Mohammad Seyedzadeh, and Rami Melhem. 2015. Cafo: Cost aware flip optimization for asymmetric memories. In *Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 320–330.

Justin Meza, Jichuan Chang, HanBin Yoon, Onur Mutlu, and Parthasarathy Ranganathan. 2012a. Enabling efficient and scalable hybrid memories using fine-granularity DRAM cache management. *IEEE Computer Architecture Letters* 11, 2 (2012), 61–64.

Justin Meza, Jing Li, and Onur Mutlu. 2012b. *Evaluating Row Buffer Locality in Future Non-Volatile Main Memories*. SAFARI Technical Report, TR-SAFARI-2012-002. Carnegie Mellon University.

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 *Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 309–319.

Hyung-Rok Oh, Beak-hyung Cho, Woo Yeong Cho, Sangbeom Kang, Byung-gil Choi, Hye-jin Kim, Ki-sung Kim, Du-eung Kim, Choong-keun Kwak, Hyun-geun Byun, and others. 2006. Enhanced write performance of a 64-Mb phase-change random access memory. *IEEE Journal of Solid-State Circuits* 41, 1 (2006), 122–126.

Poovaiah M. Palangappa and Kartik Mohanram. 2016. CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 90–101.

Gaetano Palumbo and Domenico Pappalardo. 2010. Charge pump circuits: An overview on design strategies and topologies. *IEEE Circuits and Systems Magazine* 10, 1 (2010), 31–45.

Matt Poremba and Yuan Xie. 2012. NVMain: An architectural-level main memory simulator for emerging non-volatile memories. In *Proceedings of the 2012 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 392–397.

Moinuddin K. Qureshi, Michele M. Franceschini, Ashish Jagmohan, and Luis A. Lastras. 2012. PreSET: Improving performance of phase change memories by exploiting asymmetry in write times. *ACM SIGARCH Computer Architecture News* 40, 3 (2012), 380–391.

Moinuddin K. Qureshi, Michele M. Franceschini, and Luis A. 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)*. IEEE, 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. *Acm Sigarch Computer Architecture News* 38, 3 (2010), 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 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 14–23.

Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, and Jude A. Rivers. 2009b. Scalable high performance main memory system using phase-change memory technology. *ACM SIGARCH Computer Architecture News* 37, 3 (2009), 24–33.

Simone Raoux, Geoffrey W. Burr, Matthew J. Breitwisch, Charles T. Rettner, Yi-Chou Chen, Robert M. Shelby, Martin Salinga, Daniel Krebs, S.-H. Chen, Hsiang-Lan Lung, and others. 2008. Phase-change random access memory: A scalable technology. *IBM Journal of Research and Development* 52, 4.5 (2008), 465–479.

ITRS Roadmap. 2013. International technology roadmap for semiconductors. *Semiconductor Industry Association* (2013).

Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. 2010. Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. *ACM SIGARCH Computer Architecture News* 38, 3 (2010), 383–394.

Clinton W. Smullen, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R. Stan. 2011. Relaxing non-volatility for fast and energy-efficient STT-RAM caches. In *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE, 50–61.

Jeffrey Stuecheli, Dimitris Kaseridis, David Daly, Hillery C. Hunter, and Lizy K. John. 2010. The virtual write queue: Coordinating DRAM and last-level cache policies. *ACM SIGARCH Computer Architecture News* 38, 3 (2010), 72–82.

Guangyu Sun, Yaojun Zhang, Yu Wang, and Yiran Chen. 2012. Improving energy efficiency of write-asymmetric memories by log style write. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*. ACM, 173–178.

Zhe Wang, Shuchang Shan, Ting Cao, Junli Gu, Yi Xu, Shuai Mu, Yuan Xie, and Daniel A. Jiménez. 2013. WADE: Writeback-aware dynamic cache management for NVM-based main memory system. *ACM Transactions on Architecture and Code Optimization (TACO)* 10, 4 (2013), 51.

J. Warnock, Y. Chan, W. Huott, S. Carey, M. Fee, Huajun Wen, M. J. Saccamango, Frank Malgioglio, P. Meaney, D. Plass, and others. 2011. A 5.2 GHz microprocessor chip for the IBM zenterprise system. In *Proceedings of the 2011 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 70–72.

H. S. Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P. Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E. Goodson. 2010. Phase change memory. *Proceedings of the IEEE* 98, 12 (2010), 2201–2227.

Fei Xia, Dejun Jiang, Jin Xiong, Mingyu Chen, Lixin Zhang, and Ninghui Sun. 2014. DWC: Dynamic write consolidation for phase change memory systems. In *Proceedings of the 28th ACM International Conference on Supercomputing*. ACM, 211–220.

Fei Xia, De-Jun Jiang, Jin Xiong, and Ning-Hui Sun. 2015. A survey of phase change memory systems. *Journal of Computer Science and Technology* 30, 1 (2015), 121–144.

Xilinx. 2015a. Vivado Design Suite User Guide. (June 2015). http://www.xilinx.com/support/documentation/sw_manuals/xilinx2015_2/ug973-vivado-release-notes-install-license.pdf.

Xilinx. 2015b. Zynq-7000 All Programmable SoC. (November 2015). http://www.xilinx.com/support/documentation/data_sheets/ds191-XC7Z030-XC7Z045-data-sheet.pdf.

Byung-Do Yang, Jae-Eun Lee, Jang-Su Kim, Junghyun Cho, Seung-Yun Lee, and Byoung-Gon Yu. 2007. A low power phase-change random access memory using a data-comparison write scheme. In *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems*. IEEE, 3014–3017.

HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael A. Harding, and Onur Mutlu. 2012. Row buffer locality aware caching policies for hybrid memories. In *Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 337–344.

Hanbin Yoon, Justin Meza, Naveen Muralimanohar, Norman P. Jouppi, and Onur Mutlu. 2015. Efficient data mapping and buffering techniques for multilevel cell phase-change memories. *ACM Transactions on Architecture and Code Optimization (TACO)* 11, 4 (2015), 40.

Jianhui Yue and Yifeng Zhu. 2012. Making write less blocking for read accesses in phase change memory. In *Proceedings of the 2012 IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 269–277.

Jianhui Yue and Yifeng Zhu. 2013a. Accelerating write by exploiting PCM asymmetries. In *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA'13)*. IEEE, 282–293.

Jianhui Yue and Yifeng Zhu. 2013b. Exploiting subarrays inside a bank to improve phase change memory performance. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 386–391.

Hang Zhang, Nong Xiao, Fang Liu, and Zhiguang Chen. 2016. Leader: Accelerating ReRAM-based main memory by leveraging access latency discrepancy in crossbar arrays. In *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 756–761.

XianWei Zhang, Le Jang, Youao Zhang, Chuanjun Zhang, and Jun Yang. 2013. WoM-SET: Low power proactive-SET-based PCM write using WoM code. In *Proceedings of the 2013 IEEE International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 217–222.

Xianwei Zhang, Youtao Zhang, and Jun Yang. 2015. TriState-SET: Proactive SET for improved performance of MLC phase change memories. In *Proceedings of the 2015 33rd IEEE International Conference on Computer Design (ICCD)*. IEEE, 659–665.

Jishen Zhao, Onur Mutlu, and Yuan Xie. 2014. FIRM: Fair and high-performance memory control for persistent memory systems. In *Proceedings of the 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 153–165.

Miao Zhou, Yu Du, Bruce Childers, Rami Melhem, and Daniel Mossé. 2012. Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems. *ACM Transactions on Architecture and Code Optimization (TACO)* 8, 4 (2012), 53.

Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. A durable and energy efficient main memory using phase change memory technology. *ACM SIGARCH Computer Architecture News* 37, 3 (2009), 14–23.

Wen Zhou, Dan Feng, Yu Hua, Jingning Liu, Fangting Huang, and Yu Chen. 2016. An efficient parallel scheduling scheme on multi-partition PCM architecture. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 344–349.

Received May 2016; revised October 2016; accepted October 2016