Exploiting More Parallelism from Write Operations on PCM

Zheng Li, Fang Wang[‡], Yu Hua, Wei Tong[‡], Jingning Liu, Yu Chen
and Dan Feng
Wuhan National Lab for Optoelectronics, School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, China
{lizheng, wangfang, csyhua, Tongwei, jnliu, chenyu0713, dfeng}@hust.edu.cn

***Abstract*—The size of write unit in PCM, namely the number of bits allowed to be written concurrently at one time, is restricted due to high write energy consumption. It typically needs several serially executed write units to finish a cache line service when using PCM as the main memory, which results in long write latency and high energy consumption. To address the poor write performance problem, we propose a novel PCM write scheme called Min-WU (Minimize the number of Write Units). The key idea behind Min-WU is to minimize the number of serially executed write units in a cache line service through sFPC (simplified Frequent Pattern Compression), eRW (efficient Reordering Write operations method) and fWP (fine-tuned Write Parallelism circuits). Using Min-WU, the zero parts of write units can be indicated with predefined prefixes and the residues can be reordered and written simultaneously under power constraints. Experimental results under PARSEC 2.0 workloads show that Min-WU reduces 32.5% running time and 48% energy as well as 44% latency compared with the conventional write scheme. When combined with partly data flip, the variation of Min-WU (Min-WU-PF) yields 12% running time reduction, 23% energy saving and 22% latency reduction compared with state-of-the-art Flip-N-Write.**

***Keywords*—PCM, write performance, write energy**

I. Introduction

Nonvolatile Memories (NVMs) such as Phase Change Memory (PCM), Magnetic Resistive RAM (MRAM) and Resistive Random Access Memory (RRAM) have better scalability with lower power consumption while DRAM scalability reaches its bottleneck [1]. PCM has extremely low leakage power and better capacity scalability, which allows PCM to be an attractive alternative of DRAM based main memory [2][3][4][5]. However, there are multiple technical problems in PCM. First, write performance is not satisfying (almost *10x* slower than DRAM). Second, endurance is still a weakness, i.e. 10^9 for PCM compared with 10^{15} for DRAM. In addition, although PCM does not need energy to do refresh operations, it suffers from high bit-write energy [3][6]. Due to power delivering challenge and serious power noise in PCM, the size of write unit in PCM is settled, namely the number of bits allowed to be written concurrently for one time is restricted [7]. As a result, all write operations must be performed serially in *write unit* [4][5]. The common sizes of *write unit* are 4, 8 and 16 bits, and it typically needs many serially executed write units to finish a cache line service when using PCM as the main memory, which results in long write latency and high energy consumption. As shown in Figure 1, assuming the cache line size is 64 bytes, the size of write unit is 16, and four PCM chips compose a memory bank. It takes $(64*8)/(16*4)=8$ write units for a cache line service [5][6][4][7][8].

To address the poor write problem of PCM, we propose a novel write scheme called Min-WU (Minimize the number of Write Units). The key idea behind Min-WU is to minimize the actual number of write units to accelerate the write operation. Min-WU has two main approaches: First, it reduces the total amounts of data by leveraging simple data coding. Second, it tries to finish the cache line service with fewer write units by encapsulating more data bits into a write unit. Min-WU design includes three components: **sFPC** (simplified Frequent Pattern Compression), **eRW** (efficient Reordering Write operations method) and **fWP** (fine-tuned Write Parallelism circuits). Min-WU strikingly minimizes the number of write units, which accelerates the write and reduces the energy consumption.

The main contributions of this paper are: i) A low overhead compression and data redistribution method called **sFPC**. We observe some frequent zero-extended values dominate the write data patterns in typical multi-threaded applications. Experimental results of 12 multi-threaded workloads show that zero-extended values occupy more than 60% of all memory accesses on average and up to 80% in some workloads. With sFPC, the amount of written data of each chip is reduced and balanced with low overhead (two cycles). ii) An efficient write reordering method named **eRW**. By reordering the execution sequence of write in the descending order of power demand, we get the fewer number of write units and hence improve the write performance. iii) Fine-tuned hardware circuits called **fWP** to support more write parallelism with low overhead and small hardware changes.

Based on the results under 12 multi-threaded PARSEC 2.0 workloads, Min-WU can gain 43% latency reduction, 31% running time decrease and 46% energy saving on average compared with the conventional write scheme. The variation of Min-WU, Min-WU-PF (Min-WU with partly data flip) yields 22% more latency reduction, 12% more running time reduction and 23% less energy consumption compared with the state-of-the-art FNW. In addition, Min-WU and Min-WU-PF have the great potential in multi-threaded workloads.

The remainder of this paper is structured as follows. Section II describes the background and the details of system design. Section III presents and analyzes the experimental results. Section IV introduces the related work. Finally, section V offers conclusions.

[‡] Co-corresponding authors:
Wei Tong (Tongwei@hust.edu.cn) and Fang Wang (wangfang@hust.edu.cn)

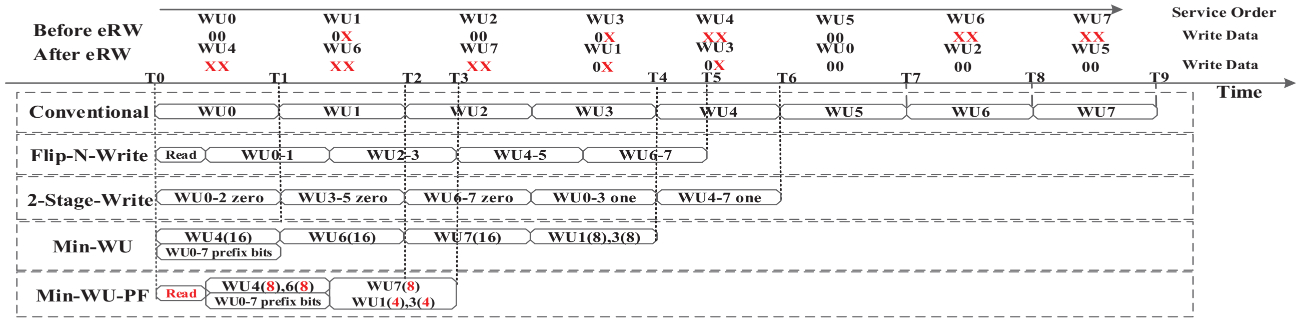

Fig. 1: Timing diagram for different schemes (Each '0' or 'X' refers to the value of one byte (8 bits)) Assuming power budget is 16 and the value following each WU presents the write unit power use in the worst case

II. The System Design

A. Background

PCM exploits the unique behavior of chalcogenide glass, such as Ge2Sb2Te5 (GST), in a memory cell to store digital information. Resistance varies hugely between crystalline and amorphous states, thus the current values are quite different at the same voltage level. Through a heating element, we can make the PCM cell amorphous via quickly heating and quenching the glass. Similarly, holding the glass in its crystallization temperature for a while can make the PCM cell crystalline. Meanwhile, *Write unit* limits the number of bytes can be written concurrently to memory banks at one time and it typically needs many serially executed write units to finish a cache line service, which greatly degrades the write performance. Assuming the cache line size is 64 bytes, *write unit* size is 16 bits and 4 PCM chips make up a memory bank. We need 8 serially executed write units $(64/(\frac{16}{8} * 4) = 8)$ to finish a cache line service [5][4][7][8][6].

Conventional write scheme regardless of the write values considers the power demand of each write unit in the worst case (all '1'). Therefore, it needs many serially executed write units to finish a cache line service. As shown in Figure 1, the constant in each write unit refers to the power requirement in the worst case. Write service of a cache line completes at T9 under the conventional scheme. Define T_{set} as the time to set a PCM cell, M refers to the number of total bits and N refers to the size of write unit. We can summarize the service time of a cache line service under the conventional write scheme as Equation 1.

$$T_{Conventional} = \frac{M}{N} T_{set} \tag{1}$$

We set M = 64 and N = 8 in our example, so $T_{conventional} = 8T_{set}$. In summary, many serially executed write units are the primary cause of poor write performance on PCM and the key to solve it is to minimize the number of write units.

B. FNW and 2-Stage-Write

Many write schemes try to expand the size of power budget to improve the poor write performance. FNW (Flip-N-Write) [5] first reads original data and compares original data with new data before writing. If more than half of bits need to be changed, FNW flips the new data. FNW uses one extra bit to mark whether the data has been flipped or not. With the data compression and a revision of hardware circuit, FNW doubles the size of write unit under power constraints and reduces the service time of writing a cache line. In the simple example as shown in Figure 1, all write units are finished at T5. The average service time of writing a cache line can be concluded in Equation 2. $T_{FNW} = 4.3T_{set}$ when M = 64, N = 8 and T_{set} is 3X longer than T_{read}.

$$T_{FNW} = T_{read} + \frac{M}{2 * N} T_{set} \tag{2}$$

2-Stage-Write [4] leverages the time and power asymmetry of writing "0" and "1" in PCM. 2-Stage-Write divides a write into two stages: stage "0" and stage "1". In stage "0", all "0" bits are written in a fixed speed. For writing "0" is much faster than writing "1", "0" stage can be finished quickly. In "1" stage, all "1" can be written in parallel for the current need of writing "1" is only half of writing "0". To achieve more parallelism, 2-Stage-Write flips new data if the number of bits "1" is more than half of total bits in the new data. 2-Stage-Write doubles the size of write unit in stage "1" again.

When writing "0" is 8X faster than "1", PCM-based main memory system can benefit from the service time reduction of a cache line but when the time ratio of "0" and "1" is shorter than 4X, 2-Stage-Write may not gain more significant write performance improvement than FNW. According to our experimental results on our real hardware prototype [9] as shown in Figure 2, the time ratio between writing "0" and "1" is about 3X (far below 8X). As shown in Figure 1, when the time ratio of writing "0" and "1" is 3, the "0" stage needs at least 3 write units ($\lfloor 8/3 \rfloor$) to finish writing all "0" and the cache line service is finished at T6. Assuming writing "0" is K times faster than writing "1" with L power needs,

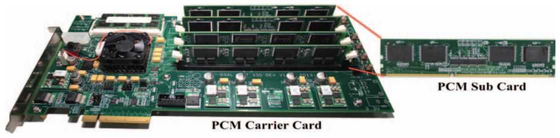

Fig. 2: The PCM hardware prototype

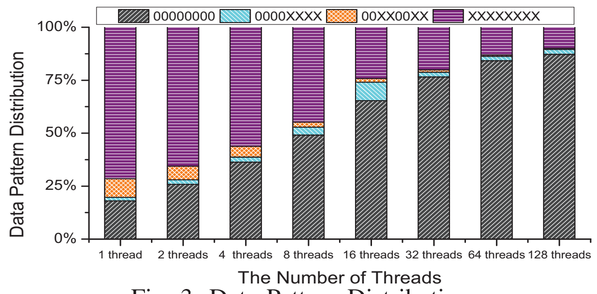

Fig. 3: Data Pattern Distribution

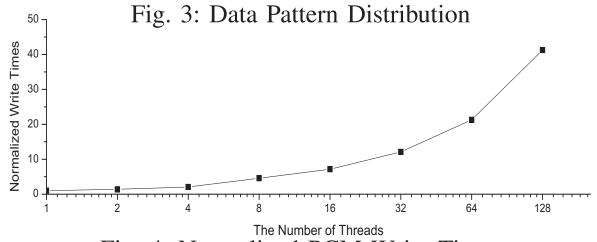

Fig. 4: Normalized PCM Write Times

TABLE I: sFPC Coding Scheme (Each '0' or 'X' refers to the value of one byte (8 bits))

Data type	Prefix Bits	Description	Data Example	After Compression	Data Size after Compression
$Type1$	00	All Zero	00000000	0	0 bits
$Type2$	01	4 Bytes Zero (i)	0000XXXX	XXXX	32 bits
$Type3$	10	4 Bytes Zero (ii)	00XX00XX	XXXX	32 bits
$Type4$	11	Uncompressed	XXXXXXXX	-	Original(64 bitl)

we can conclude the average service time as Equation 3. $T_{2-Stage-Write} = 5T_{set}$ when L = 2, K = 3, M = 64 and N = 8.

$$T_{2-Stage-Write} = \frac{M}{K*N}T_{set} + \frac{M}{2*N*L}T_{set} \qquad (3)$$

Another thing to note is 2-Stage-Write brings no advantages to bit-write reduction, which has potential benefits on endurance and energy improvement.

C. Important insights

Nowadays, more and more applications use multi-threaded programming. It grows commonplace that one application runs with hundreds of threads for taking full advantage of the abundant physical resources in a data center. Multi-threaded applications that exhibit evident access locality and some typical data patterns occupy a large fraction of memory data accesses [10][11][12]. One of the multi-threaded PARSEC 2.0 experiment results (blackscholes) is shown in Figure 3 and Figure 4, in which '0' or 'X' refers to the value of one byte (8 bits). '0' presents that the byte value is zero while 'X' shows there is at least one non-zero bit. We observe that data patterns 00000000, 00XX00XX and 0000XXXX occupy 22% up to more than 80% of the memory accesses with the increasing number of threads. We define these typical data values as **zero-extended values**. Zero-extended values occupy more than 60% of all memory accesses on average and up to 80% according to PARSEC 2.0 experiment results as shown in Figure 7. We obtain two useful insights from the experimental results:

- Zero-extended values dominate the write data in the multi-threaded workloads. The value property becomes more obvious with the increasing number of threads.
- The number of write increases with the increasing number of threads, which results in a potential negative influence on PCM lifespan.

All these data patterns may result from the data structure alignment [10]. Many small values are 4, 8, 16, 32 bits, but are stored in 64 bits for data structure alignment purpose to improve the memory access efficiency and it is necessary to insert some meaningless zero values [11]. It is demonstrated that SPECint95 and SPECint2000 benchmarks exhibit more than 40% zero values in memory accesses on average [12]. It also shows that the integer benchmarks exhibit more zero-extended values than floating point workloads, because of the differences of storage format between integer and float.

In short, multi-threaded applications present a typical trend that zero-extended values dominate the data patterns in memory accesses. It is important to utilize these special frequent values to gain more latency and energy reduction since multi-threaded programming will be the commonplace in data-centered processing in the future.

D. Min-WU

Min-WU (Minimize the number of Write Units) is quite different from FNW or 2-Stage-Write. Finding that multiple serially executed write units are the primary cause of the poor write performance, the key idea behind Min-WU is to utilize the frequent zero-extended values to minimize the number of write units. First, Min-WU reduces the total amounts of data by leveraging sFPC. Second, Min-WU tries to finish the cache line service with fewer write units by encapsulating more data bits into a write unit. Min-WU has three main components: 1) sFPC, a simplified FPC data coding and redistribution method, 2) eRW, efficient Reordering Write according to their power demand and 3) fWP, fine-tuned Write Parallelism circuits.

Min-WU first codes the write data based on its data patterns with sFPC. As shown in Table I, each '0' or 'X' refers to the value of one byte (8 bits), when data type is 00000000, 0000XXXX or 00XX00XX, the data amounts are reduced after sFPC with prefix bits indicating the zeroes in data values. For example, if the data value is 0000XXXX, data is compressed to XXXX after sFPC and the residual bytes 0000 can be indicated with the prefix bits "01". eRW is used for reordering write operations execution sequence to minimize the number of write units. After sFPC, as shown in Figure 1, eRW reorganizes the write execution sequence in the descending order of the prefix bits. After that, write units that can be performed concurrently under the power budget are sent to fWP. fWP provides hardware circuits supports and finishes all received write units in parallel under power constraints. In general, Min-WU can effectively reduce the number of write execution and significantly improve the write performance.

When the prefix bits of a write unit are "11", such as WU4, WU6 and WU7 in Figure 1, these units will occupy the whole power budgets in the worst case (all 16 in our example) and these write units can only be written sequentially under power constraints. So these units shall be serviced first. Under the same power constraints, two units that prefix bits are "01" or "10" can be serviced together for the power is halved after sFPC ($8 + 8 <= 16$). When the prefix bits of a write unit are "00", such as WU0, WU2 and WU5, there is no need to do the write for the data can be indicated with the prefix bits. In our example, all write units can be finished at T4. N_{type1}, N_{type2}, N_{type3} and N_{type4} are the numbers of various data types shown in Table I, respectively. The average service time of a cache line is concluded in Equation 4.

$$T_{Min-WU} = (\frac{N_{type2} + N_{type3}}{2} + N_{type4})T_{set} \qquad (4)$$

$T_{Min-WU} = 3T_{set}$ in our example. However, the performance degrades strikingly if the write data are $Type4$-dominant.

E. Min-WU-PF

To solve the performance degradation of Min-WU when write data is $Type4$-dominant, we propose a variation of Min-WU called Min-WU-PF (data Partly Flip). Min-WU-PF flips the data when data is not $Type1$ and more than half of bits have to be changed. Thus, we also double the write unit size and can get more parallelism improvement even data is $Type4$-dominant. As shown in Figure 1, the power use of WU4 and WU6 is halved after the simple data processing and they can be written in parallel under power constraints ($8 + 8 <= 16$). Likewise, WU7 can be serviced with WU1 and WU3 ($8 + 4 +$

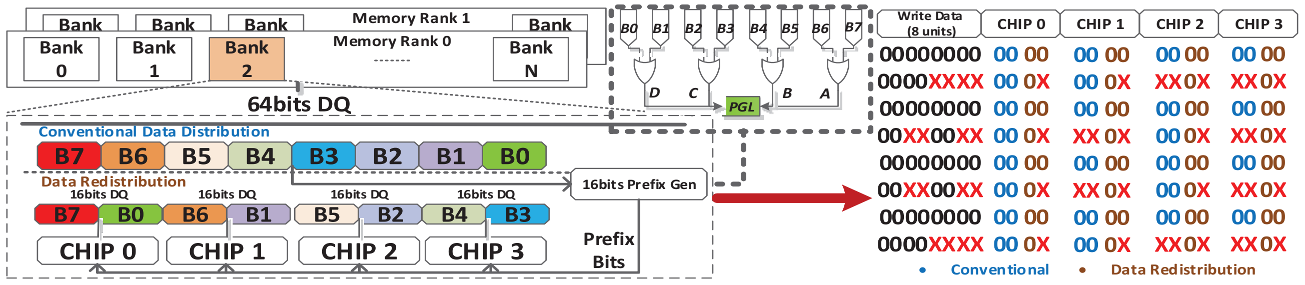

Fig. 5: Hardware Architecture

TABLE II: Differences of Various Write Schemes

Scheme	Key Idea	Reduce Latency	Reduce Energy
FNW	Difference of writing data and stored data	YES	YES
2-Stage-Write	Asymmetry of writing zero and one	YES	NO
Min-WU & Min-WU-PF	Minimize the number of write units	YES	YES

TABLE III: The truth table of prefix generation

ABCD	Prefix	ABCD	Prefix	ABCD	Prefix	ABCD	Prefix
0000	00	0010	01	0100	10	011X	11
0001	01	0011	01	0101	10	1XXX	11

$4 <= 16$). Min-WU-PF further enhances the write parallelism and reduces the number of write execution. As shown in Figure 1, all write units are finished at T3, i.e. $2T_{set}$, which is shorter than all above write schemes. The cache line service time of Min-WU-PF, i.e. T_{MWP}, is shown in Equation 5.

$$T_{MWP} = T_{read} + (\frac{N_{type2} + N_{type3}}{4} + \frac{N_{type4}}{2})T_{set} \qquad (5)$$

In summary, although FNW and 2-Stage-Write can reduce the write latency, they do not focus on reducing the number of write units. As concluded in Table II, FNW focuses on the differences of new data and stored data while 2-Stage-Write focuses on the time and power asymmetry of writing “1” and “0”. Differently, Min-WU and Min-WU-PF care about reducing the number of write units to accelerate write. All write schemes can reduce latency while 2-Stage-Write has no benefit on energy saving. When write data have many zero-extended values, Min-WU has good write performance improvement. Min-WU-PF performs better than both FNW and 2-Stage-Write even the data is $Type4$-dominant. Min-WU/Min-WU-PF can get more performance improvement since more multi-threaded programming will be used in data-centered processing and more zero-extended values will be obtained in the future. Besides, we can make customized sFPC scheme according to the learning of workloads rather than fixed data patterns.

F. Hardware Implementation

To support the write parallelism proposed in Min-WU/Min-WU-PF, we carry out carefully designed hardware architecture. The bit-writes of each chip may vary much when using Min-WU/Min-WU-PF under traditional system architecture as shown in Figure 5. The data amounts of each chip are quite different since Min-WU only writes “X” and “0” can be indicated by the prefix bits. Chip0, chip1 and chip2 have to wait for the completion of the heaviest bit-writes chip3 (most $Type4$ values ‘XX’), which results in low bandwidth utilization and long write service time. To address this problem, we redistribute the data as shown in Figure 5. In conventional memory architecture, chip(i) write Byte(7-2*i) and Byte(6-2*i). As we mentioned, the data amounts may vary much in each chip. In our design, while the data displays three mainly patterns,

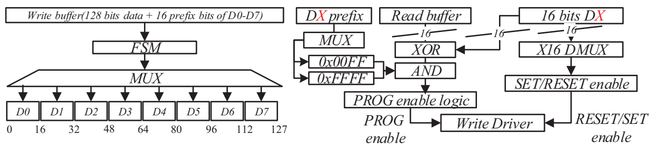

Fig. 6: Write Control Logic

we change the data distribution slightly. Chip(i) corresponds to Byte(7-i) and Byte(i) in our design. For example, chip0 corresponds to Byte7 and Byte0. Whatever the data type is 0000XXXX or 00XX00XX, all chips receive data 0X. There is no extra overhead in the data redistribution and all chips have the same amount of data no matter what the data is.

Prefix Generate Logic (PGL) is a low overhead prefix bits generation circuit and is illustrated in Figure 5. PGL can be released to a simple combinational logic circuit. Typically, we use an OR gate to judge whether the value of a byte is zero. PGL automatically creates the 2-bits prefix based on the values of A, B, C and D. The truth table of 2-bits prefix generation is summarized in Table III. The prefix can be generated quickly using multiplexer and adder circuits. According to our results performed in our real hardware prototype DSAL-SSD [9], only two cycles are needed to do the prefix generation. The low overhead prefix generation scheme won’t deliver any negative influence on the service time of a cache line.

To provide write parallelism supports, we carry out a fine-tuned hardware circuit named fWP based on an industrial prototype from Samsung [7]. Figure 6 shows how the write logic works. The on-chip write buffer stores 128 bits data and 16 bits prefix of these 8 write units (128/16 = 8). Shared Finite State Machine (FSM) decides which data units to be executed first (D0 to D7) according to the prefix values. If prefix bits of a write unit are ‘00’, this write unit won’t be sent for the prefix bits can imply the data value. FSM first sends whose prefix bits are ‘11’ because it takes the whole power budget in the worst case. Then the FSM chooses two DX whose prefix bits are ‘01’ or ‘10’ since no more than half of the total bits are changed after sFPC. To achieve independent bit control, we introduce an extra control signal named PROG similar to FNW. SET and RESET together with PROG signal activate the cell with little overhead (just an AND gate). PROG signals generation can be done easily with a Multiplexer and an AND gate. The

TABLE IV: Parameters of Simulation

Parameter	Value
CPU	4-Core CMP, 2GHz, ALPHA processor
L1 Cache	32KB I-cache, 32KB D-cache, 2 cycles latency
L2 Cache	8-way, 2MB, 64B cache line, 20 cycles latency
L3 Cache	16-way, 8MB, 64B cache line, 50 cycles latency
Memory Controller	FCFRFS scheduling algorithm
Memory Organization	4GB, 64 bits data width, 2 Ranks, 8 Banks
PCM Chips Organization	4 chips per bank, 8 bytes write unit size
Read/Write a cell time	50ns/153ns

TABLE V: Benchmarks

Benchmark	Introduction	Benchmark	Introduction
blackscholes	Option pricing with Black-Scholes PDE	fluidanimate	Fluid dynamics for animation purposes with SPH method
bodytrack	Body tracking of a person	freqmine	Frequent itemset mining
swaptions	Pricing of a portfolio of swaptions	canneal	Simulated cache-aware annealing to optimize routing cost of a chip design
dedup	Next-generation compression with data deduplication	streamcluster	Online clustering of an input stream
facesim	Simulates the motions of a human face	vips	Image processing
ferret	Content similarity search server	x264	H.264 video encoding

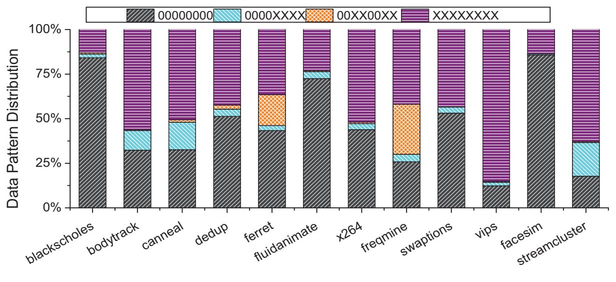

Fig. 7: Data Patterns of 12 multi-threaded workloads

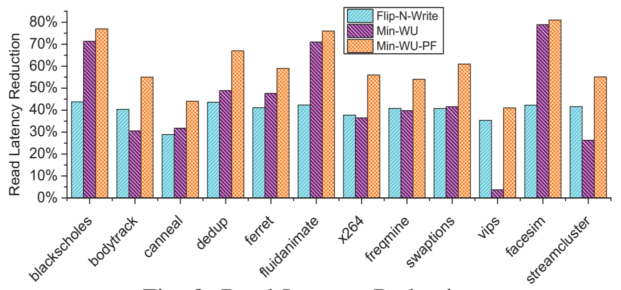

Fig. 8: Read Latency Reduction

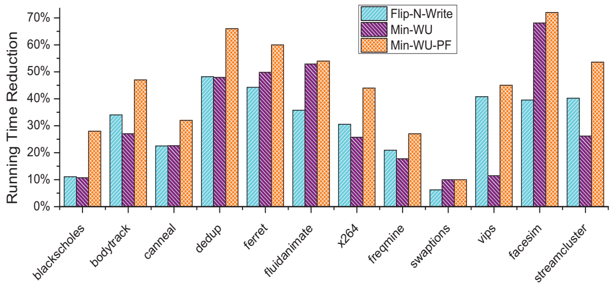

Fig. 9: Running Time Reduction

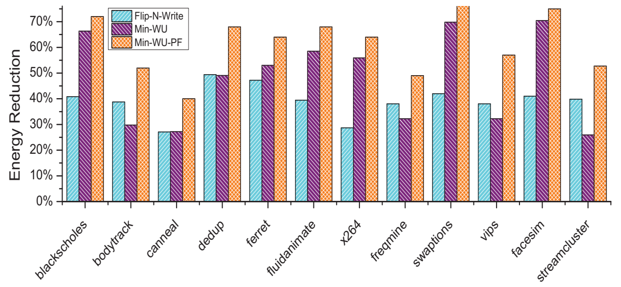

Fig. 10: Energy Improvement

input value of the AND gate is decided by the prefix bits. When prefix bits are '01' or '10', *0x00FF* is selected for only writing 'X' of data '0X'. Otherwise, *0xFFFF* is selected for writing XX value. Especially, Min-WU-PF first reads the new data and flips it if more than half of bits need to be changed. Min-WU-PF sends PROG enable only to the cells that need to be changed with a low overhead XOR gate.

III. Evaluation

In this section, we evaluate the efficiency of our design using multi-threaded PARSEC 2.0 benchmarks. We present the results of read latency, applications running time as well as energy. Specifically, we first present the parameters and the experimental environment. We use the event-driven GEM5 simulator [13] to evaluate our design and the simulation parameters are shown in Table IV. The parameters of PCM are taken from the prototype of Samsung published in [7] and the results from our DSAL-SSD hardware prototype with actual PCM chips provided by Micron [9]. Partly energy parameters are taken from CACTI [14] and Ref. [7]. We compare Min-WU/Min-WU-PF with state-of-the-art FNW with all 12 benchmarks under PARSEC 2.0 [15] without selectively choosing. The details of benchmarks are concluded in Table V. Our goal is to find the most suitable application scenarios of Min-WU and Min-WU-PF.

1) Data Patterns Distribution: We first measure all data patterns distribution of the PARSEC 2.0 benchmarks to verify the motivations. The results are shown in Figure 7. We observe that zero-extended values dominate the write values in all benchmarks and occupy more than 60% of all memory accesses on average. Three programs (blackscholes, fluidanmate and facesim) show more than 70% zero-extended values while the least one has more than 20% (vips). It proves that it is important to utilize these commonplace zero-extended values for write performance improvement and energy reduction.

2) Read Latency: Read latency is crucial for the main memory system performance and is the bottleneck of the whole system performance. We use the conventional PCM write scheme without any optimization as the baseline. Figure 8 shows the read latency reduction of Min-WU/Min-WU-PF and FNW compared with the baseline. Overall, Min-WU significantly outperforms FNW in some benchmarks while being equivalent in the others except vips. This reason is the write data is $Type4$-dominant in vips. Min-WU-PF outperforms FNW in all benchmarks. FNW can get 29%-43% read latency reduction compared with the baseline while Min-WU can get 32%-79% read latency improvement. Min-WU shows 6% more latency reduction considering the awful performance of vips. Min-WU-PF shows 61% read performance improvement compared with baseline and outperforms state-of-the-art FNW by 21% on average.

3) Running Time: Workloads completion time is one of the most important metrics of the whole system performance. The workloads running time results are shown in Figure 9. Min-WU/Min-WU-PF can significantly reduce the service time of a cache line service with writing more units currently under power constraints. The experimental results show that Min-WU/Min-WU-PF can gain 31%/45% running time reduction against the baseline on average, respectively. Moreover, Min-WU-PF outperforms FNW by 14% on average.

4) Energy Improvement: Energy improvement can bring significant benefits both to the environment and economy. As shown in Figure 10, although many workloads have small read latency improvement, they show a good energy consumption improvement. On one hand, Min-WU and Min-WU-PF decrease the bits need to be written to PCM cells with implementing sFPC. On the other hand, our designs significantly shorten the service time of requests and hence reduce the system's stand-by energy consumption. It is remarkable that Min-WU outperforms state-of-the-art FNW by more than 20% in five workloads. Min-WU earns 46% less energy compared with the baseline and outperforms FNW by 11% on average. Min-WU-PF reduces 62% energy consumption compared with the baseline and outperforms FNW by more than 22% on average.

IV. Related Work

FNW [5] utilizes the power budget for write parallelism improvement. If the different bits are more than half of the total bits, new data will be flipped. FNW doubles the write unit size under the power constraints and reduces the service time of write. FNW introduces an extra bit to store the status whether associated data have been flipped or not. 2-Stage-Write [4] leverages the time and power asymmetry of writing "0" and "1". Unlike FNW, 2-Stage-Write focuses on the values of new data and there is no extra read operation overhead. 2-Stage-Write divides a write process into 2 stages: stage 0 and stage 1. In stage 0, all "0" in every write unit can be finished in a settled speed. In stage 1, the write unit size is doubled for the write power need of "1" is only half of "0". Furthermore, if the number of "1" is more than half of the total bits, 2-Stage-Write flips the data and the write unit size of stage 1 is doubled again.

Compression is widely used in the capacity constrained cache and disk backup systems. Compression reduces the size of data and thus improves the valid capacity and reduces the high-cost paging from storage devices (such as disks) to main memory. Frequent Pattern Compression (FPC) divides a cache line into many words (typically 32bits a word) and compresses each word according to data pattern [11]. There are some works utilizing FPC to reduce the bit-writes in NVM. Recently in ref. [16], data compression agriculture of PCM is proposed to combine the FPC with memory controller to reduce the number of bit-writes, write energy and improve the endurance. Dgien et al. proposes a compression-based memory architecture in Nonvolatile Memories (NVMs) combining FPC with FNW [17]. With the finding that FNW cannot work efficiently if the data has been compressed, the author carries out a fine-gained FNW to get further bit-write reduction.

V. Conclusion

To address the poor write performance, we propose a novel write scheme called Min-WU. Finding that multiple serially executed write units are the primary cause of the poor write performance, the key idea behind Min-WU is to minimize the number of write units to accelerate the write operation. Min-WU has two main approaches: First, Min-WU reduces the total amounts of data by leveraging simple data coding. Second, Min-WU tries to finish the cache line service with fewer write units by encapsulating more data bits into a write unit. Min-WU design includes three components: **sFPC** (simplified Frequent Pattern Compression), **eRW** (efficient Reordering Write operations) and **fWP** (fine-tuned Write Parallelism circuits). Min-WU strikingly minimizes the number of write units, which accelerates the write while reduces the energy consumption of PCM. Min-WU is highly effective and efficient in improving the write performance and reducing the write energy consumption compared with state-of-the-art FNW.

All-around experimental results under 12 PARSEC 2.0 benchmarks demonstrate the efficiency of Min-WU and Min-WU-PF. Based on the results of 12 multi-threaded workloads, Min-WU can gain 43% read latency reduction, 31% running time decrease and 46% energy saving on average compared with the conventional write scheme without any optimization. Min-WU-PF yields 22% more latency reduction, 12% more running time reduction and 23% less energy than the state-of-the-art FNW scheme. In addition, our design has the great potential in multi-threaded applications.

Acknowledgment

This work was supported by the 863 Project No.2015AA015301, No.2013AA013203, No.2015AA016701; NSFC No.61173043, No.61303046, No.61472153; Wuhan Applied Basic Research Project (No.2015010101010004); This work was also supported by Key Laboratory of Information Storage System, Ministry of Education, China.

References

[1] L. Wilson, "International technology roadmap for semiconductors (ITRS)," *Semiconductor Industry Association*, 2013.

[2] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," *Proc.ISCA*, pp. 2–13, 2009.

[3] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in MLC phase change memory," *Proc.HPCA*, pp. 1–10, 2012.

[4] J. Yue and Y. Zhu, "Accelerating write by exploiting PCM asymmetries," *Proc.HPCA*, pp. 282–293, 2013.

[5] S. Cho and H. Lee, "Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance," *Proc.MICRO*, pp. 347–357, 2009.

[6] J. Yue and Y. Zhu, "Making write less blocking for read accesses in phase change memory," *Proc.MASCOTS*, pp. 269–277, 2012.

[7] K.-J. Lee, B.-H. Cho, W.-Y. Cho *et al.*, "A 90 nm 1.8 V 512 Mb diode-switch PRAM with 266 MB/s read throughput," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 150–162, 2008.

[8] J. Yue and Y. Zhu, "Exploiting subarrays inside a bank to improve phase change memory performance," *Proc.DATE*, pp. 386–391, 2013.

[9] Z. Li, S. Zhang, J. Liu, W. Tong, Y. Hua, D. Feng, and C. Yu, "A software-defined fusion storage system for PCM and NAND flash," *Proc.NVMSA*, pp. 1–6, 2015.

[10] M. Arjomand, A. Jadidi, A. Shafiee, and H. Sarbazi-Azad, "A morphable phase change memory architecture considering frequent zero values," *Proc.ICCD*, pp. 373–380, 2011.

[11] A. R. Alameldeen and D. A. Wood, "Frequent pattern compression: A significance-based compression scheme for L2 caches," *Dept. of Computer Sciences, University of Wisconsin-Madison, Tech. Rep*, 2004.

[12] J. Yang, Y. Zhang, and R. Gupta, "Frequent value compression in data caches," *Proc.MICRO*, pp. 258–265, 2000.

[13] N. Binkert, B. Beckmann, G. Black *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[14] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches," *HP Laboratories*, pp. 22–31, 2009.

[15] C. Bienia, "Benchmarking Modern Multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[16] P. M. Palangappa and K. Mohanram, "Flip-Mirror-Rotate: An architecture for bit-write reduction and wear leveling in non-volatile memories," *Proc.GLSVLSI*, pp. 221–224, 2015.

[17] D. B. Dgien, P. M. Palangappa, N. A. Hunter, J. Li, and K. Mohanram, "Compression architecture for bit-write reduction in non-volatile memory technologies," *Proc. NANOARCH*, pp. 51–56, 2014.