Time and Space-Efficient Write Parallelism in PCM by Exploiting Data Patterns

Zheng Li, Fang Wang, Dan Feng, *Member, IEEE*, Yu Hua, *Senior Member, IEEE*, Jingning Liu, Wei Tong, Yu Chen, and Salah S. Harb

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). We observe data access locality that some frequent zero-extended values dominate the write data patterns in typical multi-threaded applications (more than 40 and 44.9 percent of all memory accesses in PARSEC workloads and SPEC 2006 benchmarks, respectively). By leveraging carefully designed chip-level data redistribution method, the data amount is balanced and the data pattern is the same among all PCM chips. The key idea behind Min-WU is to minimize the number of serially executed write units in a cache line service after data redistribution 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. Our design can improve the performance, energy consumption and endurance of PCM-based main memory with low space and time overhead. Experimental results of 12 multi-threaded PARSEC 2.0 workloads show that Min-WU reduces 44 percent read latency, 28 percent write latency, 32.5 percent running time and 48 percent energy while receiving 32 percent IPC improvement compared with the conventional write scheme with few memory cycles and less than 3 percent storage space overhead. Evaluation results of 8 SPEC 2006 benchmarks demonstrate that Min-WU earns 57.8/46.0 percent read/write latency reduction, 28.7 percent IPC improvement, 28 percent running time reduction and 62.1 percent energy reduction compared with the baseline under realistic memory hierarchy configurations.

Index Terms—PCM, write unit, performance evaluation, write energy

1 Introduction

The data scale is growing rapidly. According to IDC's study, data we create and copy is about 4.4ZB in 2013 and will be 44ZB in 2020 [1]. Businesses are demanding faster and easier to access information for reliable and smart decisions. Wal-Mart handles more than 1 million transactions per hour and feeds databases estimated to be in PB scale. Facebook deals with 2.5 PB of user data and YouTube streams 48 hours of videos per minute [2]. The official train ticket site of China, 12306.cn, deals with 30 billion of PV (page views) on the peak day during 2015 Spring Festival travel rush [3]. Citigroup reported that in the financial business, every millisecond lost results in millions of dollars economic losses per annum [4]. Alibaba, the biggest online retailer of China, processes hundreds of millions of orders on "11.11" online shopping spree [5]. Data-intensive processing requires massive memory capacity. However, the supply of capacity is far behind the striking demands. DRAM scalability reaches its bottleneck and it is difficult to maintain the stabilization and reliability under 1X nm node [6]. On the other hand, DRAM-based memory contributes more than 40 percent of the total system power consumption, which has become the primary concern in current data centers [7], [8], [9], [10]. Google's data centers use around 260 million watts of power per year, which accounts to 0.01 percent of global energy and about a quarter of the output of a nuclear power plant [11].

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 [6]. PCM has extremely low leakage power and better scalable capacity, which allows PCM to be an attractive alternative of DRAM based main memory [12], [13], [14], [15], [16], [17]. However, there are multiple technical problems in PCM. First, write performance is not satisfying (almost *10x* slower than DRAM) [18]. Second, endurance is still a weakness, i.e., 10^9 for PCM compared with 10^{15} for DRAM [19], [20], [21], [22]. In addition, although PCM does not need energy to do refresh operations, it suffers from high bit-write energy [15], [23], [24], [25]. 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

• *The authors are with the Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China, Hubei, Sheng 430074, China. E-mail: {lizheng, wangfang, dfeng, csyhua, jnliu, Tongwei, chenyu0713, salah}@hust.edu.cn.*

Manuscript received 22 July 2016; revised 20 Feb. 2017; accepted 26 Feb. 2017. Date of publication 2 Mar. 2017; date of current version 15 Aug. 2017. Recommended for acceptance by G. Min.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2017.2677903

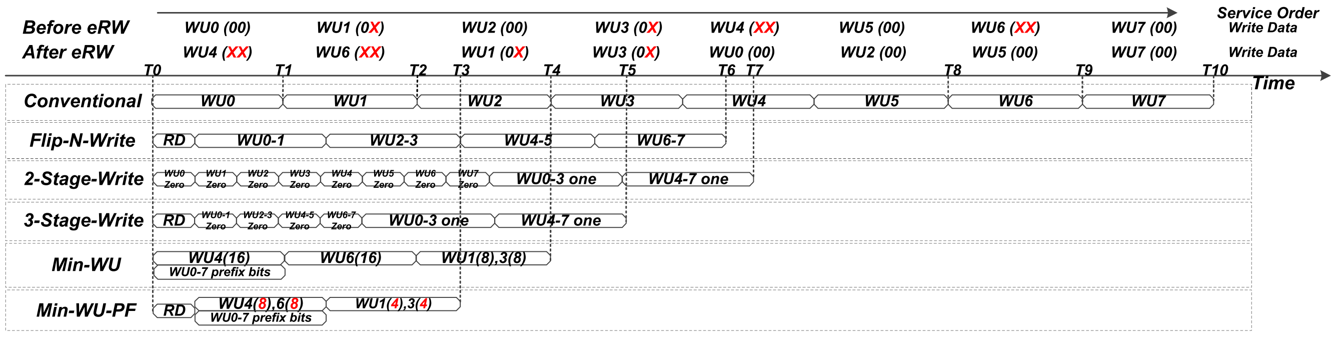

Before eRW
After eRW
WU0 (00)
WU4 (XX)
WU1 (0X)
WU6 (XX)
WU2 (00)
WU1 (0X)
WU3 (0X)
WU3 (0X)
WU4 (XX)
WU0 (00)
WU5 (00)
WU2 (00)
WU6 (XX)
WU5 (00)
WU7 (00)
WU7 (00)
Service Order
Write Data
Write Data
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10
Time
Conventional
WU0 WU1 WU2 WU3 WU4 WU5 WU6 WU7
Flip-N-Write
RD WU0-1 WU2-3 WU4-5 WU6-7
2-Stage-Write
WU0 Zero WU1 Zero WU2 Zero WU3 Zero WU4 Zero WU5 Zero WU6 Zero WU7 Zero WU0-3 one WU4-7 one
3-Stage-Write
RD WU0-1 Zero WU2-3 Zero WU4-5 Zero WU6-7 Zero WU0-3 one WU4-7 one
Min-WU
WU4(16) WU6(16) WU1(8),3(8)
WU0-7 prefix bits
Min-WU-PF
RD WU4(8),6(8) WU1(4),3(4)
WU0-7 prefix bits

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.

concurrently for one time is restricted [26]. As a result, all write operations must be performed serially in *write unit* [16], [17]. The common sizes of *write unit* are 4, 8 and 16 bits, and it typically requires 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 [27], [28], [29]. As shown in Fig. 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 $(64 \cdot 8)/(16 \cdot 4) = 8$ write units for a memory cache line service [16], [17], [23], [26], [30].

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 amount of data by leveraging typical patterns of write data. Second, it tries to finish a cache line service with less write units by encapsulating more data bits into one write unit. The main contributions of this paper are:

1) A novel PCM write scheme Min-WU with three critical components: *sFPC* (simplified Frequent Pattern Compression), *eRW* (efficient Reordering Write operations method) and *fWP* (fine-tuned Write Parallelism circuits). We observe *zero-extended values* dominate the data write patterns in typical applications (more than 40 and 44.9 percent of all memory accesses in PARSEC workloads and SPEC 2006 benchmarks, respectively) and Min-WU strikingly minimizes the number of write units, which accelerates the write and reduces the energy consumption with low overhead and small hardware changes.
2) Carefully designed hardware architecture for efficiently combining proposed methods (sFPC, eRW, fWPs) and PCM chips. By leveraging data redistribution, the amount of written data of each chip is balanced. Moreover, data-prefix-separation designs can effectively improve chip parallelism without losing the accuracy and reduce the overall space overhead of Min-WU and Min-WU-PF.

The rest of this paper is organized as follows. Section 2 describes the brief background, our motivations and the details of the proposed write schemes. Section 3 presents the hardware implementation. Section 4 presents and analyzes the experimental results. Section 5 introduces the related work. Finally, Section 6 offers conclusions.

2 The System Design

2.1 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 and 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. Compared with RESET and SET operations, the READ operation only needs a small current to identify the resistance level of the GST. Moreover, PCM write shows great power and time asymmetry that the minimum current needs and duration of RESET vary largely. The read and write processes of PCM are shown in Fig. 2. Meanwhile, Due to power delivering challenge, serious power noise inside PCM chip, the size of write unit in PCM is settled, namely the number of bits allowed to be written concurrently for one time is restricted [26]. As a result, all write operations must be performed serially in *write unit* [16], [17]. The common sizes of *write unit* are 4, 8 and 16 bits, and it 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.

Conventional write scheme regardless of the write values considers the power demand of each write unit in the worst case (all “0”). Therefore, it needs many serially executed write units to finish a cache line service. As shown in Fig. 1, the constant in each write unit refers to the power requirement in the worst case. Write service of a cache line completes at T_{10} 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

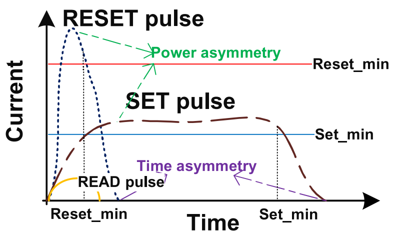


Fig. 2. Illustrations of PCM read and write operations.

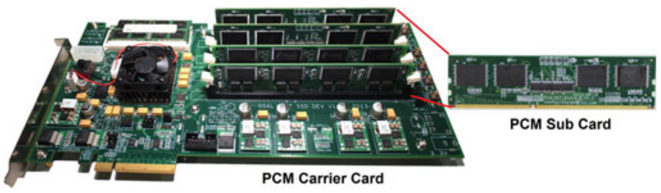


Fig. 3. The PCM hardware prototype.

line service under the conventional write scheme as Equation (1)

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

$T_{conventional} = 8T_{set}$ as M = 64 and N = 8 in our example. In summary, serially executed write units are the primary cause of the poor write performance and the key to address the problem is minimizing the number of write units.

2.2 FNW, 2-Stage-Write and Three-Stage-Write

Flip-N-Write (FNW) [17] 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 have 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 Fig. 1, all write units are finished at T_6. 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 [16] leverages the time and power asymmetries 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 at 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”, the 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 previous art [31] and our experimental results on our real hardware prototype [32] as shown in Fig. 3, the time ratio between writing “0” and “1” is about 3X (far below 8X). As shown in Fig. 1, when the time ratio of writing “0” and “1” is 3, the “0” stage needs almost 3 write units ($\lfloor 8/3 \rfloor$) to finish writing all “0” and the cache line service is finished at T_7. Assuming writing “0” is K times faster than writing “1” with L power needs, we can conclude the average service time as Equation (3). $T_{2\text{-}Stage\text{-}Write} = 4.6T_{set}$ when L = 2, K = 3, M = 64 and N = 8. Another thing to note is that 2-Stage-Write brings no advantages to bit-write reduction, which has potential benefits on endurance and energy improvement

$$T_{2\text{-}Stage\text{-}Write} = \frac{M}{K*N} T_{set} + \frac{M}{2*N*L} T_{set}. \tag{3}$$

Recently Three-Stage-Write [33] tries to fix the long stage-0 problem by adding another read stage before stage-0 and stage-1. In general, it combines the merits of the Flip-N-Write and 2-Stage-Write, and the write service time is concluded in Equation (4). Under the same parameters, the write service is finished at T_5 and $T_{Three\text{-}Stage\text{-}Write} = 3.6\ T_{set}$

$$T_{Three\text{-}Stage\text{-}Write} = \frac{M}{2*K*N} T_{set} + \frac{M}{2*N*L} T_{set}. \tag{4}$$

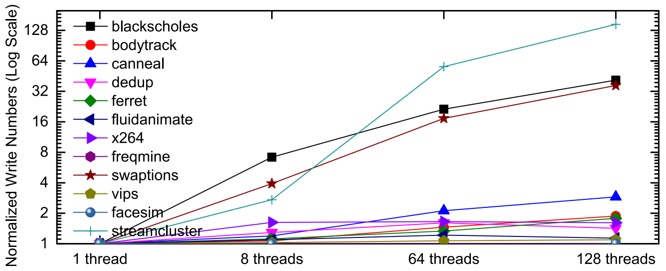


Fig. 5. Normalized PCM write times with the increasing number of threads.

2.3 Important Insights

Nowadays, more and more applications use multi-threaded programming. It becomes increasingly more common that highly parallel applications run with hundreds of threads for taking full advantage of the abundant physical resources in a data center and the scale is still on the rise with the developing of CMPS and CUDA [34], [35]. Multi-threaded applications that exhibit evident access locality and some typical data patterns occupy a large fraction of memory data accesses [36], [37], [38]. Multi-threaded PARSEC 2.0 experiment results are shown in Figs. 4 and 5, in which “0” or “X” refers to the value of one byte (8 bits). “0” presents that the byte value is zero while “X” means any value type. $Type1$,

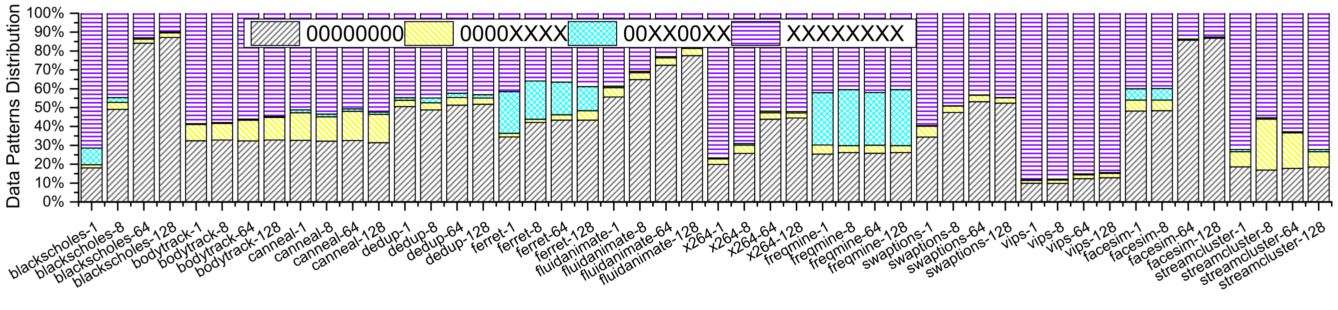


Fig. 4. Data pattern distribution with the increasing number of threads.

TABLE 1
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 bytes	00000000	0	0 bit
$Type2$	01	4 zero bytes (i)	0000XXXX	XXXX	32 bits
$Type3$	10	4 zero bytes (ii)	00XX00XX	XXXX	32 bits
$Type4$	11	Uncompressed	XXXXXXXX	-	Original(64 bits)

$Type2$, $Type3$ and $Type4$ compose all data of memory write that each data belong to one of the defined data types. We obtain two useful insights from the experimental results:

Key Insight 1. We observe that data patterns 00000000, 00XX00XX and 0000XXXX, with each letter representing one byte,[1] occupy 10 percent (vips) up to more than 80 percent (facesim) of the memory accesses with different workloads and number of threads, as shown in Fig. 4. We define these typical data values as *zero-extended values* [36]. Zero-extended values occupy more than 40 percent of all memory accesses on average and up to more than 60 percent in some benchmarks (blackscholes, fluidanimate and facesim). However, in some typical benchmarks, there are not many zero-extended values (vips and streamcluster). Another finding is that the value property becomes more obvious with the increasing number of threads. For example, there may be not that many zero-extended values in some workloads when the number of thread is 1 but the values occupy a dominant position with the increasing number of threads, especially in blackscholes, fluidanimate and facesim.

Key Insight 2. The number of data write increases with the increasing number of threads, which results in a negative influence on PCM lifespan. As shown in Fig. 5, the number of data write in streamcluter increases 3X, 56X and more than 100X when the number of threads is 8, 64 and 128, respectively. In comparison, the results of freqmine, vips and facesim are not significant (within 10 percent increasing when the number of threads is 128). On average, the number of data write increases 1.6X, 7.8X and 18X with 8-threaded, 64-threaded and 128-threaded, respectively.

Previous work points out that there exists frequent-value locally in the workload execution. Within any period of application execution, some typical data values, named frequent data values, may occupy most of the memory data accesses [39], [40]. All these data patterns may result from data structure alignment or word-alignment [36]. On one hand, many small values are 4, 8, 16, 32 bits, which 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, which is called data structure padding [37]. On the other hand, if we store two small numbers (XX) into two words, we will also get two leading zero bytes in each word due to the word-alignment. Moreover, zero-extended pattern goes up when the number of threads is increased for some workloads. We are not sure about the specific reasons and it may be due to the communication between the threads. It is demonstrated that SPECint95 and SPECint2000 benchmarks exhibit more than 40 percent zero values in memory accesses on average [38]. 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.

Moreover, splitting a program into multiple threads leads to write amplification. In general, the write amplification is less than 3X when the number of threads grows to 128X compared with the baseline. This phenomenon may be due to the interaction between threads, which needs to store necessary information. However, some workloads show large write amplification (100X for 128 threads). Part of the reason for this situation is that some workloads are not memory-intensive and the number of write requests is small. The increase in the number of threads leads to an increase in write requests and the write amplification is very huge since the base is quite small.

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.

2.4 Min-WU

Minimize the number of Write Units (Min-WU) 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 amount 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 encoding 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 1, each "0" or "X" refers to the value of one byte (8 bits), when data type is 00000000, 0000XXXX or 00XX00XX, the data amount is reduced after sFPC with prefix bits indicating the zeroes in data values. For example, if the data value is 0000XXXX, data are compressed to XXXX after sFPC and the residual bytes 0000 can be replaced 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 Fig. 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.

1. To avoid duplication of data patterns, we examine pattern 00000000, 00XX00XX, 0000XXXX and XXXXXXXX in turn, i.e., one data belongs to only one data pattern.

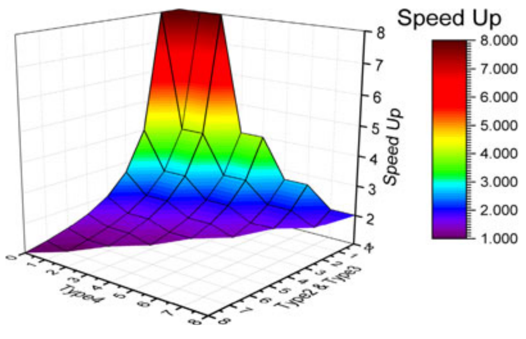


Fig. 6. Speed Up of Min-WU compared with the baseline.

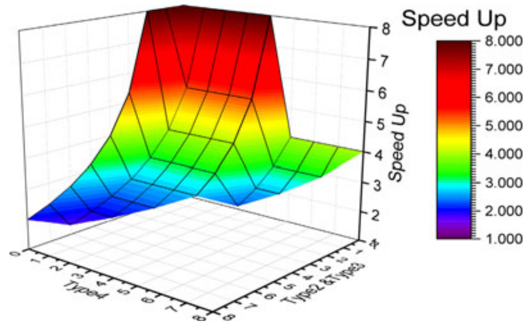


Fig. 7. Speed Up of Min-WU-PF compared with the baseline.

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 and WU6 in Fig. 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 because the power requirement is halved after sFPC ($8 + 8 <= 16$). When the prefix bits of a write unit are "00", such as WU0, WU2, WU5 and WU7, there is no need to do the write since the data can be indicated with the prefix bits. In our example, all write units can be finished at T_4. N_{type1}, N_{type2}, N_{type3} and N_{type4} are the numbers of various data types shown in Table 1, respectively. The average service time of a cache line is concluded in Equation (5). $T_{Min\text{-}WU} = 3T_{set}$ in our example. However, the performance degrades strikingly if the write data are $Type4$-dominant,

$$T_{Min\text{-}WU} = \left(\frac{N_{type2} + N_{type3}}{2} + N_{type4}\right)T_{set}. \tag{5}$$

We define the write improvement over conventional write scheme as $SpeedUp$, which is shown in Equation (6).

$$T_{SpeedUp} = \frac{T_{Conventional}}{T}, \tag{6}$$

Fig. 6 shows the speedup of Min-WU compared with the baseline without any write optimization. X-axle presents the number of data units whose patterns are $Type2$ and $Type3$ and Y-axle denotes the number of data units whose types are $Type4$. When there are only $Type1$-data, only one write unit is needed for writing the prefix bits of all data units, i.e., the maximum speedup over conventional write scheme is 8. In the worst case, all data units are $Type4$ and there is no improvement over the conventional scheme.

2.5 Min-WU-PF

To solve the performance degradation of Min-WU when write data are $Type4$-dominant, we propose a variation of Min-WU called Min-WU-PF (data Partly Flip). Min-WU-PF flips the data when data are not $Type1$ and more than half of bits should be changed. Thus, we also double the write unit size and can get more parallelism improvement even data are $Type4$-dominant. As shown in Fig. 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 similar to FNW scheme ($8 + 8 \leq 16$). Likewise, WU1 and WU3 can be processed in parallel ($4 + 4 < 16$). Min-WU-PF further enhances the write parallelism and reduces the number of write execution. As shown in Fig. 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 (7),

$$T_{MWP} = T_{read} + \left(\frac{N_{type2} + N_{type3}}{4} + \frac{N_{type4}}{2}\right)T_{set}. \tag{7}$$

Fig. 7 shows the speedup of Min-WU-PF over the baseline. In most cases, Min-WU-PF outperforms conventional more than 3X and the worst case occupies only a small part of all cases (purple indicates areas of none performance improvement compared with the conventional).

In summary, although FNW and 2-Stage-Write can reduce the write latency, they do not focus on reducing the number of write units by leveraging the special data patterns. As concluded in Table 2, FNW focuses on the differences of new data and stored data while 2-Stage-Write

TABLE 2
Differences of Various Write Schemes

Scheme	Key Idea	Reduce Latency & Energy	Overhead
FNW	Difference between the written data and stored data	YES & YES	Extra read and inversion processes and circuits for individual write support
2-Stage-Write	Asymmetry of writing zero and one	YES & NO	Extra counter for bits inversion and circuits for 2-Stage-Write support
Three-Stage-Write	2-Stage-Write with bit-flip	YES & YES	Extra read and inversion processes and circuits for Three-Stage-Write support
Min-WU & Min-WU-PF	Minimize the number of write units	YES & YES	Extra encode and decode processes and circuits for parallel write support

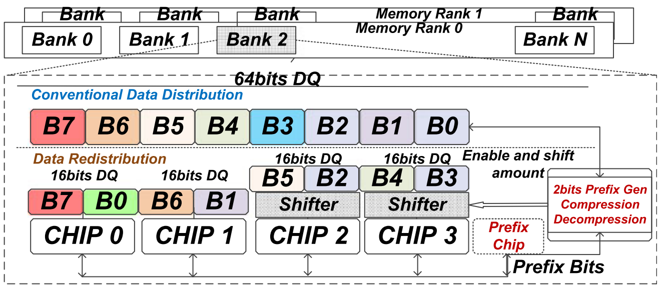


Fig. 8. Hardware architecture.

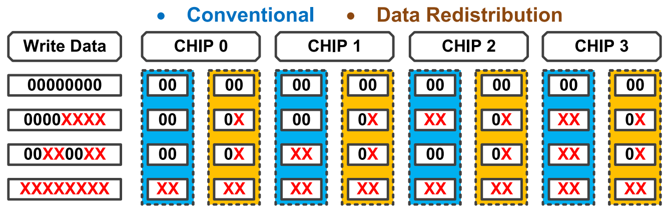


Fig. 9. Data redistribution.

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 are $Type4$-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.

3 Hardware Implementation

3.1 Hardware Architecture

To support the write parallelism proposed in Min-WU/Min-WU-PF, we carry out carefully designed hardware architecture. We adopt data redistribution methods and add some critical components to support proposed designs. To implement data and prefix data separation, a new chip, i.e., prefix chip is adopted to exploit chip-level parallelism. As only 2 prefix bits are needed for 64 data bits, only 16 prefix bits are needed for 64B data. As the size of write unit per chip in our design is 16b, prefix bits of WU0 to WU7 can be written in one write unit. With chip-level parallelism improvement, prefix bits are processed with data bits, as shown in Fig. 1.

Our hardware architecture includes three new components, i.e., *Prefix Generate Logic*, *compression* and *decompression*, compared with traditional memory architecture. Prefix Generate Logic (PGL) is a low overhead prefix bits generation circuit. PGL can be released to a simple combinational logic circuit and automatically creates the 2-bits prefix based on the data value. The prefix can be generated quickly using multiplexer and adder circuits. Data compression and decompression logic are used for encoding and decoding data with proposed sFPC scheme as shown in Table 1. After the data compression, the total amount of data can be significantly reduced, which provides great supports for Min-WU and Min-WU-PF write schemes. However, the compression and decomposition processes will deliver extra overhead when writing and reading the data. The overhead and details of the compression/decompression’s designs will be present in the following section. To reduce the overall write latency and improve the space efficiency, we separate the prefix bits and data bits into different chips, i.e., an individual chip named prefix chip is used to store the prefix bits of the written data. As illustrated in Fig. 8, the data-prefix-separation design can effectively improve chip parallelism without losing the accuracy and reduce the overall space overhead of sFPC’s implementation.

3.2 Data Redistribution

The bit-writes of each chip may vary much when using Min-WU/Min-WU-PF under traditional system architecture as shown in Fig. 8. The data amount of each chip is quite different since Min-WU only writes “X” and “0” can be indicated by the prefix bits. Chip 0, Chip 1 and Chip 2 have to wait for the completion of the heaviest bit-writes Chip 3 (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 Fig. 8. In conventional memory architecture, chip (i) write Byte (7-2·i) and Byte (6-2·i). As we mentioned, the data amount may vary much in each chip. In our design, while the data display three main patterns, we change the data distribution slightly. In details, Chip 0 is mapped with Byte 7 and Byte 0 and Chip 1 corresponds Byte 6 and Byte 1. The data distribution of Chip 2 and Chip 3 is decided by the data type. If data type is $Type2$, i.e., 0000XXXX, Chip 2 corresponds to Byte 5 and Byte 2 while Chip 3 corresponds to Byte 4 and Byte 3. If data type is $Type3$, i.e., 00XX00XX, Chip 2 corresponds to Byte 2 and Byte 5 while Chip 3 corresponds to Byte 3 and Byte 4. This can be released easily by adopting two individual shifters with enable signals. As shown in Fig. 9, 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 are.

3.3 Circuits Designs

In this section, we will first introduce the circuits’ designs of data compression/decomposition. Then, the design details of data path, write control logic as well as write driver of Min-WU and Min-WU-PF are also provided. Finally, we also analyze the space and time overhead of proposed designs.

3.3.1 Prefix Generation and Data (De)Compression

To meet the goals of proposed sFPC as illustrated in Table 1, we need low-delay prefix generation, compression and decompression circuits to reduce the impact on critical read latency of the main memory. The RTL schematics of compression and decompression are shown in Figs. 10 and 11, respectively. The prefix generation is combined with the data compression. Different data patterns have various write (compression) and read (decompression) overhead. The time overhead is concluded in Table 3.

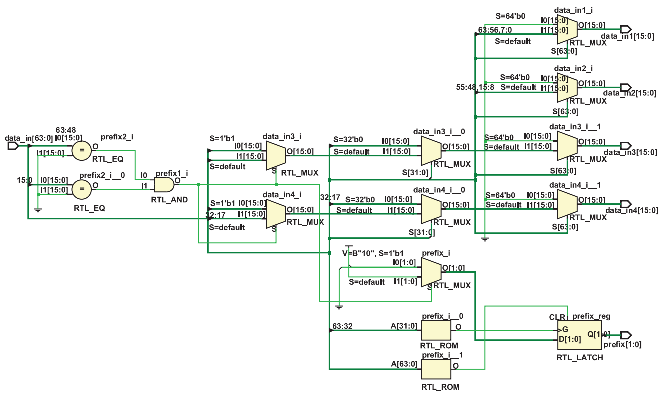


Fig. 10. Data compression RTL schematic (WRITE).

In general, if the written data pattern is “00”, only 2 cycles are consumed to finish the data compression. As shown in Fig. 10, only a comparator and a multiplexer are needed. In the same way, if the data pattern is “01”, the written data must pass a comparator and two multiplexers, i.e., 3 cycles are needed to accomplish the data compression process. In comparison, when written data present patterns “10” and “11”, the data compression is more complicated compared with the situation of “00” and “01”. In brief, one comparator, one AND gate and three multiplexers are needed. As a result, 5 cycles are consumed to encode the data. Data decompression of sFPC is much more complicated compared with compression as illustrated in Fig. 11. If the written data pattern is “00”, 5 cycles are necessary for data decoding. In short, one comparator, three multiplexers and one data latch are implemented. When the data pattern is “01”, four extra cycles are needed to decode the data. In summary, one comparator, two multiplexers as well as one data latch are combined when prefix bits are “01”. The situation when prefix bits are “10” is similar to the case of “00”, i.e., extra 5 cycles are consumed when performing data decompression. The case of “11” has the lowest overhead compared with other cases. Only one comparator, one multiplexer as well as one data latch are required to perform data processing and only 3 cycles are consumed.

3.3.2 Min-WU

To provide write parallelism supports, we carry out a fine-tuned hardware circuit named fWP based on an industrial prototype from Samsung [26]. In addition, we will introduce the details of our circuit design including the data path, write control logic as well as write driver of Min-WU and Min-WU-PF, respectively.

The red part of Fig. 12 shows the overall data path of Min-WU. Compared with the design of FNW, we add an individual write logic layer named Min-WU write logic. The data path consists of 1-word synchronous burst domain and 8-word prefetch domain. To meet the design goals of Min-WU write scheme, we expand the size of write buffer but not the size of the array. The on-chip write buffer stores 128 bits data and 16 prefix bits of these 8 write units (128/16 = 8). The prefix bits will be sent to the write logic and write driver for write units reordering. The middle may deliver extra overhead when writing the data. But above all, the middle layer won’t deliver any overhead on the critical read path, which is the key performance bottleneck of the system.

The red part of Fig. 13 shows how the write control logic works. The primary goal of Min-WU is to improve the write concurrent under the power constraints. Accordingly, the primary purpose of the write control logic is to choose which data units should be written at one time. Shared Finite State Machine (FSM) continuously 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 since 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. It’s worth noting that the units choosing of “01” or “10” can be done when other data are written and the overhead of units choosing can be hidden by the long write time.

The red part of Fig. 14 shows the write driver of Min-WU. 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 input value of the AND gate is decided by the prefix

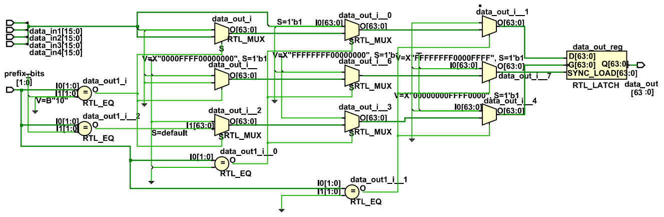


Fig. 11. Data decompression RTL schematic (READ).

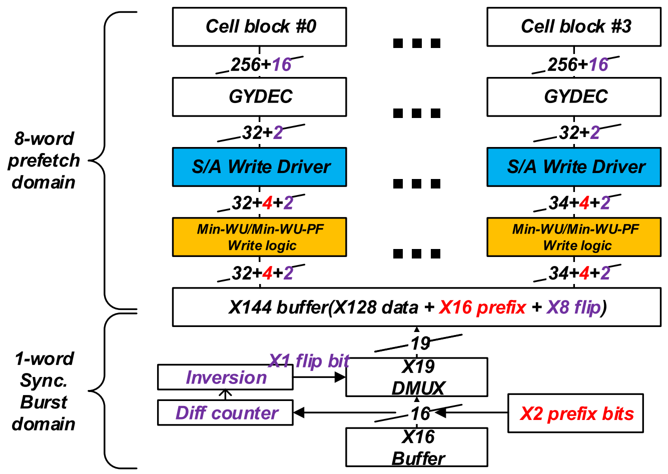


Fig. 12. Datapath of Min-WU and Min-WU-PF.

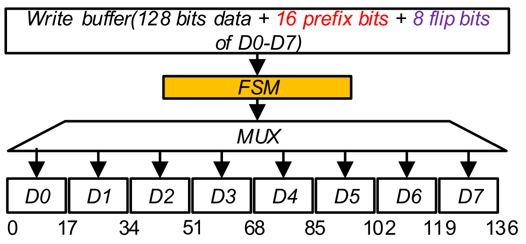

Write buffer(128 bits data + 16 prefix bits + 8 flip bits of D0-D7)
FSM
MUX
D0
D1
D2
D3
D4
D5
D6
D7
0
17
34
51
68
85
102
119
136

Fig. 13. Write control logic of Min-WU and Min-WU-PF.

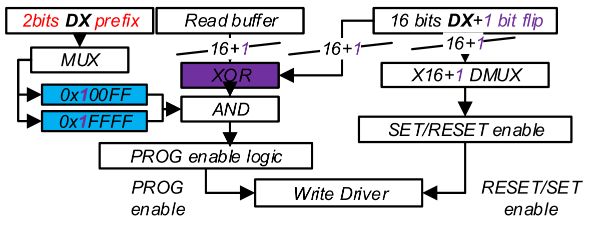


Fig. 14. Write driver of Min-WU and Min-WU-PF.

bits. When the prefix bits are “01” or “10”, *0x00FF* is selected for only writing “X” of data “0X”. Otherwise, *0xFFFF* is selected for writing “XX” value. Thus, when data patterns are “01” and “10”, only half of the total bits will be written, i.e., the size of the write unit is doubled under this situation.

3.3.3 Min-WU-PF

The red and purple parts of Fig. 12 shows the overall data path of Min-WU-PF. Compared with the design of Min-WU, the overall length of data path is equal, i.e., no overhead is added to the critical read path, which is the key performance bottleneck of the system. Min-WU-PF write scheme flips the residual data if more than half of bits need to be changed after sFPC. Furthermore, we expand the size of write buffer again and add 8 flip bits for 8 write units. Therefore, the on-chip write buffer stores 128 bits data, 16 prefix bits and 8 flip bits. In addition, flip bits are stored in the PCM array together with the data and word lines are extended from 32 bits to 34 bits. The red and purple parts of Fig. 13 shows the write control logic of Min-WU-PF. After flip operations on partial data, the bits need to be written is no more than half of the total bits. Thus, two data units that the prefix bits are “11” can be written concurrently under the power constraints, i.e., the size of write units is doubled when the data are $Type4$. In the same way, four data units that prefix bits are “01” or “10” can be finished in parallel since sFPC and data partly flip quadruple the size of write unit. Simply, Shared Finite State Machine (FSM) first chooses two data units whose prefix bits are “11” because they take the whole power budget in the worst case. Then the FSM chooses four DX whose prefix bits are “01” or “10” when previous data are in processing. As extra flip bits are adopted to reduce the amount of written data, the offset of units is also changed, as shown in Fig. 13. The red and purple parts Fig. 14 shows the write driver of Min-WU-PF. Especially, a read buffer is added for data comparison. Min-WU-PF first reads the new data and flips it if more than half of bits need to be changed after data compression. Min-WU-PF sends PROG enable only to the cells that need to be changed, which can be easily realized with a low overhead

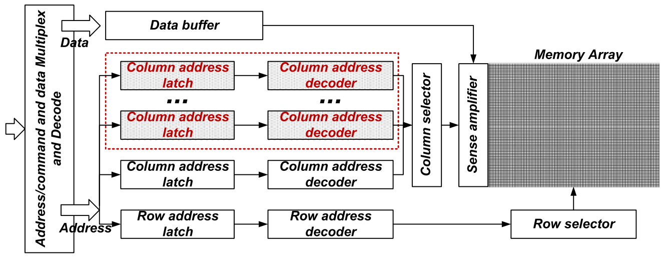


Fig. 15. Modification to PCM chip composition.

TABLE 3
Time Overhead

Data type	Prefix Bits	Write (cycles)	Read (cycles)
$Type1$	00	2	5
$Type2$	01	3	4
$Type3$	10	5	5
$Type4$	11	5	3

XOR gate. After that, the results are combined with *0x00FF* and *0xFFFF* with an AND gate similar to Min-WU.

3.3.4 Modification to PCM Chip

Min-WU and Min-WU-PF operate multiple write units one time to improve the write parallelism under the power constraint. However, in the standard design, each cell block has only one column decoder with the specific column address, which makes it a great challenge to release our proposed fWP scheme. In order to support concurrent writes in multiple columns, we adopted previous art and modified the PCM chip composition [41], [42], [43]. As shown in the dashed portion of Fig. 15, our design adopts multiple column address latches and decoders to select different columns and operate them in parallel. The number of column address latches and decoders is equal to the number of write units under conventional write scheme since we only target the data within one cache line write service. In addition, we did not make any changes to the row address latch and decoder.

3.4 Overhead

3.4.1 Time Overhead

The implementation of Min-WU and Min-WU-PF delivers extra time overhead. The time overhead is mainly caused by the prefix generation, data compression and decompression. According to our results performed in our real hardware prototype DSAL-SSD [32], [44], only several cycles are needed to generate the prefix. The low overhead prefix generation scheme won’t deliver any negative influence on the service time of a cache line. As shown in Table 3, different data patterns may deliver various extra compression or decompression delays. Compared with hundreds of write cycles (153 ns) and tens of read cycles (53 ns), the time overhead is relatively slight and acceptable. Moreover, the selection of data units may also cause time overhead. In order to reduce the time overhead, Min-WU and Min-WU-PF adopt analyze-under-write scheme, i.e., we only choose the units that need to be written at the beginning. The units need to be executed next will be selected under the process of previous write units. Since it takes hundreds of cycles per write unit, the selection time of units can be hidden.

3.4.2 Space Overhead

A simple schematic diagram of space overhead of FNW, naive design (simple sFPC and FNW combination), Min-WU and Min-WU-PF schemes is shown in Fig. 16. Considering an example of two 4-bits PCM chips, the old data are “0111” and “1110” while the new data are “0000” and “0000”. We will introduce and analyze the space overhead of FNW, naive design, Min-WU and Min-WU-PF

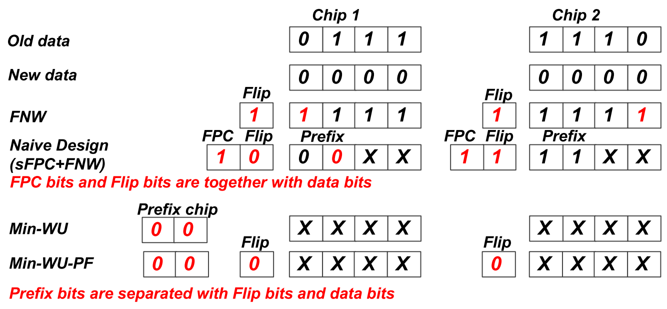


Fig. 16. Illustration of FNW, Naive Design, Min-WU and Min-WU-PF methods.

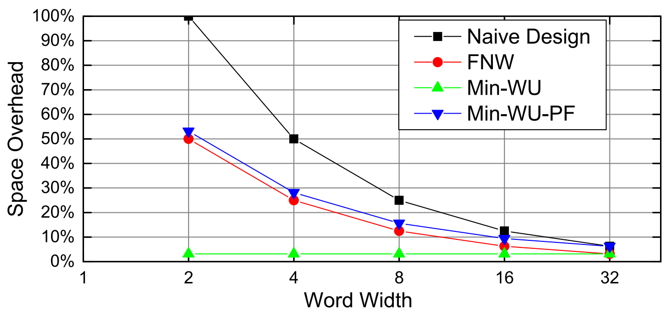


Fig. 17. Space overhead.

individually between different chip's word width. We assume the data width is 64-bits and *SpaceOverhead* is referred to the number of extra bits divided by the data width (64-bits size in this study), as shown in Equation (8).

$$SpaceOverhead = \frac{Extra - bits}{datawidth} \tag{8}$$

FNW. In our example, the old data and the new all differ 3 bits both in chip 1 and chip 2. Because FNW flips the data to be written if more than half of bits have to be changed, both chips (chip 1 and chip 2) have to do the data inversion ($3 > 4/2$ in chip 1 and $3 > 4/2$ in chip 2). Thus, the flip bit will be set to "1" and the data will be flipped to "1111" in both chips. The space overhead, i.e., the extra array area overhead of FNW can be concluded as Equation (9), where N presents the word width of the memory chip. The overhead is decreased with the word width increases as shown in Fig. 17

$$SpaceOverhead_{FNW} = 1/N, \tag{9}$$

Naive Design. As we use sFPC to reduce the total amount of data, the design can also be combined with data inversion to further reduce the data amount. The native process of sFPC and data inversion combination is to compress the data with sFPC first, and if the residual data still have more than half bits diff, the data shall be flipped with an extra bit indexing it. In our example, the new data of chip 1 and chip 2 will be compressed to 2-bits prefix "00", i.e., we only need to write "00" rather than all new data. The FPC index will be set to "1" in both chips and the prefix bits are together with the data bits. In chip 2, the written data "00", i.e., the prefix bits have 2 different bits compared with old data and it will be flipped to "11" with "Flip" bit indexing it. The space overhead is shown in Equation (10)

$$SpaceOverhead_{Naive} = 1/N(Flip) + 1/N(sFPC). \tag{10}$$

Min-WU. Unlike FNW and naive designs, Min-WU uses an extra chip, i.e., prefix chip to store the prefix bits, as shown in Fig. 8. No matter what the new data are, the space overhead is limited to 2 bits, i.e., the size of prefix bits. In our example, the data are compressed by the prefix bits "00" and there is no bit-write in chip 1 and chip 2. The space overhead of Min-WU is concluded in Equation (11)

$$SpaceOverhead_{Min\text{-}WU} = 2. \tag{11}$$

Min-WU-PF. Based on the designs of Min-WU and FNW, Min-WU-PF also introduces an extra one bit to index whether the data are flipped or not. In our example, the data are compressed with 2 prefix bits indexing the data. The overhead of sFPC is constant (the size of prefix bits) while the overhead of data inversion is related to the word width. Likewise, the extra array area overhead of Min-WU-PF can be concluded as Equation (12)

$$SpaceOverhead_{Min\text{-}WU\text{-}PF} = 1/N + 2. \tag{12}$$

3.4.3 Area and Power Overhead

As Min-WU and Min-WU-PF change the chip-level circuits design, such as introducing multiple column latches and decoders, individual write logic layer and modifying write driver logic, the area and power consumption may increase due to the added circuits. In details, Min-WU uses the FSM to release the units selection and Min-WU-PF introduces extra data inversion process to reduce the amount of data compared with Min-WU. However, write is not on the critical performance path while read latency is quite important for the main memory system. Min-WU and Min-WU-PF only extend the datapath of write and the critical read path is the same compared with the design of baseline. In previous art [41], [42], [43], the area overhead caused by the added column latches and decoders is less than 0.05 percent and the power overhead is less than 0.5 percent. In addition, the workload of added circuits is light and the overhead is minimal. For example, FSM needs to deal with the selection of only 8 units, which is not an area-hungered component. The modification on write driver is slight, only some sample XOR gates and AND gates. Moreover, the added logics, such as FSM or data inversion, are much less complicated compared with some critical cost-sensitive or area-hungry components, such as the program-and-verification circuits inside the PCM chip [16]. Similarly, the power consumption of added circuits is hence small compared with the baseline.

4 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, write latency, IPC, applications running time, bit-write reduction as well as energy consumption.

Specifically, we first present the parameters and the experimental environment. We implemented our Min-WU and Min-WU-PF on the event-driven GEM5 simulator [45], [46] to evaluate our design and the simulation parameters are shown in Table 4. The GEM5 simulator is an open

TABLE 4
Parameters of Simulation

Parameter	Value
CPU	4-Core 2 GHz ALPHA O3 processor
L1 Cache	32 KB I-cache, 32 KB D-cache, 2 cycles latency
L2 Cache	8-way, 2 MB, 64B cache line, 20 cycles latency
L3 Cache	16-way, 8 MB cache, 64B cache line, 50 cycles latency
Memory Controller	FRFCFS scheduling algorithm, 32-entry R/W queues
Memory Organization	4 GB SLC PCM, 64 bits data width, 2 ranks, 8 banks
PCM Organization	4 chips per bank, 8 bytes write unit size
Read, Reset and Set time	50 ns, 53 ns and 153 ns

source modular platform for computer system architecture research. In this paper, we use 4-core ALPHA-like CMP system with 2GHz frequency. We also simulate the whole memory hierarchy and three-level cache (L1, L2 and last level L3 cache) is adopted. All caches have 64B cache line size. In addition, a faithfully main memory controller and 4 GB PCM memory are also implemented in our simulation. The PCM main memory has 2 ranks and 8 banks. The main memory controller has individual read and write queues and it uses FRFCFS (first ready first come first served) scheduling algorithm that schedules reads requests first and only dealing write requests when the write queue is full. The parameters of PCM are taken from past work [17], [31], the prototype of Samsung published in [26] and the results from our DSAL-SSD hardware prototype with actual PCM chips provided by Micron [32]. Partly energy parameters are taken from CACTI [47] and [26], [31]. The details of benchmarks are concluded in Table 5. All benchmarks are from different areas, including video processing, financial analytics, physics simulation, picture processing etc. We compare Min-WU/Min-WU-PF with state-of-the-art FNW with all 12 benchmarks under PARSEC 2.0 [48] without selectively choosing. In addition, all instructions are simulated before the application exits for each PARSEC workload. Our goal is to find the most suitable application scenarios of Min-WU and Min-WU-PF. In general, we use the conventional PCM write scheme without any optimization as the baseline.

TABLE 5
PARSEC 2.0 Benchmarks

Benchmark	Introduction	RKPI	WPKI
blackscholes	Option pricing with Black-Scholes Partial Differential Equation (PDE)	0.04	0.02
fluidanimate	Fluid dynamics for animation purposes with Smoothed Particle Hydrodynamics (SPH) method	0.59	0.32
bodytrack	Body tracking of a person	0.72	0.24
freqmine	Frequent itemset mining	0.62	0.25
swaptions	Pricing of a portfolio of swaptions	0.04	0.02
canneal	Simulated cache-aware annealing to optimize routing cost of a chip design	2.76	0.19
dedup	Next-generation compression with data deduplication	0.82	0.49
streamcluster	Online clustering of an input stream	10.42	6.53
facesim	Simulates the motions of a human face	0.43	0.37
vips	Image processing	2.56	1.56
ferret	Content similarity search server	1.67	0.95
x264	H.264 video encoding	1.01	0.23

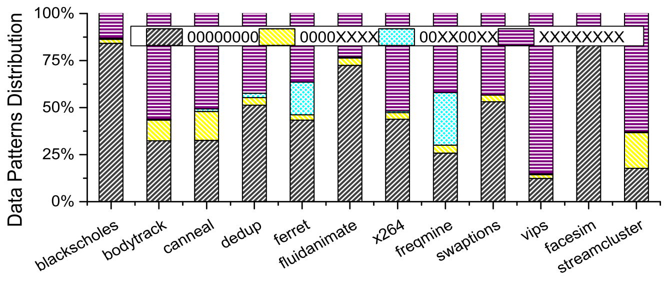


Fig. 18. Data patterns of 12 PARSEC 64-threaded workloads.

4.1 Data Pattern Distribution

We first measure all data pattern distribution of the PARSEC 2.0 benchmarks to verify the motivations. We use 64 threads per program and the results are shown in Fig. 18. We observe that zero-extended values dominate the write values in all benchmarks and occupy more than 40 percent of all memory accesses on average. Three programs (blackscholes, fluidanmate and facesim) show more than 70 percent zero-extended values while the least one has more than 20 percent (vips). It proves that it is important to utilize these commonplace zero-extended values for write performance improvement and energy reduction.

4.2 Performance

4.2.1 Read Latency

Read latency is crucial for the main memory system performance and it is the bottleneck of the whole system performance. Fig. 19 shows the read latency reduction of Min-WU/Min-WU-PF, FNW, and Three-Stage-Write compared with the baseline. Overall, Min-WU significantly outperforms FNW in some benchmarks while being equivalent in the others except vips. The reason is the write data are *Type4*-dominant in vips. Min-WU-PF outperforms FNW in all benchmarks. FNW can get 29-43 percent read latency reduction compared with the baseline while Min-WU can

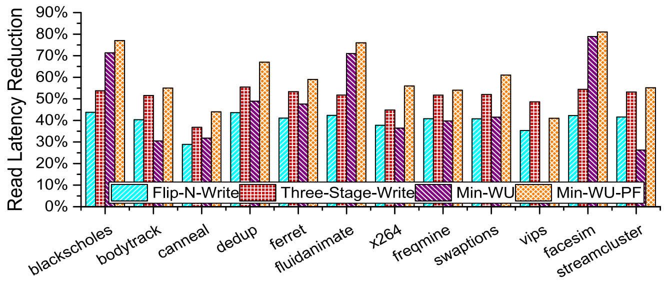


Fig. 19. Read latency reduction.

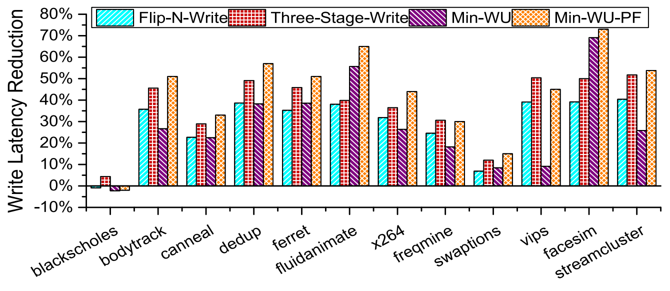


Fig. 20. Write latency reduction.

get 32-79 percent read latency improvement. Min-WU shows 6 percent more latency reduction considering the low performance of vips. Min-WU-PF shows 61 percent read performance improvement compared with baseline and outperforms state-of-the-art FNW and Three-Stage-Write by 21 and 11 percent on average, respectively. We use 64 threads in the experiment and we can get more read latency improvement with the number of thread increases.

4.2.2 *Write Latency*

Min-WU and Min-WU-PF can significantly reduce the total time of a cache line service, so the write requests can be finished more quickly compared with the conventional write scheme. Experimental results of write latency are shown in Fig. 20. We notice that Min-WU outperforms FNW greatly in some benchmarks but fall behind in some workloads similar to the results of read latency. We also observe that Min-WU and Min-WU-PF show performance degradation compared with FNW (typically in blackscholes). Many reasons may cause this problem. First, FRFCFS (first ready first come first served) scheduling algorithm schedules read requests first and processes write requests when the write queue is full. On the other hand, Min-WU and Min-WU-PF introduce extra overhead when reading and writing data. Besides, the blackscholes is a read-dominant workload and write requests are very few in number, the write latency is sensitive and particularly vulnerable to impacts. In summary, Min-WU shows 28 percent write latency improvement considering all workloads on average compared with the baseline. Moreover, Min-WU-PF outperforms the state-of-the-art FNW and Three-Stage-Write, and decreases 15 and 7 percent overall write latency, respectively.

4.2.3 *IPC*

IPC (Instructions per cycle, i.e., the average number of instructions executed for each clock cycle) is one of the most important indicators of the processor and system's performance. Highly efficient main memory system can improve the computer speed of the application benchmarks. The results of IPC improvement are illustrated in Fig. 21. In summary, Min-WU can gain 32 percent IPC improvement compared with the baseline while Min-WU-PF shows 44 percent IPC increment. Moreover, Min-WU-PF respectively get 12 and 8 percent more IPC improvement compared with the FNW and Three-Stage-Write, respectively.

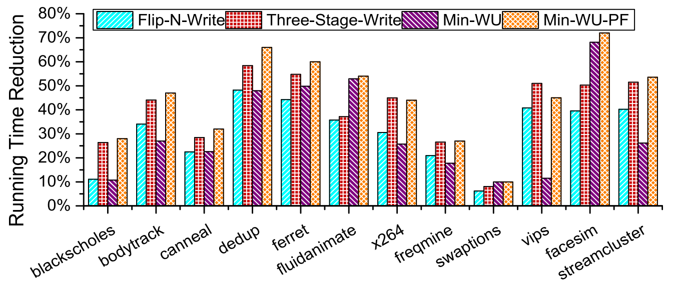


Fig. 22. Running time reduction.

4.2.4 *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 Fig. 22. 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 percent running time reduction against the baseline on average, respectively. Moreover, Min-WU-PF outperforms FNW and Three-Stage-Write by 14 and 12 percent on average, respectively.

4.3 Endurance and Energy

4.3.1 *Write-Bits Reduction*

Min-WU and Min-WU-PF decrease the data amount by leveraging sFPC as illustrated in Table 1, i.e., the write-bits can be significantly reduced after compression. The write-bits reduction benefits both the lifespan of PCM and overall system power consumption. Experimental results of 12 PARSEC benchmarks are shown in Fig. 23. On average, the number of bits needs to be written with FNW is only 60 percent compared with the baseline and it is the same with Three-Stage-Write because they use the same data dealing process. In comparison, Min-WU outperforms 13 percent compared with FNW while Min-WU-PF reduces 16 percent data amount with data partly flip. Min-WU-PF introduces extra bits, i.e., flip bits, compared with Min-WU, so in some workloads, Min-WU may show better write-bits reduction (e.g., vips and streamcluster).

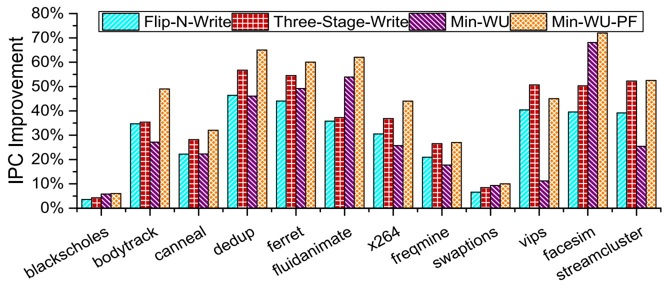


Fig. 21. IPC improvement.

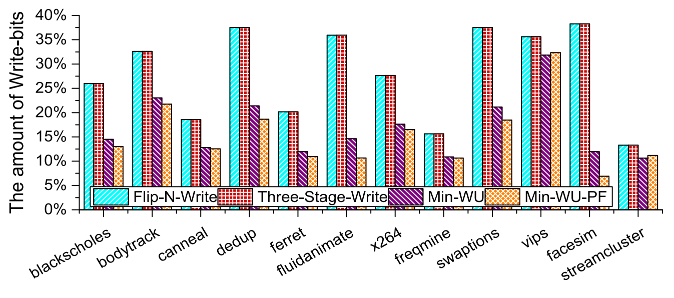


Fig. 23. The amount of write-bits.

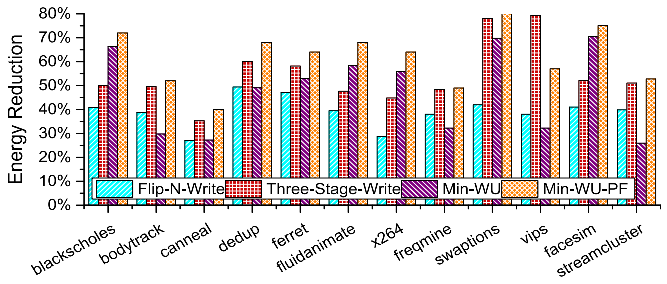


Fig. 24. Energy improvement.

TABLE 6
SPEC 2006 Benchmarks

Benchmark	Introduction	RKPI	WPKI
bwaves	4 copies of bwaves	11.19	5.67
sjeng	4 copies of sjeng	0.21	0.18
gobmk	4 copies of gobmk	0.14	0.13
zeusmp	4 copies of zeusmp	13.56	3.18
bzip2	4 copies of bzip2	8.81	4.64
astar	4 copies of astar	0.53	0.38
libquantum	4 copies of libquantum	10.95	6.93
leslie3d	4 copies of leslie3d	7.11	2.59

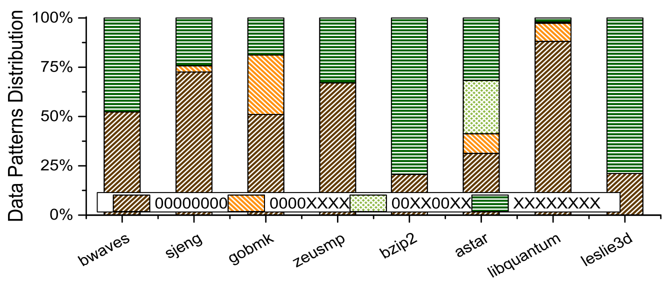


Fig. 25. Data patterns of 8 SPEC 2006 workloads.

4.3.2 *Energy Improvement*

Energy consumption is an important problem in current data centers. High-energy consumption leads to serious heat problems and numerous refrigerating devices are deployed to cool down the data center. Energy improvement can bring significant benefits to both the environment and economy. As shown in Fig. 24, 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 percent in five workloads. Min-WU gains 46 percent less energy compared with the baseline and outperforms FNW by 11 percent on average. Min-WU-PF reduces 62 percent energy consumption compared with the baseline and respectively outperforms FNW and Three-Stage-Write by more than 22 and 18 percent on average.

4.4 Design Space Exploration

In order to prove the efficiency and effectiveness of our proposed Min-WU and Min-WU-PF schemes under different

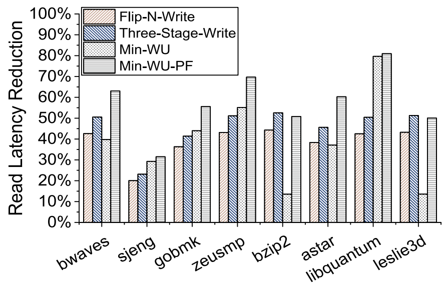


Fig. 26. Read latency reduction.

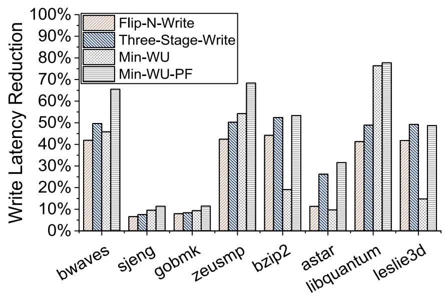


Fig. 27. Write latency reduction.

memory configurations and workloads, we had redone the experiment with more memory intensive benchmarks (8 SPEC 2006 workloads [49]) and larger L3 cache capacity (8 MB per core, 32 MB in total). Detailed benchmarks information is concluded in Table 6. In addition, one billion instructions are simulated for each SPEC 2006 workload after fast-forwarding one billion instructions.

Data pattern distribution of 8 SPEC 2006 benchmarks is shown in Fig. 25. The experimental results are similar to our prior observations that zero-extended values dominate the write values. In detail, zero-extended occupy more than 44.9 percent of all memory accesses on average and five workloads show more than 60 percent occupancy of zero-extended values (sjeng, gobmk, zeusmp, astar and libquantum). Even the worst test result still shows almost 23 percent zero-extended values (bzip2). The results of read latency are illustrated in Fig. 26. On average, FNW gets 38.8 percent read latency reduction compared with the baseline. In comparison, Three-Stage-Write gets 45.7 percent read latency reduction and Min-WU earns 40.0 percent improvement compared with the baseline. Min-WU-PF shows 19.0 and 11.9 percent more latency reduction compared with FNW and Three-stage-write, respectively. Fig. 27 illustrates the experimental results of write latency reduction of compared write schemes. In summary, Min-WU and Min-WU-PF respectively shows 29.9 and 46.0 percent write latency reduction compared with the baseline. Moreover, Min-WU-PF outperforms FNW and Three-Stage-Write, and shows 16.4 and 9.6 percent more write latency improvement. The experimental results of IPC improvement are illustrated in Fig. 28. Overall, Min-WU can get 28.7 percent IPC improvement and Min-WU-PF can get 43.0 percent IPC boost compared with the baseline. As a comparison, FNW and Three-Stage-Write can get 29.1 and 34.4 percent IPC improvement compared with the baseline, respectively. The experimental results of running time reduction are similar to IPC improvement. As shown in Fig. 29, Min-WU outperforms FNW and Three-Stage-Write in some benchmarks that are

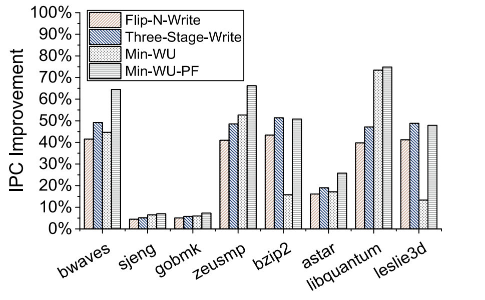


Fig. 28. IPC improvement.

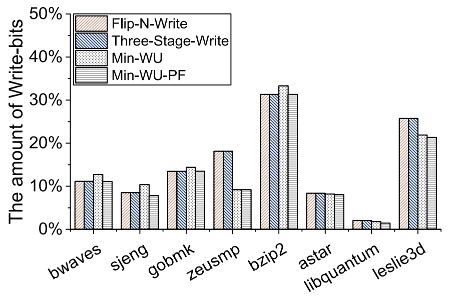


Fig. 30. The amount of write-bits.

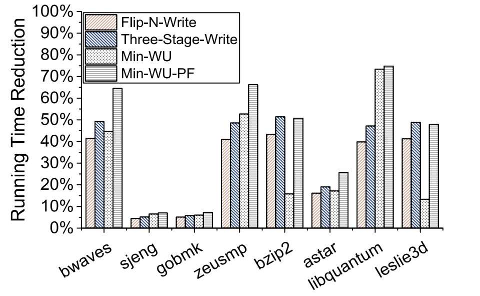


Fig. 29. Runnming time reduction.

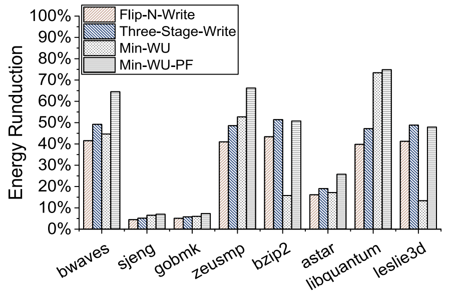


Fig. 31. Energy improvement.

zero-extended values dominant, such as sjeng, zeusmp and libquantum. In addition, Min-WU-PF outperforms FNW and Three-Stage-Write in all benchmarks and shows 14.0 and 8.7 percent more running time reduction. Results of the amount of write-bits are shown in Fig. 30. On average, the amount of write-bits of FNW and Three-Stage-Write is 13.3 percent compared with the baseline. As comparisons, Min-WU and Min-WU-PF reduces 87.2 and 88.2 percent data amount on average, respectively. As for the reduction of energy consumption illustrated in Fig. 31, Min-WU scheme can reduce the average energy consumption by 43.4 percent and Min-WU-PF can reduce 62.1 percent energy consumption compared with the baseline. In addition, FNW and Three-Stage-Write respectively show 39.7 and 47.0 percent energy consumption reduction compared with the baseline.

In summary, it proved that our proposed schemes, including Min-WU and Min-WU-PF, are efficient and effective with the capacity of LLC cache grows. Our designs also show significant performance improvement and energy consumption reduction under memory intensive workloads.

5 Related Work

DCW [50] is a quite sample and effective write PCM scheme for reducing the energy and improving endurance. By leveraging data-comparison write method, only different bits are written down to the PCM array. FNW [17] tries to extend the size of power budget to improve write parallelism. 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 [16] 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. Recently Three-Stage-Write is proposed in [33], the authors try to unite the work of FNW and 2-Stage-Write. By combining the data inversion of FNW with the 2-stage-write, the write process is divided into 3 stages, i.e., comparison, write “0” and write “1”.

Compression is widely used in the capacity constrained cache and disk backup systems [51]. 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 32 bits a word) and compresses each word according to data pattern [37]. There are some works utilizing FPC to reduce the bit-writes in NVM. Dgien et al. propose a compression-based memory architecture in Nonvolatile Memories (NVMs) combining FPC with FNW [52]. With the finding that FNW cannot work efficiently if the data have been compressed, the author carries out a fine-gained FNW to get further bit-write reduction. Recently in [53], 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.

6 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. We observe some frequent zero-extended values dominate the write data patterns in typical multi-threaded applications (more than 40 and 44.9 percent of all memory accesses in PARSEC workloads and SPEC 2006 benchmarks, respectively). By leveraging carefully designed data redistribution method, the data amount of each chip is balanced and the data pattern of each chip is the same. Min-WU has two main approaches: First, Min-WU reduces the total amount of data by leveraging simple data coding. Second, Min-WU tries to finish the cache line service with less write units by encapsulating more data bits into a write unit. Min-WU strikingly minimizes the number of write units, which accelerates the write while reducing 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. Extensive 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 reduces 44 percent read latency, 28 percent write latency, 32.5 percent running time and 48 percent energy while receiving 32 percent IPC improvement compared with the conventional write scheme. When combined with partly data flip, the variation of Min-WU (Min-WU-PF) yields 22 percent read latency reduction, 15 percent write latency decrease, 12 percent running time reduction, 23 percent energy saving and 12 percent IPC improvement, compared with Flip-N-Write. Min-WU and Min-WU-PF respectively reduce 53 and 56 percent data amount and can improve the endurance of PCM-based main memory. To explore the design space of our proposed schemes, we evaluate our design with memory-intensive SPEC 2006 benchmarks with a larger L3 cache. Experimental results show that Min-WU-PF yields 11.9 percent read latency reduction, 9.6 percent write latency decrease, 8.6 percent running time reduction, 15.1 percent energy saving and 8 percent IPC improvement, compared with state-of-the-art Three-Stage-Write. In addition, our design has the great potential in multi-threaded applications. We can get much more encouraging results in future with the wide use of multi-threaded programming.

Acknowledgments

This work was supported by the 863 Project No. 2015AA015301, No. 2013AA013203, No. 2015AA016701; NSFC No. 61502190, No. 61303046, No. 61472153; Wuhan Applied Basic Research Project (No. 2015010101010004). Fang Wang is the corresponding author.

References

[1] M. Zwolenski and L. Weatherill, “The digital universe: Rich data and the increasing value of the internet of things,” *Australian J. Telecommun. Digital Economy*, vol. 2, no. 3, 2014, Art. no. 47.

[2] V. L. Bernhardt and B. J. Geise, *Data, Data Everywhere*. Abingdon, U.K.: Routledge, 2015.

[3] New record set for China’s train ticket sales. (2014, Dec. 21). [Online]. Available: http://news.xinhuanet.com/english/china/2014-12/21/c_133869726.htm

[4] R. Martin, “Wall Street’s quest to process data at the speed of light,” *Inform. Week*, vol. 4, no. 21, 2007, Art. no. 07.

[5] S. Hoffmann, VODC Mini case study Alibaba Group. (2014, Nov. 16). [Online]. Available: http://www.adaptivecycle.nl/images/Use_Case_Alibaba.pdf

[6] L. Wilson, “International technology roadmap for semiconductors (ITRS),” *Semicond. Ind. Assoc.*, Washington, DC, USA, 2013.

[7] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, “Energy management for commercial servers,” *IEEE Comput.*, vol. 36, no. 12, pp. 39–48, 2003.

[8] L. A. Barroso and U. Hölzle, “The case for energy-proportional computing,” *IEEE Comput.*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[9] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, “RAIDR: Retention-aware intelligent DRAM refresh,” *ACM SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 1–12, 2012.

[10] D. Meisner, B. T. Gold, and T. F. Wenisch, “PowerNap: Eliminating server idle power,” *ACM Sigplan Notices*, vol. 44, no. 3, pp. 205–216, 2009.

[11] M. Pedram, “Energy-efficient datacenters,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 10, pp. 1465–1484, Oct. 2012.

[12] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, “Architecting phase change memory as a scalable DRAM alternative,” *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 2–13, 2009.

[13] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, “A durable and energy efficient main memory using phase change memory technology,” *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 14–23, 2009.

[14] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, “Scalable high performance main memory system using phase-change memory technology,” *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 24–33, 2009.

[15] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, “Improving write operations in MLC phase change memory,” in *Proc. IEEE Int. Symp. High-Performance Comp. Archit.*, 2012, pp. 1–10.

[16] J. Yue and Y. Zhu, “Accelerating write by exploiting PCM asymmetries,” in *Proc. IEEE 19th Int. Symp. High Performance Comput. Archit.*, 2013, pp. 282–293.

[17] S. Cho and H. Lee, “Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance,” in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 347–357.

[18] W. Zhou, D. Feng, Y. Hua, J. Liu, F. Huang, and Y. Chen, “An efficient parallel scheduling scheme on multi-partition PCM Architecture,” in *Proc. Int. Symp. Low Power Electron. Des.*, 2016, pp. 344–349.

[19] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, “Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling,” in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 14–23.

[20] N. H. Seong, D. H. Woo, and H.-H. S. Lee, “Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping,” *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 383–394, 2010.

[21] F. Huang, et al., “Security RBSG: Protecting phase change memory with security-level adjustable dynamic mapping,” in *Proc. IEEE Int. Parallel Distrib. Processing Symp.*, 2016, pp. 1081–1090.

[22] W. Zhou, D. Feng, Y. Hua, J. Liu, F. Huang, and P. Zuo, “Increasing lifetime and security of phase-change memory with endurance variation,” in *Proc. 22nd IEEE Int. Conf. Parallel Distrib. Syst.*, 2016, pp. 861–868.

[23] J. Yue and Y. Zhu, “Making write less blocking for read accesses in phase change memory,” in *Proc. IEEE 20th Int. Symp. Modeling, Anal. Simul. Comput. Telecommun. Syst.*, 2012, pp. 269–277.

[24] A. Hay, K. Strauss, T. Sherwood, G. H. Loh, and D. Burger, “Preventing PCM banks from seizing too much power,” in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2011, pp. 186–195.

[25] L. Jiang, Y. Zhang, B. R. Childers, and J. Yang, “FPB: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory,” in *Proc. Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2012, pp. 1–12.

[26] K.-J. Lee, et al., “A 90 nm 1.8 V 512 Mb diode-switch PRAM with 266 MB/s read throughput,” *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 150–162, Jan. 2008.

[27] Z. Li, et al., “Exploiting more parallelism from write operations on PCM,” in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2016, pp. 768–773.

[28] Z. Li, et al., “Tetris write: Exploring more write parallelism considering PCM asymmetries,” in *Proc. 45th IEEE Int. Conf. Parallel Process.*, 2016, pp. 159–168.

[29] Z. Li, F. Wang, D. Feng, Y. Hua, J. Liu, and W. Tong, "MaxPB: Accelerating PCM write by maximizing the power budget utilization," *ACM Trans. Archit. Code Opt.*, vol. 13, no. 4, 2016, Art. no. 46.

[30] J. Yue and Y. Zhu, "Exploiting subarrays inside a bank to improve phase change memory performance," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2013, pp. 386–391.

[31] G. Sun, Y. Zhang, Y. Wang, and Y. Chen, "Improving energy efficiency of write-asymmetric memories by log style write," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Des.*, 2012, pp. 173–178.

[32] Z. Li, et al., "A software-defined fusion storage system for PCM and NAND flash," in *Proc. Non-Volatile Memory Syst. Appl. Symp.*, 2015, pp. 1–6.

[33] Y. Li, X. Li, L. Ju, and Z. Jia, "A three-stage-write scheme with flip-bit for PCM main memory," in *Proc. 20th Asia South Pacific Des. Autom. Conf.*, 2015, pp. 328–333.

[34] B. Correa, R. Mesquita, and L. Amorim, "CUDA approach for meshless local Petrov-Galerkin method," *IEEE Trans. Magn.*, vol. 51, no. 3, pp. 1–4, Mar. 2015.

[35] G. Liu, T. Schmidt, R. Domer, A. Dingankar, and D. Kirkpatrick, "Optimizing thread-to-core mapping on manycore platforms with distributed Tag Directories," in *Proc. 20th Asia South Pacific Des. Autom. Conf.*, 2015, pp. 429–434.

[36] M. Arjomand, A. Jadidi, A. Shafiee, and H. Sarbazi-Azad, "A morphable phase change memory architecture considering frequent zero values," in *Proc. IEEE 29th Int. Conf. Comput. Des.*, 2011, pp. 373–380.

[37] A. R. Alameldeen and D. A. Wood, "Frequent pattern compression: A significance-based compression scheme for L2 caches," Dept. Comput. Sciences, Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1500, 2004.

[38] J. Yang, Y. Zhang, and R. Gupta, "Frequent value compression in data caches," in *Proc. 33rd Annu. ACM/IEEE Int. Symp. Microarchitecture*, 2000, pp. 258–265.

[39] Y. Zhang, J. Yang, and R. Gupta, "Frequent value locality and value-centric data cache design," *ACM SIGOPS Operating Syst. Rev.*, vol. 34, no. 5, pp. 150–159, 2000.

[40] G. Sun, D. Niu, J. Ouyang, and Y. Xie, "A frequent-value based PRAM memory architecture," in *Proc. 16th Asia South Pacific Des. Autom. Conf.*, 2011, pp. 211–216.

[41] F. Xia, D. Jiang, J. Xiong, M. Chen, L. Zhang, and N. Sun, "DWC: Dynamic write consolidation for phase change memory systems," in *Proc. 28th ACM Int. Conf. Supercomputing*, 2014, pp. 211–220.

[42] H. Luo, L. Shi, M. Zhao, Q. Zhuge, and C. J. Xue, "Improving MLC PCM write throughput by write reconstruction," in *Proc. IEEE Non-Volatile Memory Syst. Appl. Symp.*, 2015, pp. 1–6.

[43] H. Luo, P. Dai, L. Shi, C. J. Xue, Q. Zhuge, and E. H. Sha, "Write reconstruction for write throughput improvement on MLC PCM based main memory," *J. Syst. Archit.*, vol. 71, pp. 62–72, 2016.

[44] Z. Li, et al., "A user-visible solid-state storage system with software-defined fusion methods for PCM and NAND flash," *J. Syst. Archit.*, vol. 71, pp. 44–61, 2016.

[45] N. Binkert, et al., "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[46] M. Poremba, T. Zhang, and Y. Xie, "NVMain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE Comput. Archit. Lett.*, vol. 14, no. 2, pp. 140–143, Jul.–Dec. 2015.

[47] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches," HP Laboratories, Palo Alto, CA, USA, Tech. Rep. HPL-2009-85, pp. 22–31, 2009.

[48] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Department of Computer Science, Princeton University, Princeton, NJ, USA, Jan. 2011.

[49] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.

[50] B.-D. Yang, et al., "A low power phase-change random access memory using a data-comparison write scheme," *Proc. IEEE Int. Symp. Circuits Syst.*, 2007, pp. 3014–3017.

[51] P. M. Palangappa and K. Mohanram, "CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM," in *Proc. IEEE Int. Symp. High Performance Comput. Archit.*, 2016, pp. 90–101.

[52] 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," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, 2014, pp. 51–56.

[53] P. M. Palangappa and K. Mohanram, "Flip-Mirror-Rotate: An architecture for bit-write reduction and wear leveling in non-volatile memories," in *Proc. 25th Edition Great Lakes Symp. VLSI*, 2015, pp. 221–224.

Zheng Li received the BE degree in computer science and technology from the Huazhong University of Science and Technology (HUST), China, in 2013. He is currently working toward the PhD degree in computer architecture from HUST. His research interest includes reconfigurable computing on FPGA and non-volatile memory-based storage system. He publishes several papers in major journals and conferences including TACO, JSA, DATE, ICPP, NVMSA etc..

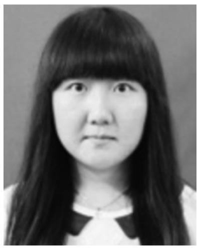
Fang Wang received the BE, ME, and PhD degrees in computer science and technology from the Huazhong University of Science and Technology (HUST), China, in 1994, 1997, and 2001, respectively. She is a professor of the School of Computer Science and Technology, HUST. Her research interests include computer architecture, massive storage systems, and parallel file systems. She has more than 40 publications to her credit in journals and international conferences including ACM TACO, SC, MSST, ICPP, ICA3PP, HPDC and ICDCS.

Dan Feng received the BE, ME, and PhD degrees in computer science and technology from the Huazhong University of Science and Technology (HUST), China, in 1991, 1994, and 1997, respectively. She is a professor and vice dean of the School of Computer Science and Technology, HUST. Her research interests include computer architecture, massive storage systems, and parallel file systems. She has more than 80 publications to her credit in journals and international conferences, including the *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, JCST, USENIX ATC, FAST, ICDCS, HPDC, SC, ICS and ICPP. She is a member of the IEEE.

Yu Hua received the BE and PhD degrees in computer science from the Wuhan University, China, in 2001 and 2005, respectively. He is a professor with Huazhong University of Science and Technology, China. His research interests include computer architecture, cloud computing and network storage. He has more than 60 papers to his credit in major journals and international conferences including the *IEEE Transactions on Computers (TC)*, the *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, USENIX ATC, FAST, INFOCOM, SC, ICDCS and MSST. He has served on the program committee for multiple international conferences, such as INFOCOM, RTSS, ICDCS, MSST, ICNP, ICPP, IWQoS. He is a senior member of the IEEE and the CCF.

Jingning Liu received the BE degree in computer science and technology from the Huazhong University of Science and Technology (HUST), China, in 1982. She is a professor in the HUST and engaged in researching and teaching of computer system architecture. Her research interests include computer storage network system, high-speed interface and channel technology, embedded system and FPGA design. She has more than 20 publications in journals and international conferences including ACM TACO, NAS, MSST and ICA3PP.

Wei Tong received the BE, ME, and PhD degrees in computer science and technology from the Huazhong University of Science and Technology (HUST), China, in 1999, 2002, and 2011, respectively. She is a lecturer of the School of Computer Science and Technology, HUST. Her research interests include computer architecture, network storage system, and solid state storage system. She has more than 10 publications in journals and international conferences including ACM TACO, MSST, NAS, FGCN.



Yu Chen received the BE degree in computer science and technology from the Huazhong University of Science and Technology (HUST), China, in 2013. She is currently working toward the PhD degree in computer architecture from HUST. Her research interest includes Software-defined Storage. She publishes several papers in major conferences including DATE, ISLPED etc.

Salah S. Harb received the BS and MS degrees both in computer engineering from Jordan University of Science and Technology (JUST), Jordan, in 2011 and 2014, respectively. He is now doing a research on cryptographic algorithms to get the PhD from Huazhong University of Science and Technology (HUST). His research interests include the hardware implementations of encryption algorithms and the embedded system designs for cryptosystems.