IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 12, 2022, accepted January 21, 2022, date of publication January 25, 2022, date of current version February 1, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3145986

WL-WD: Wear-Leveling Solution to Mitigate Write Disturbance Errors for Phase-Change Memory

MOONSOO KIM[1], HYOKEUN LEE[2], (Member, IEEE), HYUN KIM[3], (Member, IEEE), AND HYUK-JAE LEE[2], (Member, IEEE)

[1]Samsung Electronics, Hwasung 18448, South Korea
[2]Inter-University Semiconductor Research Center, Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea
[3]Research Center for Electrical and Information Technology, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea

Corresponding author: Hyun Kim (hyunkim@seoultech.ac.kr)

This work was supported in part by the Ministry of Trade, Industry and Energy (MOTIE) (DRAM/PRAM Heterogeneous Memory Architecture and Controller IC Design Technology Research and Development) under Grant 10080613; and in part by the Technology Innovation Program (Development of Open Convergence Memory Solution and Platform for Next Generation Memories) funded by MOTIE, South Korea, under Grant 20011074.

ABSTRACT Phase-change memory (PCM) is a promising non-volatile memory device due to its attractive properties such as fast access time and byte-addressability. However, PCM is still difficult to be used as a main memory because of its weakness in endurance and write disturbance. Conventional wear-leveling algorithms have attempted to handle short endurance, but they have not addressed the issue of write disturbance even though both issues are caused by write operations to PCM. This paper proposes a wear-leveling algorithm that addresses not only the endurance, but also the write disturbance. From the observation that the write disturbance errors and short endurance issues mostly take place in hot addresses, the proposed algorithm first detects hot addresses and maps them to 'hot' regions, which are customized memory regions designed to be robust to write disturbance errors and to support effective wear-leveling. The 'hot' region is not fixed to a specific part, but it moves over an entire memory space to make every cell belong to 'hot' regions in a uniform manner. On the other hand, cold addresses are mapped in 'normal' memory regions by simple linear mapping to reduce the hardware overhead. The proposed algorithm can reduce the write disturbance errors by more than 70% with only a slight instruction per cycle (IPC) degradation. Moreover, the wear-leveling performance is also enhanced by more than 3% compared to other wear-leveling algorithms.

INDEX TERMS Phase-change memory, write disturbance error, wear-leveling, memory reliability, memory endurance.

I. INTRODUCTION

A modern computer system requires large amounts of main memory owing to its multi-core structure and complex applications. In particular, data-intensive applications such as big data and deep learning require a large main memory to support large amounts of data. As a result, the need for large main memory has become important, and studies on the use of phase-change memory (PCM) as main memory have been actively conducted [1]–[4]. The cell size of PCM is smaller than that of DRAM, so the module can be more dense, enabling large memory capacity [5]. Furthermore, owing to the non-volatile characteristics of PCM, it is advantageous over DRAM in terms of power efficiency and data retention time.

Despite these advantages, PCM suffers from low reliability and endurance problems, which need to be addressed in order to use PCM as main memory. Reliability issues in PCM are mainly caused by write disturbance errors (WDEs) [6]–[8]. WDEs refer to a phenomenon whereby surrounding cells of the written cell are damaged owing to thermal interference. Frequent exposure to thermal interference causes errors in the surrounding cells. As PCM cells are scaled down, the thermal interference between cells becomes severe, resulting

The associate editor coordinating the review of this manuscript and approving it for publication was Luca Cassano.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

in the incidence of more WDEs. WDEs were first observed at 54 nm [9], and then appear to be more severe at 20 nm and below [10], [11]. The use of PCM as main memory makes WDE issues worse because main memory performs much more write operations than storage.

Another obstacle for using PCM as a main memory is low endurance [3], [12]. If a write operation occurs more frequently than the endurance, the cell can no longer store data. Similar to the WDE issue, the endurance issue becomes severe when PCM is used as main memory. In order to use PCM as long as possible under the limited endurance, all cells in the memory must be written uniformly. An algorithm to achieve this goal is called a wear-leveling algorithm.

To use PCM as the main memory, both issues (i.e., reliability and endurance issues) must be properly addressed. Until now, however, these two issues have been addressed independently. WDE-mitigating schemes [5], [13]–[16] do not consider the endurance, which may adversely affect endurance. Similarly, wear-leveling algorithms [17]–[19] do not take into account WDE occurrence. To address both issues, the WDE-mitigating scheme and wear-leveling algorithm must be used together, increasing the memory-access latency. The motivation of this work is that it is possible to remove WDEs when designing the wear-leveling algorithm. This is because a wear-leveling algorithm redirects the physical address to distribute the write operation uniformly, and therefore, the occurrence of WDEs is affected by this address redirection. For example, if write operations are concentrated in a single address, the neighbor addresses become very vulnerable to WDEs. On the other hand, if write operations are distributed to consecutive addresses, WDE occurrence becomes relatively low. This means that the occurrence of WDEs varies depending on how the wear-leveling algorithm is performed.

This paper presents a proposal of WL-WD, a wear-leveling algorithm that mitigates WDEs. Given the fact that WDE and endurance issues occur mostly in hot addresses that are frequently accessed, the proposed algorithm manages hot addresses separately in a customized 'hot' region. The 'hot' region utilizes additional resources to support robustness to WDEs and an effective wear-leveling algorithm. In the case of cold addresses where WDE and endurance issues are not severe, the algorithm uses a simple method without requiring the use of additional hardware resources. As a result, the proposed algorithm demands a relatively small overhead to effectively mitigate WDEs and enhance the wear-leveling performance. It should be noted that the ratio of hot address is only about 1.6% of the total addresses; hence, the hardware overhead can be significantly reduced. In particular, the major contributions of this paper are summarized as follows:

- This paper presents a wear-leveling algorithm robust to WDEs. Previous wear-leveling algorithms do not consider WDE issues, and therefore, both schemes, WDE mitigation solution and wear-leveling solution, need to be used together. This leads to inefficiencies in terms of system performance. The proposed WL-WD can achieve a 10-25% increase in the system performance by addressing two issues with a single algorithm.
- The wear-leveling performance of the WL-WD is relatively high. When PCM is used as the main memory, passive wear-leveling is mainly used because active wear-leveling incurs a large amount of metadata. The WL-WD uses active wear-leveling only to the 'hot' region, whereas most of the remaining mapping is controlled by passive wear-leveling. By adaptively using active and passive wear-leveling, the WL-WD gives a greater than 3% increase in the wear-leveling performance while reducing the write disturbance errors by more than 70%.

The rest of this paper is organized as follows. Section II introduces the background of PCM, WDE, and wear-leveling. Section III illustrates the proposed algorithm in detail. In Section IV, experimental results are presented. Finally, Section VI concludes the paper.

II. BACKGROUND

A. PCM BASICS

PCM stores data using phase-change material such as Ge2Sb2Te5 (GST) [5]. Phase-change material has two states, the high-impedance amorphous state (bit "0") and the low-impedance crystalline state (bit "1"). If a high temperature that exceeds the melting point of the material is applied, the material changes to the amorphous state (RESET operation). However, if the temperature is below the melting point, the material enters the crystalline state (SET operation). In summary, RESET is the operation to change data from 1 to 0, whereas SET switches the data from 0 to 1.

The PCM chip has a similar structure to the DRAM chip. According to the PCM chip implementation in [20], the PCM chip consists of several partitions (see Figure 1), each of which is composed of several tiles. A tile is an array structure in which many word lines (WLs) and bit lines (BLs) overlap each other. PCM cells are located where WL and BL meet. According to [20], an 8-Gb PCM chip has eight partitions, each of which consists of 128 tiles. A tile consists of 4096 WLs and 2048 BLs.

The physical layout of the partition is divided into 128 tiles, but the address mapping considers the entire partition as a single bundle. This single bundle, which is depicted in the rightmost side of Figure 1, has 4K WLs and $2K \cdot 128$ BLs. When 128 BLs are grouped together into a single column, a whole partition is divided into 2K columns. This means that the partition is mapped with 4K rows and 2K columns. When a physical address is specified to access the PCM chip, the address is decoded to generate partition, row, and column addresses. The partition address first selects the partition, and the row and column addresses are used to select the row and column inside the partition. The corresponding 128-bit unit is applied to the output. In summary, a partition is addressed to $4K \cdot 2K$ 128-bit units.

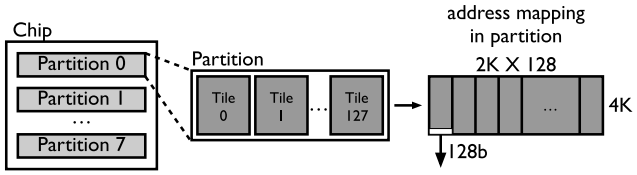


FIGURE 1. Diagram showing an overview of the PCM chip structure.

B. WRITE DISTURBANCE ERRORS IN PCM

Write disturbance errors (WDEs) are caused by thermal interference between PCM cells. The heat applied to one cell can be transferred along BL and WL to change the state of the surrounding cells. In particular, the high temperature that is applied during RESET operations significantly affects surrounding cells. The temperature of the surrounding cell that receives heat by thermal interference is clearly lower than that of the origin cell. However, it can still reach the temperature at which the SET operation occurs. When this situation is repeated, a SET operation occurs in the surrounding cells, and errors occur. This means that a RESET operation repeatedly takes place in one cell, the cell which stores "0" in the surrounding cells is damaged by a SET operation, and WDE occurs.

Previous studies have mainly used probabilistic modeling of WDE [13], [14], [16]. The probabilistic model assumes that WDE occurs with a fixed probability for every RESET operation. However, studies have shown that counter-based modeling is more plausible than probabilistic modeling [21], [22]. Counter-based modeling means that WDE occurs when the cumulative time of the disturbance exposure in one cell exceeds a certain value. The difference between two models is illustrated with an example. Assume that the WDE occurs once every 100 RESETs. The probabilistic model models WDE with a probability of 0.01 for every RESET, while the counter-based model definitively generates WDE at the 100th RESET. In this paper, we use counter-based modeling as in [21], [22].

Several studies have been conducted to mitigate the WDEs. The simplest and most straightforward approach is Verify and Correct (VnC) [5]. When a write operation takes place, VnC reads neighboring cells and verifies the data. If errors occur in the read data, they are corrected and restored by performing additional write operations. VnC is a powerful method that is employed to eliminate WDEs. However, it incurs significant performance degradation because of additional read and write operations. According to [16], when using VnC, the number of cycles per instruction (CPIs) increases by up to 2 times. Further, additional writes worsen the endurance.

Other approaches reduce WDE vulnerable patterns by encoding data [13], [16]. WDE occurs in neighboring "0" cells when a cell takes a RESET operation. Therefore, data patterns such as "0" and "10" are vulnerable to WDEs. Encoding approaches remove those patterns to reduce WDE occurrence. Since the encoding increases the number of data bits, those approaches first compress the data and then encode them. This means that if the compression ratio is not enough, the encoding cannot be done effectively either.

Decongest [15] stores frequently accessed hot addresses in a spare region in which WDE does not occur. The spare region is designed to have a cell size of $8F^2$, which is twice the size of the general cell ($4F^2$). However, this method has a problem in that the spare region quickly wears out because it holds frequently accessed addresses in a small spare region.

C. WEAR-LEVELING IN PCM

The cell endurance of the PCM is the maximum number of times that a cell can be written. If the number of writes in a cell exceeds the endurance value, the cell will no longer be able to store data, indicating that the cell is dead. To maximize the use of every cell in a PCM module without a dead cell, writes must be distributed uniformly across all cells. An algorithm with this function is called a wear-leveling algorithm.

Conventional wear-leveling algorithms are divided into two categories: active and passive wear-leveling. First, the active wear-leveling algorithm performs address mapping dynamically using a mapping table for each access unit. The active wear-leveling shows relatively better wear-leveling performance because it can change mapping as desired. These kinds of algorithms are suitable for storage devices because the size of an access unit is relatively large (about 4 KB) [23]. However, it is difficult to use these methods for main memory with a smaller access unit (about 64 B) because the number of entries in the mapping table is very large. For example, the number of total access units is 128M, considering an 8-GB main memory with a 64 B granularity. Assuming that a mapping table entry has a size of about 3B, the size of the whole mapping table is about 384 MB. This overhead is almost 1/20 of the assumed memory size, making the methods unsuitable for main memories. On the other hand, passive wear-leveling uses a simple method such as periodically changing the address under certain rules. The metadata overhead is small because these methods do not store the mapping table for each unit. However, the wear-leveling performance is worse than that of active wear-leveling.

The cell endurance of PCM is higher than that of flash memory, and the size of the access unit is relatively small (64 B - 512 B). Therefore, when PCM is used as the main memory, passive wear-leveling is typically used [17], [18]. Start-gap [17] periodically changes the address with a consecutive neighboring address. Security Refresh [18] generates an address by performing an XOR operation with a stored key value. The key value is changed periodically, making the generated address also change accordingly. These algorithms show considerable wear-leveling performance with small overhead. However, an additional WDE mitigating scheme must be accompanied because they do not consider WDEs.

III. DESIGN DETAILS OF WL-WD

In this section, we explain WL-WD, a wear-leveling solution to mitigate WDEs.

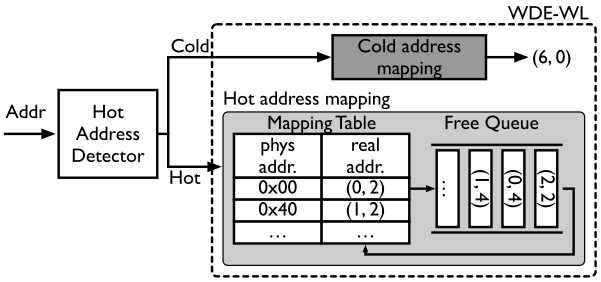


FIGURE 2. Diagram giving an overview of the WL-WD.

A. MOTIVATION AND ARCHITECTURAL OVERVIEW

The ideal way to remove WDEs is to newly write the cell before WDEs occur. For example, if the WDE occurs when a cell has been disturbed 1,000 times by write operations on adjacent cells, the goal of the WDE reduction algorithm is to allocate the new write operation before the disturbance count reaches 1,000. To this end, disturbance count for every address must be stored, which results in a severe overhead.

To reduce this overhead, the proposed algorithm manages hot and cold addresses separately. Hot addresses that represent frequently accessed addresses are vulnerable to both WDE and endurance issues. Therefore, these hot addresses are dynamically mapped to 'hot' region that are specially designed to solve both issues effectively. It should be noted that the 'hot' regions occupy only small portion and thus, incurring only a small overhead. The remaining memory space holds cold addresses that are not frequently accessed. In this 'normal' region, the address mapping is done by static linear mapping to reduce the metadata overhead. In other words, the proposed algorithm chooses dynamic and static mapping schemes depending on the frequency of memory access, thereby taking advantage of both algorithms.

An important feature of the proposed algorithm is that the 'hot' region is not fixed in a specific area. Although active wear-leveling makes cells in the 'hot' region uniformly written, the entire 'hot' region can wear-out faster than the 'normal' region because the whole region is accessed more frequently. Therefore, the proposed algorithm slides the 'hot' region periodically. If the slide operation is repeated continuously, all memory space will eventually experience the 'hot' region evenly. In addition, the passive wear-leveling of the cold addresses is achieved naturally by this slide operation.

Figure 2 shows the overview of the proposed algorithm. When the address is set, the hot address detector (HAD) distinguishes between hot and cold addresses. We use the HAD proposed in HAD-TWL [24]. It should be noted that HAD itself is not within the research scope of this paper, and the algorithm can work similarly even if other HADs are used. The hot address determined by the HAD is mapped to the 'hot' region using the hot address mapping system. Hot address mapping, which supports active wear-leveling, uses a mapping table and free queue information. In the free queue, the locations of free units (not mapped unit) in the 'hot' region are stored in the form of (*row*, *column*). The location of a free unit is popped from the free queue and mapped with the address. This mapping information is stored in the mapping table. When the address is accessed again, the mapping is changed to the oldest free unit in the queue. As such, mapping in the 'hot' region requires additional resources, such as the mapping table and the free queue. Therefore, the 'hot' region is assigned only as needed, and the remaining region is mapped using cold address mapping, which does not require additional resources. Cold addresses are not susceptible to the WDE and endurance issues; thus, they are handled with a simple linear mapping.

The following subsections describe the structure of the 'hot' region, hot address mapping, and cold address mapping method in detail.

B. STRUCTURE OF THE 'Hot' REGION

Figure 3 shows the detailed structure of the 'hot' region. Firstly, a partition in the chip is divided by several sub-partitions. Each sub-partition has a separate WL-WD structure, as shown in Figure 2. Denote the number of rows and columns of a sub-partition as r and c, respectively. Each sub-partition has its own 'hot' region. Figure 3 (a) shows the initial state of the 'hot' region. The gray dotted box indicates the location of 'hot' region. Basically, 'hot' region occupies a columns, which means that the size of the 'hot' region is $r \cdot a$. These 'hot' regions require additional PCM cells. Typically, PCM chips have spare regions that can be used for various purposes, such as error correction [25], [26], or wear-leveling [17]. Therefore, the spare region can also be utilized for the proposed algorithm because the proposed algorithm is related to error reduction and wear-leveling.

An important feature of the 'hot' region is that it is not fixed to a specific part of the PCM. Owing to the nature of the 'hot' region to which the hot addresses are mapped, most writes are concentrated on the 'hot' region. If the 'hot' region is fixed in a specific region, the cells in the region are dead quickly. To prevent this situation, the proposed 'hot' region is moved using a slide operation. The slide operation is repeatedly performed, eventually resulting in all of the cells experiencing the 'hot' region evenly. The number of writes that triggers the slide operation is denoted as the slide interval (SI). For example, if the SI value is 256, the slide operation is performed every 256 writes.

The slide operation is performed by swapping the hot unit with the cold unit. Figure 3 (b) shows the detailed slide operation. The position of the unit is indicated in the form of (*row*, *column*). The unit in front of the 'hot' region is (0, 0). This (0, 0) unit is swapped with a cold unit, which is (0, 8). After the slide operation, the unit at the front of the 'hot' region becomes (1, 0), and the unit at the back becomes (0, 8). As a result, unit (0, 0) is no longer the 'hot' region, and it is changed to the cold unit. The start point of the 'hot' region is denoted as a 'hot' region pointer (HRP). A single slide operation moves the HRP by one. For example, the slide operation in Figure 3 changes HRP from (0, 0) to (1, 0).

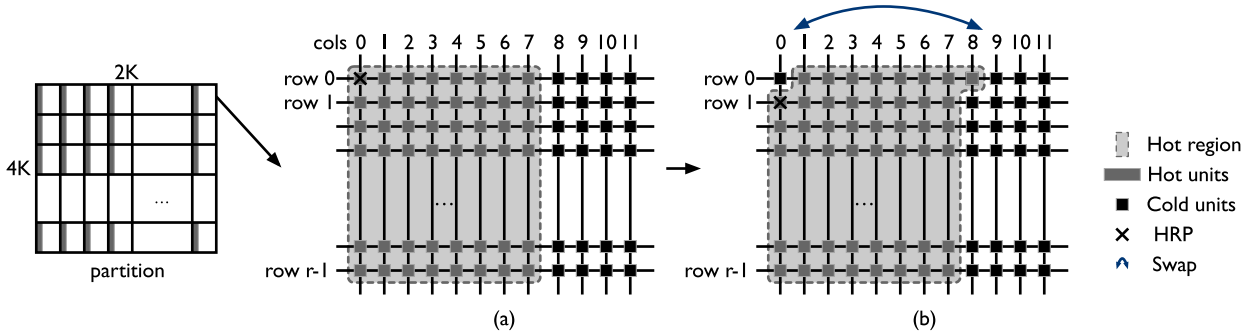


FIGURE 3. Example structure of the 'hot' region and its slide operation: (a) the initial state of the hot region, (b) the hot region after carrying out the slide operation.

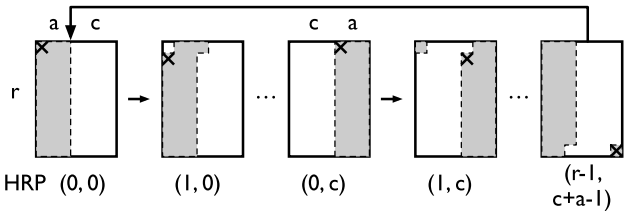


FIGURE 4. Slide cycle of the 'hot' region.

By repeating the slide operation, the 'hot' region is distributed evenly across the whole partition.

Figure 4 shows the complete cycle of the slide operation. The total number of rows and columns of the sub-partition are r and c, respectively. The number of columns in the 'hot' region is denoted as a. When a slide occurs in the initial state, HRP is changed from (0, 0) to (1, 0). The HRP finally reaches $(0, c)$ after $r \cdot c$ slides. At this time, the 'hot' region is located at the right-most side of the sub-partition. If the slide operation is performed at this moment, the HRP becomes $(1, c)$, and the 'hot' region is divided into two parts: one unit in the upper left corner, and the remaining units on the right side. Finally, when the HRP reaches $(r-1,\ c+a-1)$, one unit is left in the lower right corner, and the remaining units are located on the left side. One more slide at this moment makes the 'hot' region return to the initial state.

One slide operation causes two additional writes, one for a hot unit and one for cold unit, resulting in performance degradation. The frequency of the slide operation varies according to the SI value. The larger the SI, the less frequently does the slide operation occur. As the SI value decreases, the 'hot' region moves faster within the memory space, thus improving wear leveling performance. However, the decreased SI value also result in more additional writes, incurring a performance degradation.

C. HOT ADDRESS MAPPING

This section explains hot address mapping in the 'hot' region. Hot address mapping is performed dynamically by using a mapping table and free queue. The mapping table stores the mapping information between the physical address and the location of the unit. The free queue stores the locations of the free units in the 'hot' region.

Figure 5 presents details of how the mapping table and the free queue work during the mapping process. Figure 5 (a) shows the mapping table and the free queue in the initial state. Because mapping is not yet established at this moment, the mapping table is empty, and the free queue contains all of the unit locations in the 'hot' region. At this moment, when the address 0×00 is determined as the hot address, the right-most element in the free queue is popped and mapped with 0×00. As a result, 0×00 and (0, 0) are mapped and added to the mapping table, as shown in Figure 5 (b). If the address 0×40 is then accessed, the mapping table is first visited to check that the address entry exists, and the right-most unit (1, 0) is thus popped and allocated to 0×40 because the address 0×40 does not exist in the table.

The mapping table in Figure 5 (c) shows that 0×00 and 0×40 are mapped to (0, 0) and (1, 0), respectively. In this scenario, consider the case in which the address 0×40 is again accessed. Since 0×40 is already in the mapping table, the proposed algorithm attempts to change the mapping for every write to distribute the write operation as much as possible. To this end, the existing mapping is always invalidated. Mapping 0×40 - (1, 0) is deleted, and the unit (1, 0) is pushed to the left-most side of the free queue. The reason for pushing it to the left-most side is to reuse the unit as late as possible. The invalidated unit is likely to have been used recently, and reusing the unit late enhances the wear-leveling performance. The rightmost element of the free queue is popped and newly mapped with 0×40. As a result, 0×40 is now mapped to the (2, 0) unit, as shown in (d). When the address 0×40 is accessed again, (2, 0) is inserted into the free queue and (3, 0) is popped and mapped. This indicates that if an address (0×40) is repeatedly written, the proposed algorithm distributes them over multiple units ($(1, 0) \rightarrow (2, 0) \rightarrow (3, 0)$).

When the slide operation occurs, the hot and cold units are swapped, and the mapping table and the free queue must be handled accordingly. Taking the slide operation of Figure 3 as an example, (0, 0) is removed and (0, 8) is added to the 'hot' region. Therefore, the (0, 0) unit in the mapping table or the free queue must be changed to (0, 8). If the unit exists in the

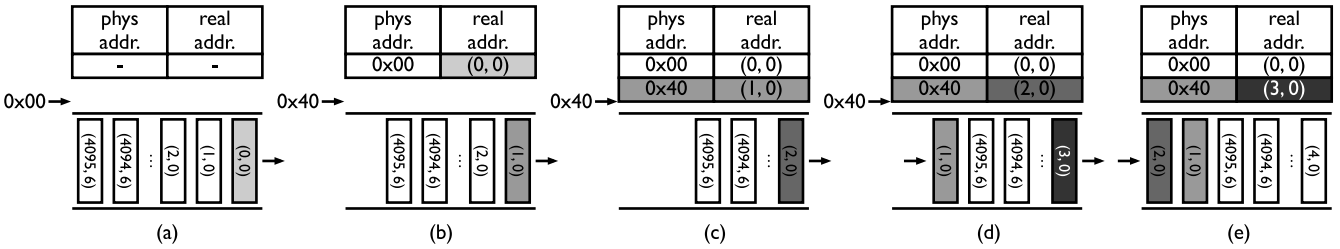


FIGURE 5. Hot address mapping examples in detail: (a) the initial state, (b) address 0 × 0 becomes the hot address, (c) address 0 × 40 is remapped to (1, 0), (d) address 0 × 40 is newly remapped to (2, 0), (e) address 0 × 40 is newly remapped to (3, 0).

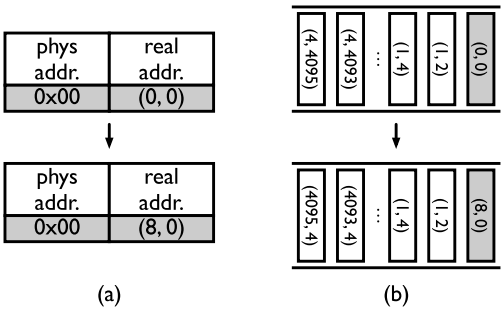


FIGURE 6. Handling the slide operation in (a) the mapping table, and (b) the free queue.

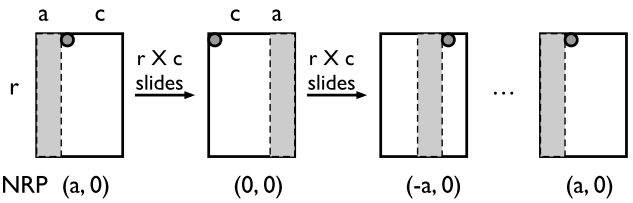


FIGURE 7. Slide cycle of the 'normal' region.

mapping table, the table entry is merely changed from (0, 0) to (8, 0), as shown in Figure 6 (a). If the unit is in the free queue, the unit is removed. Subsequently, the new unit is inserted into the free queue, as shown in Figure 6 (b). Because the newly added unit is originally a cold unit, it is likely that the added unit is less written than other units in the 'hot' region. Therefore, the unit is added to the right-most side to use it as fast as possible.

D. COLD ADDRESS MAPPING

This section describes the concept of cold address mapping, which is done in a non-'hot' region ('normal' region) of the memory, in detail. In the slide operation of Figure 3, the cold units (0, 8) is moved to (0, 0). This indicates that the slide operation moves one cold unit to the left by a (the number of columns in the 'hot' region). Figure 7 shows the slide cycle of the 'normal' region. The starting point of the 'normal' region is denoted as the 'normal' region pointer (NRP). At the initial state, the NRP is (0, a). There are $r \cdot c$ units in total, and $r \cdot c$ slides are needed to move the entire 'normal' region to the left. This means that the NRP becomes (0, 0) after $r \cdot c$ slides, the (-a, 0), and eventually returns to (a, 0) again.

The row and column addresses, which are decoded from the physical address, determine the location of the cold unit. The original location (row, $column$) is denoted as x. Figure 8 depicts the appearance of the 'normal' region. As shown in the right figure, the 'normal' region is identical to the normal memory space when the 'hot' region is excluded. The position x is used as a relative position inside the 'normal' region. The starting point of the 'normal' region (NRP) and x are added to obtain the absolute location of the unit to be mapped. The left figure of Figure 8 shows that the 'normal' region is divided into two parts. The (a) part uses the absolute location as is. However, the (b) part must be shifted right by the amount of a because of the 'hot' region width. Therefore, $(0,\ a)$ is additionally added to the absolute location for units in part (b). The condition for the absolute location (square symbol in the figure) to be in region (b) is $NRP < HRP \leq NRP + x$. In summary, cold address mapping converts the original position x determined by the physical address to y, as shown in the following equation.

$$y = \begin{cases} NRP + x + (0, a) & (NRP < HRP \leq NRP + x) \\ NRP + x & (otherwise) \end{cases} \quad (1)$$

An example is illustrated with the situation in Figure 3. In Figure 3 (a), the NRP is (0, a) and HRP is (0, 0), and $NRP \not\leq HRP$. Therefore, $y = NRP + x$ always stands in this case. In Figure 3 (b), $NRP < HRP$ is satisfied because NRP and HRP are moved to (0, 0) and (1, 0), respectively. When x is (0, 0) or (0, 1), $NRP + x$ is not less than HRP, making $y = NRP + x$. When x is greater than (0, 1) $HRP \leq NRP + x$ is satisfied and y becomes $NRP + x + (0, a)$. These derivations can also be seen intuitively in the figure. The location (0, 0) and (0, 1) are used as they are because they are on the left side of the 'hot' region, and (0, a) is added to the other locations because they are on the right side.

Mapping the cold address in this manner causes the address mapping not to be fixed to one unit location. Because NRP moves periodically from the slide operation, the y value is continuously changed for a fixed x. Therefore, repetitive writes at one location x are distributed over multiple locations, showing a wear-leveling effect and reducing the WDE.

E. PREVENTING THE WDE OCCURRENCE

In this section, the way of preventing WDE occurrence under the proposed mapping algorithm is described. To prevent

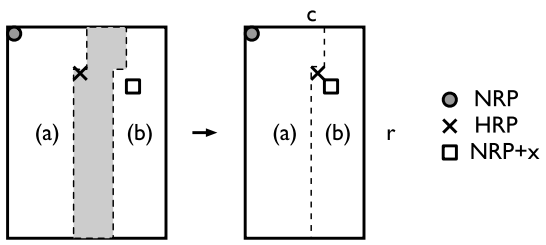


FIGURE 8. Appearance of the 'normal' region.

WDE occurrence, every unit must be moved before the WDE occurs. First, hot address mapping is analyzed. The size of the 'hot' region is $r \cdot a$ and therefore, $r \cdot a \cdot SI$ writes are needed to swap every unit in the 'hot' region. Within $r \cdot a \cdot SI$ writes, the WDE does not occur if all units in the 'hot' region are disturbed less than T (WDE threshold). Based on this observation, the following condition is derived.

$$T > r \cdot a \cdot SI \tag{2}$$

The left side of the condition is the worst-case scenario where one unit is repeatedly disturbed. If the left side is greater than the right side, the WDE does not occur. However, typical value for T, r, a, and SI is 1000, 64, 1, and 32, making the right side greater than the left side. To loosen the condition, the number of hot units, h, is set less than the size of the 'hot' region. This means that not all units in the 'hot' region are utilized, and $r \cdot a - h$ number of units are spared. For example, if the size of the 'hot' region ($r \cdot a$) is 64 and h is 60, 4 units are still spared. Under the worst-case scenario where a single address is repeatedly accessed, these spared units are used in turn. Therefore, condition (2) is loosened as follows:

$$(r \cdot a - h + 1)T > r \cdot a \cdot SI \tag{3}$$

If h is the same with $r \cdot a$, then the condition (3) is equivalent with (2). If h is set to 62, then the condition is satisfied. In conclusion, under the typical T, r, a values, the WDE in the 'hot' region is removed when h of 62 and SI of 32.

In the 'normal' region, WDE can also be prevented if SI is sufficiently small. The condition that the WDEs be completely eliminated is analyzed. Assuming the ideal HAD behavior, all cold addresses are accessed less frequently than hot addresses. This means that if all the hot addresses are accessed by T, then all cold addresses are accessed less frequently than T, and thus no WDE occurred. Based on this observation, the following condition is derived.

$$h \cdot T \geq r \cdot c \cdot SI \tag{4}$$

The left side of the condition is the total number of writes when all units in the 'hot' region are written by T. The right side is the number of writes required until the cold unit is moved. Therefore, when the left side is larger than the right side, the WDE completely disappears. For example, assume a T value of 1K, h of 62, and r and c of 64 and 32. In this scenario, WDE is eliminated when SI is 32. The relationship between the various SI values and the WDE frequency is analyzed in Section IV-B.

F. OVERHEAD OF WL-WD IMPLEMENTATION

To provide feasible implementation of WL-WD, a whole partition is separated into sub-partitions, as shown in Figure 3. Each sub-partition has a separate HAD, mapping table, and free queue. In Section III-E, the number of hot units for each sub-partition is set to 62. Considering that the latency of the SRAM is about 1 ns [27], the worst latency overhead incurred by searching the mapping table is about 62 ns. The typical latency of the PCM is about 200 ns [5], meaning that the worst case scenario gives about 30% decrease in system performance. However, it can be acceptable because the worst case occurs occasionally. Moreover, some degree of hardware parallelism could partially mitigate this overhead. In other words, instead of searching the mapping table entries one by one, searching multiple entries in parallel could greatly reduce the latency. The IPC degradation considering the SRAM latency is presented in Section IV-B.

Hardware overhead is analyzed in this section. The maximum number of entries of the mapping table and the free queue is about 256KB. The size of an entry is about 3B in total, resulting in a total of 768 KB ($= 256K \cdot 3B$) per partition. For the 8-GB PCM rank, which consists of 8 partitions, the total hardware overhead is about 6.4 MB, which takes only 0.08% of the rank size.

IV. EXPERIMENTAL RESULTS

A. EXPERIMENTAL METHODOLOGY

To evaluate the proposed algorithm, a system with a 4-core CPU and PCM main memory is simulated. We use two separate simulators for each CPU and PCM. First, the CPU is simulated using the gem5 simulator [28]. The main memory trace file is extracted from this simulator. Second, the NVMain simulator [29] is used to configure the PCM module, and the extracted trace file from gem5 is simulated according to the configured PCM structure. Configurations used in each simulator are shown in Table 1. In gem5, the CPU has 4-cores with 4-GHz operating frequency. The L1 cache is i/d separate, 32 KB, and 8-way set-associative cache, having a 64B cache line size. The L2 cache is 256 KB, 16-way, and also has a line size of 64B. For nvmain, the PCM size is set to 8GB, and the structure consists of 1 channel, 1 rank, 8 banks, and 128 tiles per bank. The tile size is set to $4K \times 2K$. PCM latencies are set to 100 and 200 ns for read and write operations, respectively. It should be noted that it is possible to cover various advanced structures such as 8 cores/2 channels and 16 cores/4 channels with the simulation results of the 4-cores/1-channel setting in this paper because the use of multiple cores causes an increase in both the memory access and the channel, and eventually the memory access per channel is maintained at a similar level through channel interleaving. For WDE modeling, counter-based modeling is used, and referring to the threshold value reported in SIWC [30] (i.e., 5K-10K), the threshold value T

TABLE 1. Basic configurations of gem5 and nvmain simulators.

gem5 confs.	CPU	4-core, out-of-order, 4GHz
	L1 cache	I/D separate, 32KB, 8-way
	L2 cache	256KB, 16-way
nvmain confs.	PCM size	8GB, 1 ch, 1 rank, 8 banks
	latencies	100 ns read, 200 ns write
	WDE model	counter based, T=1K

TABLE 2. Selection in SPEC2006 workloads for each mixed workload.

	mix1	mix2	mix3	mix4	mix5	mix6	mix7
mcf					○	○	○
lbm	○	○			○		○
leslie3d	○	○	○				○
astar	○	○	○	○			
bzip2		○	○	○	○		
gcc	○		○	○	○	○	
GemsFDTD				○		○	○
povray						○	

is set to a fairly conservative value of $1K$ to cover even the situation in which write disturbance becomes more severe as the technology node shrinks in the future.

To show the practicality of the proposed method, mixed workloads of SPEC CPU 2006 are evaluated, as in [31]. Since the average value of L2/L3 misses per kilo instructions (MPKI) in SPEC 2016 is set higher than that in SPEC 2017 [32], SPEC 2006 is more suitable in situations like this paper where the pressure on the main memory needs to be measured. Table 2 shows the benchmark selection for each mixed workload. We map each benchmark to each core in gem5; thus, four benchmarks are mixed for each workload. Additionally, mixed workloads are created to configure the harshest environment that the PCM can experience. In other words, we select workloads with high MPKIs among the SPEC 2006 benchmark and make these workloads operate on each of the four cores, resulting in very high traffic to the main memory. As a result, a synthetic workload is generated by loading a different workload for each core.

The WL-WD is evaluated under various configurations using the above simulators and workloads. The configuration that shows the best results is compared with other schemes. Because the proposed algorithm addresses both WDE and wear-leveling issues, the comparison schemes include both WDE mitigating schemes and wear-leveling algorithms. Section IV-B presents the evaluation results of the proposed algorithm, and Section IV-C shows the comparison results with other schemes.

B. WL-WD EVALUATION

This section evaluates the performance of the proposed algorithm under various parameter configurations. The parameters of the algorithm are r and c (the number of rows and columns of sub-partition, respectively), a ('hot' region column number), and SI (slide interval). The values of r and c are set to 64 and 32, respectively, and a is set to 1. A spare region of about 3.12% ($a/c \times 100\%$) is used. For the SI values, we use four values, i.e., 32, 128, 256, and 512. It is noteworthy that the value 32 completely eliminates the WDE, as discussed in Section III-E. This means that further reducing the SI value (i.e., 2, 4, 8, and 16) only causes the addition of unnecessary swap requests, and therefore, we only proceed with the additional experiment under these conditions (i.e., SI of 128, 256, and 512) to evaluate the performance in larger SIs.

The wear-leveling performance of the proposed algorithm is given first. The normalized lifetime (NL) is used as a metric of the wear-leveling performance, which is obtained as follows:

$$NL = \frac{(total\ writes)}{(worst\ WC) \cdot (total\ address\ number)}, \tag{5}$$

where the worst WC indicates the worst write count of all units. The total number of writes when the simulation finishes is used in the numerator. The denominator represents the number of writes in the ideal situation when all cells are uniformly written as the worst WC. Therefore, the NL indicates the actual number of writes compared to that of the ideal scenario. The higher NL value denotes the better wear-leveling performance.

Figure 9 shows the NL values for seven mixed workloads. The baseline configuration does not apply to the proposed algorithm, and the remaining four bar graphs show the NL values when SI is 32, 128, 256, and 512. First, it is observed that the NL values are significantly increased by applying the WL-WD scheme compared with the baseline, regardless of the SI value. In particular, for mix 1, mix 6, and mix 7, where the NL of the baseline is relatively high, the NL value becomes at most 0.9 when the WL-WD scheme is applied. The tendency of the NL values with respect to the SI value is also analyzed. As the SI value decreases, the slide operation is done more frequently, resulting in a better wear-leveling performance. This relationship can be confirmed in the bar graphs. In almost all workloads, the value of NL decreases as the SI increases. It should be noted that at some points, such as SI=128 and SI=256 in mix 4, this tendency is reversed. This is because a smaller SI leads to more frequent slides, thereby incurring more additional writes. Therefore, when the effect of these additional writes overwhelms the positive effect of slides, the NL may be reversed.

The WDE occurrence is presented next. Figure 10 shows the WDE occurrence under various SIs as a ratio to the baseline. When the SI value is 32, the WDE ratio converges to almost zero (the numbers above the graphs represent the actual WDE occurrence). Because the condition in (2) is satisfied, WDE does not occur theoretically in this case. However, the operation of the HAD is not ideal, thus incurring few WDEs. Nevertheless, the number is very small (0.1% less compared to the baseline), and they can therefore be handled by error correcting codes or memory scrubbing schemes in PCM [33]. As the SI increases, the WDE ratio also increases.

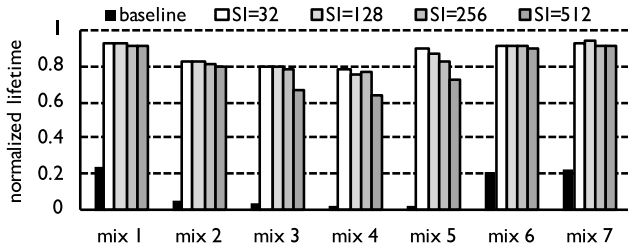


FIGURE 9. Normalized lifetime of the baseline and the WL-WD.

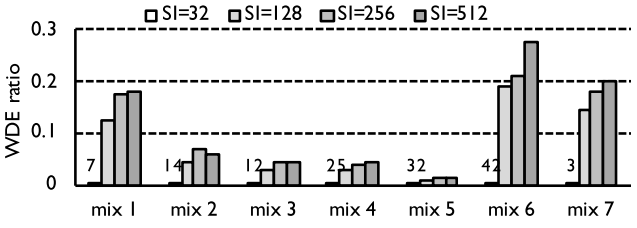


FIGURE 10. WDE ratio of the various configurations of WL-WD.

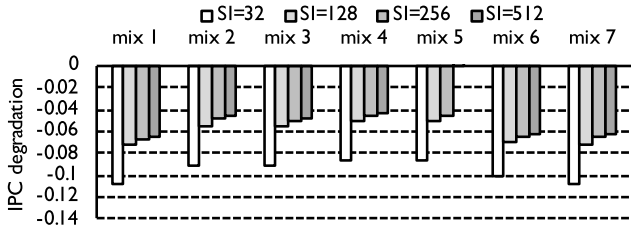


FIGURE 11. IPC degradation of the various configurations of WL-WD.

However, in most situations, the WDE ratio is less than 0.2, which means that WL-WD reduces the WDE by more than 80%.

The number of instructions per cycle (IPC) is also evaluated in Figure 11. In this study, we define IPC as the ratio of the number of instructions extracted from gem5 and the number of executed cycles in NVMain. The proposed WL-WD worsens IPC by two factors: 1) the latency increases owing to the HAD, the mapping table, and the free queues, 2) the additional writes by the slide operation. The IPC results in Figure 11 show that the IPC degradation can be mitigated as the SI increases. This is because the second factor (i.e., the additional writes) occurs less frequently at the higher SI value. The IPC is decreased by about 8-10% when the SI is 32, whereas the IPC is decreased only by 4-6% when SI is 512.

Finally, we provide the power/energy analysis according to the SI values. The unit access energy of PRAM is 1J/GB and 6J/GB for read and write operations, respectively [34]. The total power consumption can be calculated by multiplying the total number of read/write traffic by this unit energy. Table 3 shows the power consumption of WL-WD according to each SI value. The power consumption of baseline configuration is calculated as 2.145W, and when SI is 32, 128, 256, and 512, the power consumption is increased by 9.0%, 2.2%, 1.3%, and 0.8%, respectively. In detail, when SI is 32, which consumes the most power, about 0.19W of an additional power consumption occurs. However, considering that the thermal design power (TDP) of modern CPU chips is typically 85W, additional 0.19W of power consumption takes only 0.22% of the TDP, and therefore, it can be concluded that the application of WL-WD does not cause a major problem in the overall system in terms of power consumption.

TABLE 3. Power analysis on various configurations of WL-WD.

Power(W)	mix1	mix2	mix3	mix4	mix5	mix6	mix7	Average
Baseline	2.163	2.217	2.23	2.115	2.184	1.989	2.114	2.145
SI=32	2.359 (+9.1%)	2.427 (+9.5%)	2.444 (+9.1%)	2.3 (+9.1%)	2.374 (+9.1%)	2.158 (+9.1%)	2.303 (+9.1%)	2.338 (+9.1%)
SI=128	2.21 (+9.1%)	2.225 (+9.1%)	2.291 (+9.1%)	2.156 (+9.1%)	2.228 (+9.1%)	2.028 (+9.1%)	2.157 (+9.1%)	2.192 (+9.1%)
SI=256	2.189 (+9.1%)	2.223 (+9.1%)	2.269 (+9.1%)	2.136 (+9.1%)	2.209 (+9.1%)	2.01 (+9.1%)	2.136 (+9.1%)	2.172 (+9.1%)
SI=512	2.179 (+9.1%)	2.222 (+9.1%)	2.258 (+9.1%)	2.126 (+9.1%)	2.198 (+9.1%)	2.001 (+9.1%)	2.126 (+9.1%)	2.162 (+9.1%)

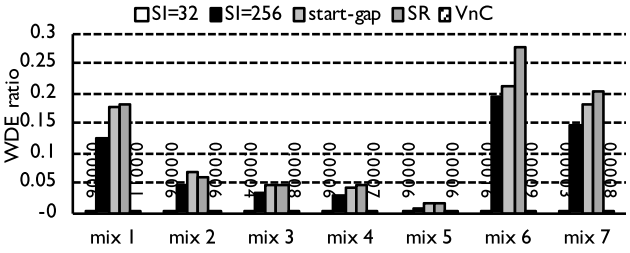


FIGURE 12. WDE ratio of the WL-WD and other schemes.

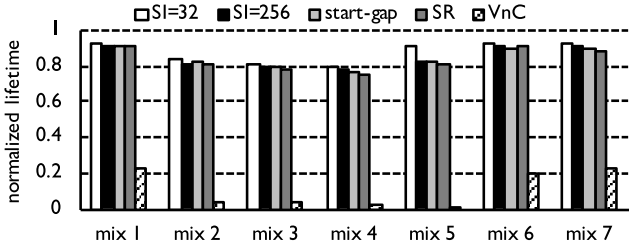


FIGURE 13. Normalized lifetime of the WL-WD and other schemes.

In summary, WL-WD has a trade-off relationship between the wear-leveling performance, WDE reduction, and IPC. Using a small SI gives better wear-leveling performance and WDE reduction, but leads to a lower IPC. On the other hand, a large SI gives a relatively lower wear-leveling and WDE performance, but provides a higher IPC. Therefore, it is possible to adjust the SI value according to the characteristics of the memory system. If the system has other reliability handling scheme such as ECC, the large SI value is selected to increase the IPC. On the other hand, if the system does not have such schemes, the small SI value is selected to enhance the endurance and reliability.

C. COMPARISON RESULTS

The proposed WL-WD schemes with two SI values, 32 and 256, are compared with other wear-leveling and

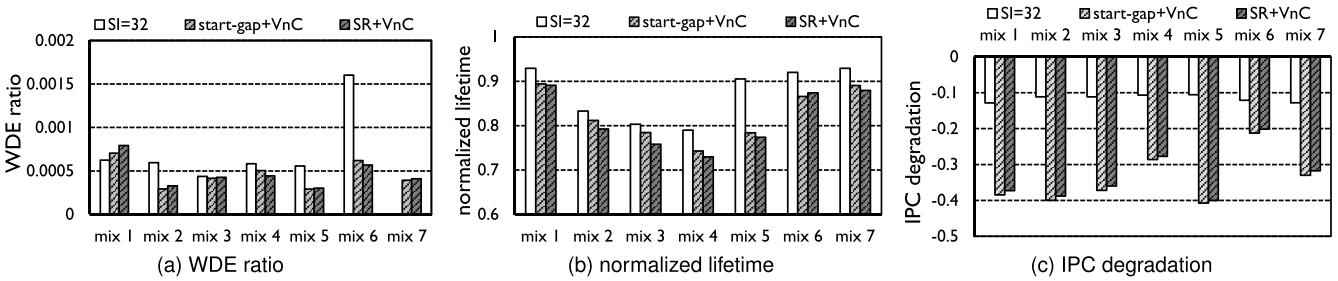


FIGURE 14. Comparison results between WL-WD and other schemes.

WDE-mitigating schemes, Start-gap [17], security refresh (SR) [18], and Verify-n-Correct (VnC) [5]. The WDE ratio with respect to the baseline is presented first. This ratio is obtained by dividing WDE occurrence under the applied scheme by that of the baseline. The smaller ratio indicates that the WDEs occur less frequently. In Figure 12, both the start-gap and SR (i.e., wear-leveling schemes) show relatively high WDE ratios that range from 0.4 to 0.7, which are considerably higher than those of the WL-WD scheme. However, WDE-mitigating scheme, VnC, almost eliminates the WDE, similar to the WL-WD scheme with the SI value of 32. On the other hand, in Figure 13, both the start-gap and SR show similar NL to the WL-WD scheme, while VnC alone shows significantly lower NL. This means that the existing wear-leveling algorithms alone cannot address the WDE issue, and the existing WDE-mitigating algorithms alone cannot prevent the endurance issue at all. Therefore, in case of the existing solutions, the wear-leveling algorithm and WDE-mitigating scheme must be used together to address the both critical issues in the PCM.

Next, since the existing wear-leveling and WDE-mitigating schemes must be used together to address the both critical issues in the PCM, combinations of previous studies are used for comparison. Figure 14 shows the comparison results of WL-WD, start-gap [17] + VnC [5], and SR [18] + VnC [5]. Figure 14 (a) shows the value of WDE ratios. As shown in Section IV-B, the WL-WD scheme with SI=32 almost eliminates the WDE. It should be noted that the scale of the y-axis in Figure 14 (a) is very small compared to Figure 12. The reason that the WDEs are not completely eliminated is that HAD used in the experiment is not ideal. However, the WDE ratio when SI is 32 is comparable with that of start-gap+VnC and SR+VnC cases. It should be noted that as shown in Figure 12, the start-gap and SR show a ratio of 0.2 or more in the specific workload (i.e., mix 6) if not combined with VnC. However, when start-gap and SR are used together with VnC, the WDE ratio also converges to zero. This means that both the WL-WD and start-gap+VnC, SR+VnC effectively mitigates the WDE. Figure 14 (b) shows the value of NL. For all workloads, the WL-WD scheme with the SI value of 32 gives the highest NL value for all workloads. When VnC is added to start-gap and SR, the NL is additionally decreased by about 2% because VnC incurs additional writes. As a result, the proposed WL-WD scheme shows NL results that are more than 10% superior to the combination of existing studies in a specific workload. Figure 14 (c) shows the value of IPC. When VnC is added, IPC degradations range from -0.2 to -0.4. These values are significantly worse than those of the WL-WD. The IPC degradation of the WL-WD ranges from -0.10 to -0.12. This indicates that the proposed WL-WD scheme has the least IPC degradation caused to resolve endurance and reliability issues compared to the combination of existing studies.

The comparison results are summarized as follows.

- The proposed method can flexibly adjust the SI value to provide an optimal endurance and reliability solution according to the characteristics of the memory system.
- Previous PCM wear-leveling schemes cannot address the WDE issue: They reduce WDE by only about 30-50%. Therefore, WDE-mitigating schemes such as VnC must be used together.
- Compared to the combination of existing wear-leveling and WDE mitigation schemes, the proposed WL-WD exhibits a better performance in terms of NL and IPC. The NL is improved by at least 3%, and IPC increases about 10-25%. This is because the WL-WD excludes the deterioration of NL and IPC due to additional reads and writes incurred by VnC.

V. DISCUSSION: APPLICABILITY OF WL-WD

The proposed scheme can be easily extended to other types of NVMs, such as ReRAM and STT-MRAM. WDEs also exist in both ReRAM and STT-MRAM. WDEs of these memories originate from the increment of the programming voltage that is used to reduce the access latency. Such voltage increments pull up the biased voltage of unselected neighboring cells, leading to unexpected programming on neighbors (i.e., WDE) [35]. A naive approach to reducing WDEs is to apply lifetime extension methods to these memories, and consequently, researches aimed at extending the lifetime in various NVMs have been well studied [36]–[40]. In [40], the lifetime of NVM is extended by incorporating ECC to recover stuck-at-fault errors. In [37], the redundant refresh operations are reduced with the refresh-aware cache replacement policy, enhancing the endurance of the STT-RAM cache.

In [39], an encoding scheme is proposed for reducing the frequency of bitflips in NVM-based caches. In [38], contents in NVM-based instruction and data caches are uniformly and periodically interchanged with the dedicated cache controller. Although these schemes may mitigate WDEs, these schemes do not specifically consider the characteristics and vulnerability patterns of WDEs. Moreover, previous studies leave the state of the slightly disturbed (but not flipped) cells unrestored. It is noteworthy that a previous study explains that WDEs of other NVMs also result from accumulative write operations [35]; that is, the WDE threshold (T) in the counter-based modeling (see Section II-B) is reasonable for these NVMs. In conclusion, the proposed scheme can be applied to either ReRAM or STT-MRAM by modulating the slide interval (SI), according to the WDE threshold.

VI. CONCLUSION

To use PCM as the main memory, both endurance and reliability issues must be addressed. However, existing wear-leveling algorithms, which address endurance issues, are vulnerable to WDEs. Therefore, a WDE mitigating scheme must be used together with the wear-leveling algorithm, incurring the additional performance overhead. The proposed WL-WD is a wear-leveling algorithm that mitigates WDEs. The WL-WD uses active and passive wear-leveling adaptively, achieving better wear-leveling performance with reasonable hardware overhead. The simulation results show WL-WD can achieve better performance than previous schemes in terms of normalized lifetime and IPC. It is noteworthy that the performance of WL-WD can be adjusted according to the setting of various SI values. As a result, WL-WD can support operations suitable for the characteristics of the memory system.

REFERENCES

[1] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proc. 36th Annu. Int. Symp. Comput. Archit (ISCA)*, New York, NY, USA, 2009, pp. 24–33, doi: 10.1145/1555754.1555760.

[2] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2009, pp. 14–23, doi: 10.1145/1555754.1555759.

[3] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2009, pp. 2–13, doi: 10.1145/1555754.1555758.

[4] H. Lee, M. Kim, H. Kim, H. Kim, and H.-J. Lee, "Integration and boost of a read-modify-write module in phase change memory system," *IEEE Trans. Comput.*, vol. 68, no. 12, pp. 1772–1784, Dec. 2019.

[5] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proc. IEEE*, vol. 98, no. 12, pp. 2201–2227, Dec. 2010.

[6] D. Ielmini, D. Sharma, S. Lavizzari, and A. L. Lacaita, "Reliability impact of chalcogenide-structure relaxation in phase-change memory (PCM) cells—Part I: Experimental study," *IEEE Trans. Electron Devices*, vol. 56, no. 5, pp. 1070–1077, May 2009.

[7] W. Jin, S. Lu, and X. Cai, "ESD-PCM: Constructing reliable super dense phase change memory under write disturbance," in *Proc. IEEE Eur. Test Symp. (ETS)*, May 2021, pp. 1–6.

[8] J. Choi, J. Jang, and L.-S. Kim, "DC-PCM: Mitigating PCM write disturbance with low performance overhead by using detection cells," *IEEE Trans. Comput.*, vol. 68, no. 12, pp. 1741–1754, Dec. 2019.

[9] S. H. Lee *et al.*, "Programming disturbance and cell scaling in phase change memory: For up to 16 nm based 4F2 cell," in *Proc. Symp. VLSI Technol.*, Jun. 2010, pp. 199–200.

[10] S. J. Ahn, Y. Song, H. Jeong, B. Kim, Y.-S. Kang, D.-H. Ahn, Y. Kwon, S. W. Nam, G. Jeong, H. Kang, and C. Chung, "Reliability perspectives for high density PRAM manufacturing," in *IEDM Tech. Dig.*, Dec. 2011, pp. 6–12.

[11] B. Kim, Y. Song, S. Ahn, Y. Kang, H. Jeong, D. Ahn, S. Nam, G. Jeong, and C. Chung, "Current status and future prospect of phase change memory," in *Proc. 9th IEEE Int. Conf. ASIC*, Oct. 2011, pp. 279–282.

[12] H. Lee, H. Jung, H.-J. Lee, and H. Kim, "Bit-width reduction in write counters for wear leveling in a phase-change memory system," *IEIE Trans. Smart Process. Comput.*, vol. 9, no. 5, pp. 413–419, Oct. 2020.

[13] L. Jiang, Y. Zhang, and J. Yang, "Mitigating write disturbance in super-dense phase change memories," in *Proc. IEEE/IFIP 44th Annu. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2014, pp. 216–227.

[14] S. Swami and K. Mohanram, "ADAM: Architecture for write disturbance mitigation in scaled phase change memory," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1235–1240.

[15] R. Wang, S. Mittal, Y. Zhang, and J. Yang, "Decongest: Accelerating super-dense PCM under write disturbance by hot page remapping," *IEEE Comput. Archit. Lett.*, vol. 16, no. 2, pp. 107–110, Jul. 2017.

[16] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "SD-PCM: Constructing reliable super dense phase change memory under write disturbance," in *Proc. 20th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, New York, NY, USA, 2015, pp. 19–31, doi: 10.1145/2694344.2694352.

[17] 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 (Micro)*, Dec. 2009, pp. 14–23.

[18] 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," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, New York, NY, USA, 2010, pp. 383–394, doi: 10.1145/1815961.1816014.

[19] H. Lee, H. Jung, H.-J. Lee, and H. Kim, "Bit-width reduction in write counters for wear leveling in a phase-change memory system," *IEIE Trans. Smart Process. Comput.*, vol. 9, no. 5, pp. 413–419, Oct. 2020.

[20] Y. Choi *et al.*, "A 20 nm 1.8 V 8 Gb PRAM with 40 MB/s program bandwidth," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2012, pp. 46–48.

[21] U. Russo, D. Ielmini, A. Redaelli, and A. L. Lacaita, "Modeling of programming and read performance in phase-change memories—Part II: Program disturb and mixed-scaling approach," *IEEE Trans. Electron Devices*, vol. 55, no. 2, pp. 515–522, Feb. 2008.

[22] J. Jang, W. Shin, J. Choi, Y. Kim, and L.-S. Kim, "Sparse-insertion write cache to mitigate write disturbance errors in phase change memory," *IEEE Trans. Comput.*, vol. 68, no. 5, pp. 752–764, May 2019.

[23] M. Murugan and D. H. C. Du, "Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead," in *Proc. IEEE 27th Symp. Mass Storage Syst. Technol. (MSST)*, May 2011, pp. 1–12.

[24] S. Kim, H. Jung, W. Shin, H. Lee, and H.-J. Lee, "HAD-TWL: Hot address detection-based wear leveling for phase-change memory systems with low latency," *IEEE Comput. Archit. Lett.*, vol. 18, no. 2, pp. 107–110, Jul. 2019.

[25] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," in *Proc. 37th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA, 2010, pp. 141–152, doi: 10.1145/1815961.1815980.

[26] M. K. Qureshi, "Pay-as-you-go: Low-overhead hard-error correction for phase change memories," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2011, pp. 318–328.

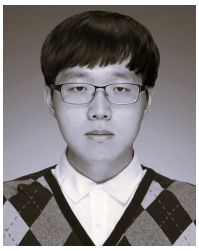
[27] M. Imani, S. Patil, and T. Rosing, "Low power data-aware STT-RAM based hybrid cache architecture," in *Proc. 17th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2016, pp. 88–94.

[28] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011, doi: 10.1145/2024716.2024718.

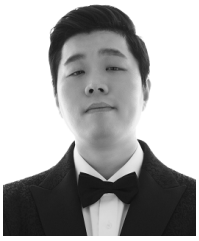
[29] M. Poremba, T. Zhang, and Y. Xie, "NVMain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE Comput. Archit. Lett.*, vol. 14, no. 2, pp. 140–143, Jul. 2015.

[30] J. Jang, W. Shin, J. Choi, Y. Kim, and L.-S. Kim, "Sparse-insertion write cache to mitigate write disturbance errors in phase change memory," *IEEE Trans. Comput.*, vol. 68, no. 5, pp. 752–764, May 2019.
[31] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006, doi: 10.1145/1186736.1186737.
[32] A. Navarro-Torres, J. Alastruey-Benedé, P. Ibáñez-Marín, and V. Vinals-Yufera, "Memory hierarchy characterization of SPEC CPU2006 and SPEC CPU2017 on the Intel Xeon Skylake-SP," *PLoS ONE*, vol. 14, no. 8, pp. 1–24, 2019.
[33] M. Kim, J. Lee, H. Kim, and H.-J. Lee, "An optimal on-demand scrubbing solution for read disturbance errors in phase-change memory," *IEIE Trans. Smart Process. Comput.*, vol. 10, no. 1, pp. 55–60, Feb. 2021.
[34] Y. Liu, C. Zhou, and X. Cheng, "Hybrid SSD with PCM," in *Proc. 11th Annu. Non-Volatile Memory Technol. Symp.*, Nov. 2011, pp. 1–5.
[35] Y. Zhang, D. Feng, W. Tong, Y. Hua, J. Liu, Z. Tan, C. Wang, B. Wu, Z. Li, and G. Xu, "CACF: A novel circuit architecture co-optimization framework for improving performance, reliability and energy of ReRAM-based main memory system," *ACM Trans. Archit. Code Optim.*, vol. 15, no. 2, pp. 1–26, May 2018, doi: 10.1145/3195799.
[36] W. J. Lee, C. H. Kim, and S. W. Kim, "A last-level cache management for enhancing endurance of phase change memory," in *Proc. 36th Int. Tech. Conf. Circuits/Syst., Comput. Commun. (ITC-CSCC)*, Jun. 2021, pp. 1–4.
[37] P. Saraf and M. Mutyam, "Endurance enhancement of write-optimized STT-RAM caches," in *Proc. Int. Symp. Memory Syst. (MEMSYS)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 101–113, doi: 10.1145/3357526.3357538.
[38] H. Farbeh, A. M. H. Monazzah, E. Aliagha, and E. Cheshmikhani, "A-CACHE: Alternating cache allocation to conduct higher endurance in NVM-based caches," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 7, pp. 1237–1241, Jul. 2019.
[39] S. Asadi, A. M. H. Monazzah, H. Farbeh, and S. G. Miremadi, "WIPE: Wearout informed pattern elimination to improve the endurance of NVM-based caches," in *Proc. 22nd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2017, pp. 188–193.
[40] H. Farbeh, H. Kim, S. G. Miremadi, and S. Kim, "Floating-ECC: Dynamic repositioning of error correcting code bits for extending the lifetime of STT-RAM caches," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3661–3675, Dec. 2016.

MOONSOO KIM received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2014 and 2020, respectively. In 2020, he joined the SoC Design Team at Samsung Electronics, Hwasung, South Korea, where he is currently working as a Staff Engineer. His research interests include cache/memory architecture and SoC design.



HYOKEUN LEE (Member, IEEE) received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2016 and 2021, respectively. He is currently working as a Postdoctoral Researcher with the Inter-University Semiconductor Center (ISRC), Seoul National University. His current research interests include non-volatile memory controller design, architecture simulation modeling, and computer architecture.



HYUN KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2009, 2011, and 2015, respectively. From 2015 to 2018, he was a BK Assistant Professor with the BK21 Creative Research Engineer Development for IT, Seoul National University. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, where he is currently an Assistant Professor. His current research interests include algorithms, computer architecture, memory, and SoC design for low-complexity multimedia applications, and deep neural networks.

HYUK-JAE LEE (Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1987 and 1989, respectively, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1996. From 1998 to 2001, he was a Senior Component Design Engineer with the Server and Workstation Chipset Division, Intel Corporation, Hillsboro, OR. From 1996 to 1998, he was a Faculty Member with the Department of Computer Science, Louisiana Tech University, Ruston, LA. In 2001, he joined the School of Electrical Engineering and Computer Science, Seoul National University, where he is currently a Professor. He is the Founder of Mamurian Design, Inc., Seoul, a fabless SoC design house for multimedia applications. His current research interests include computer architecture and SoC for multimedia applications.