Improving Block Management in 3D NAND Flash SSDs with Sub-Block First Write Sequencing

Matchima Buddhanoy
Colorado State University
Fort Collins, CO, USA
matchima.buddhanoy@colostate.edu

Kamil Khan
Colorado State University
Fort Collins, CO, USA
kamil@colostate.edu

Aleksandar Milenkovic
University of Alabama in Huntsville
Huntsville, AL, USA
milenka@uah.edu

Sudeep Pasricha
Colorado State University
Fort Collins, CO, USA
sudeep@colostate.edu

Biswajit Ray
Colorado State University
Fort Collins, CO, USA
biswajit.ray@colostate.edu

ABSTRACT

Continual vertical scaling in 3D NAND flash solid-state drives (SSDs) results in larger memory blocks, causing performance degradation due to big-block management issues. Pages within a 3D NAND flash block are traditionally written using layer first write sequencing. This paper introduces and explores the benefits of an alternative sub-block first write sequence. This method when coupled with sub-block erase operations promises to alleviate the big-block problem. Our evaluation on a commercial 32-layer 3D NAND flash SSD chip shows that though the proposed method increases the raw bit error rate (RBER), it remains below the threshold that can be corrected by error correction codes (ECCs). Simulation analysis further shows that our proposed method reduces garbage collection overhead, resulting in 36.0% lower response time and 9.6% reduction in additional writes due to garbage collection compared to traditional 3D NAND flash SSDs.

CCS CONCEPTS

• **Hardware → External storage**.

KEYWORDS

Flash memories, sub-block write, SSD storage, garbage collection, write amplification

ACM Reference Format:
Matchima Buddhanoy, Kamil Khan, Aleksandar Milenkovic, Sudeep Pasricha, and Biswajit Ray. 2024. Improving Block Management in 3D NAND Flash SSDs with Sub-Block First Write Sequencing. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3649476.3658803

1 INTRODUCTION

3D NAND flash memories are indispensable in the design of SSDs used in storage systems across a wide range of computing domains, including IoT platforms, smartphones, workstations, and servers in cloud computing platforms. The bit density continues to increase with every new generation of 3D NAND memories, driven by an increase in the number of vertical layers (3D scaling) and logical scaling that allows for storing 2, 3, or 4 bits of data per single memory cell. As a result of these trends, the number of pages in a memory block has continued to increase, causing a so-called big block problem in the management of flash SSD media [4, 9, 12].

The big block problem occurs during garbage collection (GC) when valid data on live pages must be copied from a victim block (the block selected for erasing) to another free block. Larger blocks mean more live pages to copy, resulting in increasing GC latency in the Flash Translation Layer (FTL). The increase in GC latency degrades performance and increases write amplification, negatively impacting flash memory lifetime and capacity [3, 15, 17].

To cope with the big-block problem, a sub-block erase mechanism has been proposed by industry [6, 7, 14, 16]. With this mechanism, each flash block is partitioned into multiple sub-blocks, where each sub-block can be erased independently from other sub-blocks within the block. Additional circuitry is required to electrically isolate sub-blocks during a sub-block erase operation. Several recent research efforts in academia focus on developing GC algorithms that take advantage of the sub-block erase mechanism [2, 4, 9, 12] and demonstrate that sub-block erase can significantly reduce GC latency. However, they focus solely on GC algorithms without addressing sub-block erase implementation challenges.

Generally, two challenges remain with the sub-block erase mechanism in 3D NAND flash SSDs. The first problem is that the existing approaches focus on sub-block erase operations performed on logical sub-blocks (see Section 2.2 for a detailed discussion) that cause disturbances in values stored in pages residing in boundary layers of the victim block. To address this problem, either hardware or software isolation is required [2, 9], limiting the potential gains. The second problem is related to sub-block erase operations performed on physical sub-blocks. Physical sub-blocks span multiple layers in 3D NAND flash, whereas a conventional page write sequence, called layer first write sequence (LFWS), assumes that pages are written layer-by-layer, i.e., we first write all pages in a single layer before moving to writing pages in the next layer. As requests from the host processor for data writes and updates exhibit a strong spatial locality, locating empty physical sub-blocks ready for an erase operation can be challenging.

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658803

To address these challenges, we propose a page writing sequence called sub-block first write sequence (SFWS). This mechanism is amenable to physical sub-blocks in 3D NAND flash SSDs, thus improving robustness as physical sub-block erase operations are less susceptible to program disturbances. In addition, the SFWS improves data placement, thus minimizing the number of live pages that need to be copied before a sub-block erase operation, and maximizing the number of sub-blocks that can be erased concurrently.

In summary, the novel contributions of our work are:

- We propose a new page writing sequence approach for 3D NAND flash SSDs called SFWS.
- We conduct experimental analysis on a commercial off-the-shelf (COTS) 32-layer 3D NAND chip that supports multi-level-cell (MLC) storage to contrast our SFWS approach with the traditional LFWS. Our evaluation shows that SFWS results in a slightly increased RBER relative to the LFWS. However, RBER is much below the threshold that can be corrected by ECCs.
- We show how SFWS reduces GC overhead by quantifying improvements in response time and GC write overhead across SSD workloads.

2 BACKGROUND

2.1 Fundamentals of SSD Flash Storage

Flash-based SSD storage systems consist of one or more flash chips. Figure 1(a) illustrates the overall organization of a 3D NAND flash chip. A single flash chip includes a NAND die that contains control logic and one or more planes. A plane contains a certain number of memory blocks that share the same set of bit lines (BLs). A single block contains multiple flash pages, and each flash page contains multiple flash cells. A flash memory cell is essentially a Metal Oxide Semiconductor Field Effect Transistor (MOSFET) with an additional floating gate (FG) or charge trap (CT) layer in the gate oxide stack. A traditional flash cell stores 1 bit of data known as SLC (Single-Level Cell). Advances in controlling the amount of charge on the FG/CT layer and in sensing the charge during a read operation allowed for logical scaling, further increasing bit density. Thus, modern flash can store 2 bits (MLC), 3 bits (TLC – Triple-Level Cell), or 4 bits (QLC – Quad-Level Cell) per cell.

To achieve high bit density, 3D NAND flash is designed to vertically stack multiple layers of flash cells on top of each other [8]. Figure 1(b) illustrates the physical organization of a 3D NAND flash block, where green planes represent word lines (WLs) or the stack layers. In general, a 3D NAND block is physically divided into several sub-blocks to optimize the die area utilization (as all physical sub-blocks can share the same WLs). Note that our sample in this work has 16 sub-blocks (S_0-S_{15}), as shown in Figure 1(b). All the purple poly-silicon pillars along the same y-axis form one sub-block. All pillars within a sub-block connect to the substrate when the corresponding source select gates (SSGs) are active and to individual BLs when the corresponding drain select gates (DSGs) are active. Sub-blocks are identical to each other in terms of structure and organization. All sub-blocks share the same WLs and BLs. Select transistors are used to select a particular sub-block during page read and write operations. However, all the sub-blocks within a block are simultaneously erased using a block erase operation.

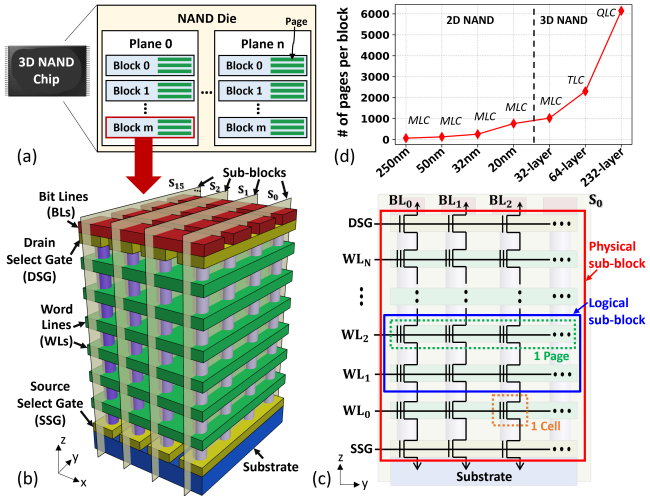

Figure 1: (a) Overall organization of a 3D NAND flash chip. (b) Schematic of a 3D NAND flash block. (c) Distinction between a logical and a physical sub-block structure. (d) Evolution of number of pages per block in NAND technology.

The circuit schematic of a physical sub-block is shown in Figure 1(c). Each sub-block contains multiple flash pages, where the page size is generally 16 kB in state-of-the-art 3D NAND flash chips. All cells in the same row share the same WL. The cells in a vertical column connect to a metal BL via its DSG and to the ground via its SSG at the bottom. With each new generation of 3D NAND, there is a substantial increase in the physical block size, measured in terms of the number of pages per block, leading to the emergence of the big block problem, as shown in Figure 1(d).

2.2 Block Erase vs. Sub-Block Erase Operation

The traditional erase operation in NAND flash memory occurs at the physical block level. Erasing a block involves setting the substrate to a high erase voltage, grounding the WLs, and activating all select transistors to remove charge from the memory cells. A flash cell set to logic '0' by a program operation can only be reset to logic '1' by erasing the entire block. To update a page of data, the new content must be written into a new erased page, and the page containing obsolete data is marked as invalid. When most pages in a block are marked as invalid, the block erase operation is performed to make the block available for reuse. However, before the erase operation, all valid (or live) pages in the block need to be copied to a different physical block. The more valid pages that need to be copied, the higher the latency penalty of this step.

Erasing at the sub-block granularity is a promising approach to reduce the performance penalty associated with the big-block problem. Two flavors of the sub-block erase mechanisms are discussed in prior work. The first approach assumes that erasing sub-blocks corresponds to physical sub-blocks (illustrated in Figure 1(c)) [6, 7, 14]. This approach ensures that only a single physical sub-block is erased while the remaining sub-blocks within the same block are electrically isolated from the erase operation. This isolation is achieved by providing independent control over SSG transistors for each sub-block. The second approach assumes logical sub-blocks that may span multiple physical layers or WLs (illustrated in Figure 1(c))

[4, 9, 12]. The logical sub-block erase is executed by selectively controlling the layers. The layers to be erased are grounded, whereas the layers associated with valid content are left floating.

There are two significant challenges with practically realizing the logical sub-block erase approach: The first challenge is erase disturbances in neighboring or boundary layer pages. This problem can be addressed by either hardware isolation or software isolation [2, 9]. Hardware isolation requires that certain neighboring pages are not used, acting as buffers that absorb erase disturbances. The major drawback of this approach is that it reduces storage capacity. Software isolation assumes the relocation of live pages from neighboring or boundary layers, thus ensuring that disturbances do not impact valid pages. A major drawback of this approach is an increased overhead due to live page migration. The second challenge is reliability concerns with potential permanent damage to the inter-layer oxides. The closely spaced vertical memory layers can experience hard breakdown of the interlayer oxides if a large voltage difference is presented during logical sub-block erase [5]. The voltage on floating WLs can reach as high as 20V due to coupling effects with the erase voltage on the substrate. This substantial voltage difference not only causes disturbance but also creates permanent defects in the inter-layer oxides, limiting endurance.

Due to these two major challenges, logical sub-block erase is difficult to realize in practice. In contrast, physical sub-block erase has fewer reliability concerns [6, 7, 14]. The physical sub-block erase is performed by activating the appropriate select gate transistors which pass the high erase voltage from the substrate and BLs inside the poly-silicon channel of the selected physical sub-block only. All the WLs are kept grounded during the erase operation which ensures less voltage stress on the memory layers. Due to its better practicality, in this paper, we utilize the physical sub-block erase approach while addressing the big-block problem.

3 RELATED WORK: BIG BLOCK PROBLEM

The big block problem arises during GC because pages containing valid data (live pages) need to be copied from the victim block – the block selected for erasing – to another free block. Larger blocks result in a larger number of live pages that need to be copied from the victim block, thus increasing the latency of the GC. The increase in latency due to GC severely degrades the overall performance of flash memory. In addition to performance loss, the big block problem exacerbates write amplification that negatively affects the lifetime and/or capacity of flash memory [4, 9, 12, 15]. The write amplification is the ratio of the amount of data actually written to the flash media (including page writes caused by GC) and the amount of data written by the host.

To cope with the big-block problem, prior research mainly focuses on developing GC algorithms that take advantage of the logical sub-block erase mechanism. Chen et al. [4] introduced a method called performance booster strategy that tries to maximize the number of victim sub-blocks (those that can be erased concurrently) within a flash block while ensuring minimal overhead due to live-page copy operations. The proposed method includes three components: (i) a runtime victim finder that finds sub-blocks that require no live-copy page overhead; (ii) a sub-block victim packer that finds a victim block with the maximum number of sub-blocks

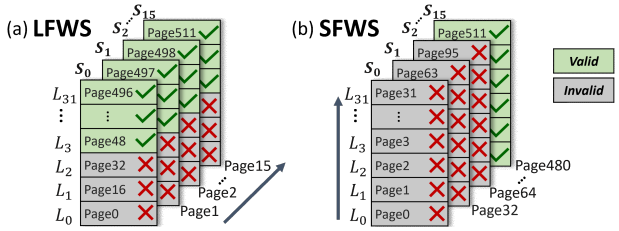

Figure 2: Comparison between (a) LFWS and (b) SFWS scheme. We assume the first 256 written pages are updated.

that can be reclaimed in one erase operation; and (iii) a free space cleaner that finds sub-blocks with a minimum number of live pages (in case the previous steps failed to find any sub-block for erasing). Their simulation-based evaluation shows that the proposed method can reduce the overhead caused by GC for up to 50% when compared to the conventional GC methods. Similarly, Liu et al. [12] propose a partial-erase – a software only mechanism that does not require any hardware modifications. They propose a custom GC algorithm that takes advantage of the proposed mechanism and shows that it can reduce the write latency by up to 48% relative to the baseline implementation. Finally, Gong et al. [9] propose a mechanism to further improve sub-block erase by placing frequently updated (hot) data into pages of blocks that are soon to be erased. This way they reduce the number of live pages that need to be copied to fresh blocks. Their simulation-based experimental analysis shows that their technique can reduce the GC latency by up to 65%.

All of these prior efforts focus on using the logical sub-block erase mechanism. However, practically realizing logical sub-block erase is very challenging, as discussed in Section 2.2. Unlike these prior efforts, we focus on the more practical physical sub-block erase mechanism and propose an approach to reduce GC overheads and write amplification, as discussed next.

4 PROPOSED SFWS METHOD

Writing flash pages in a 3D NAND flash block follows the traditional LFWS, as shown in Figure 2(a), to minimize RBER. LFWS ensures efficient pre-charge operation which is an essential step to minimize program disturbance. The pre-charge voltage is usually applied on the BLs which propagate through the top memory layers to the program layer. According to the LFWS, all pages within a given vertical layer need to be written before moving to the next layer ensuring uninterrupted propagation of pre-charge voltage. Instead of writing data in a layer-by-layer manner, we propose an alternative method of writing data in a sub-block-by-sub-block direction, called SFWS in Figure 2(b). Note that the arrow represents the writing direction. With SFWS, all flash pages within a lower-indexed sub-block are written before flash pages in a higher-indexed sub-block are written. SFWS when combined with the physical sub-block erase operation, allows treating each sub-block as an independent unit and hence it can reduce block size to the equivalent sub-block size, leading to the potential to overcome big-block management issues.

We illustrate the advantage of using SFWS in Figure 2(a)-(b). Let us consider a freshly erased 32-layer SLC flash block with 16 physical sub-blocks. In the traditional LFWS, the page addresses within a block are illustrated in Figure 2(a). Let us further assume that we receive a write request from the host processor to fill in

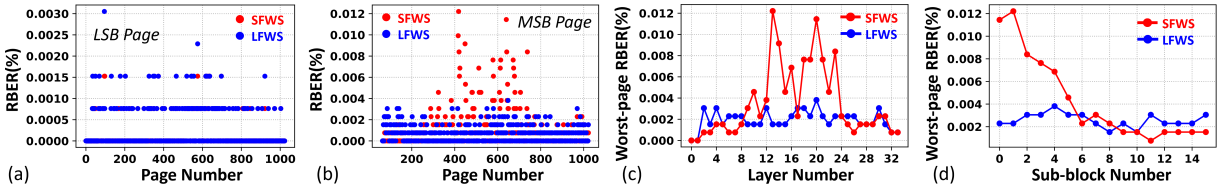

Figure 3: Comparison of RBER for SFWS and LFWS. (a) RBER for LSB pages. (b) Plot of MSB page BER vs. page number. (c) Plot of the worst-page RBER vs. layer number. (d) Plot of the worst-page RBER vs. sub-block number.

the entire block. The controller will issue sequential write requests incrementing page address after each page write, i.e., the page writing sequence is P0, P1, P2, … P511. Here all pages in layer 0 (L_0) are written before page writes are issued for pages in layer 1 (L_1). As requests from the host for data writes and updates exhibit a strong spatial locality, locating empty sub-blocks ready for erase presents a challenge. Let us assume that the first 256 pages are updated, resulting in the invalidation of pages P0-P255 in the flash block. In this case, half of the block contains invalid pages, yet every sub-block contains 16 valid pages, that need to be copied by the GC if this block is going to be recycled.

This practical limitation of the traditional LFWS approach when combined with physical sub-block erase motivates the use of our SFWS approach, even though it may increase RBER in the written page. SFWS takes advantage of physical sub-block erase operations as they are less susceptible to program disturbances, but also improves data placement to minimize the number of live pages that need to be copied before a sub-block erase. By changing the write sequence of data to SFWS, sub-blocks S_0-S_7 can be erased in a single sub-block erase operation with no additional live page copy operations, as shown in Figure 2(b).

5 EXPERIMENTAL EVALUATION

We evaluate our proposed SFWS approach using both raw NAND flash chips and workload simulation. Reliability analysis of SFWS and the corresponding simulation-based performance analysis are discussed in Section 5.1 and Section 5.2, respectively.

5.1 Reliability Analysis

The goal of our reliability analysis is to explore the effects of the proposed SFWS on RBER by comparing it to the traditional LFWS. The experimental evaluation is carried out on COTS 32-layer 3D MLC NAND flash chips from a major NAND vendor. Each chip has 2,192 blocks and each block contains 1,024 logical pages. Each logical page reserves 16 kB for user data and 2 kB for ECC. A custom-designed test board is used to interface the raw NAND chip with a workstation. This setup allows us to perform basic operations such as page read, page write, and block erase. Our software emulating FTL functions determines RBER for read operation. Our prior work [1] explained our test setup in detail. To evaluate the RBER of the traditional LFWS method, we first erase and write a full block with random data. After completing the program operations, we immediately read all 1,024 flash pages and calculate the RBER. Next, we erase the same block and write the same random data using the SFWS method. Once the entire block is written, we immediately read all pages in the same order and calculate the RBER. Finally, we analyze and compare the RBER of the two methods.

We evaluate the SFWS and LFWS approaches on a block configured in the MLC mode. Note that MLC memory contains shared memory pages - LSB (Least Significant Bit) and MSB (Most Significant Bit) pages share the same set of memory cells for a given WL. We first write all the logical pages of the MLC memory block sequentially which is the default page write sequence (i.e., LFWS). We compute the page RBER after writing all memory pages of the block. Then, we erase the memory block and re-write its logical pages following the SFWS method. We choose the appropriate page addresses (from the datasheet) to finish writing all the pages of a physical sub-block before moving to the next physical sub-block. We finish writing all pages of the block following the SFWS method, read the pages, and then compute the RBERs of individual pages.

The results of this experiment are shown in Figure 3(a)-(d). Figure 3(a)-(b) shows the RBER for the LSB and MSB pages, respectively. Blue and red dots represent the LFWS and SFWS methods, respectively. We find that the RBER for LSB pages remains comparable for both methods. However, the RBER for MSB pages programmed using the SFWS method is significantly higher than the RBER of corresponding pages programmed using LFWS. However, since the standard ECC threshold is around 0.1% [10, 13], the increased RBER is still far below the threshold value that can be corrected by the ECCs. Thus, the higher RBER with our proposed SFWS approach is correctable with the standard ECC available within the flash chip and does not create reliability issues.

Figure 3(c) shows the worst-page RBERs for each layer for LFWS and SFWS. We find that the RBERs for pages in the middle layers programmed using SFWS are higher than the equivalent ones when using the LFWS method. Lastly, Figure 3(d) shows the worst-page RBERs per each sub-block (S_0-S_{15}). The S_0 and S_1 yield significantly higher RBERs with SFWS and the RBER monotonically decreases with an increase of the sub-block number. Since S_0 and S_1 are written first, they experience the highest disturb as the subsequent sub-blocks are programmed in SFWS. Similarly, S_{15} is the last sub-block that is written and hence it has the smallest RBER among all sub-blocks. Interestingly, the worst-page RBER for SFWS falls below the one for LFWS in S_6-S_{15}. Thus, we conclude that the middle layers and the first few physical sub-blocks are slightly more erroneous in SFWS. However, the worst-page RBER values of SFWS are still correctable by the standard ECCs threshold value as they remain below ~ 0.1% [10, 13].

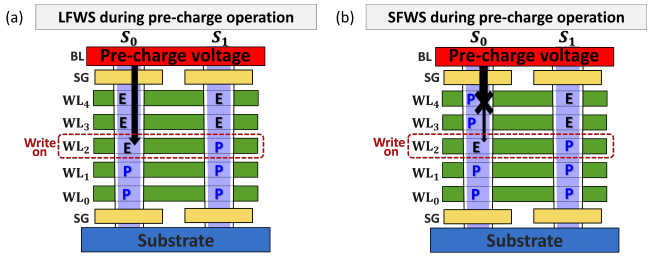

Figure 4: (a) LFWS and (b) SFWS during pre-charge operation.

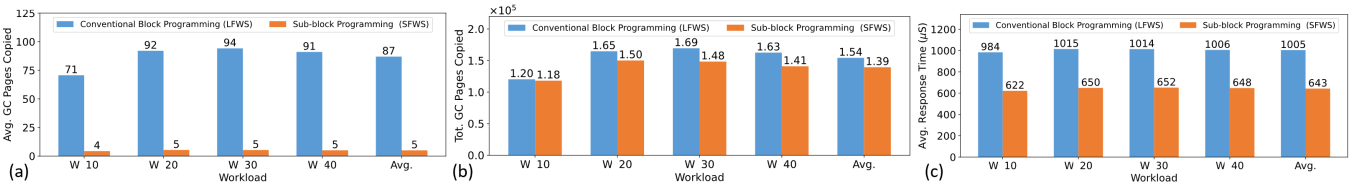

Figure 5: Comparison of (a) average number of copied pages during GC, (b) total pages copied during GC, and (c) average response time for LFWS and SFWS.

We conduct root cause analysis to explain the higher RBER in the middle layer pages for SFWS. Figure 4 illustrates a cross-section of a 3D NAND array corresponding to different sub-blocks during a pre-charge operation. The pre-charge operation, performed before applying the program pulse, boosts the poly-silicon channel voltage, preventing program disturbance. Consider a program pulse applied to WL_2 of S_0. In LFWS (Figure 4(a)), memory cells above WL_2 are erased, aiding efficient pre-charge voltage propagation. However, in SFWS (Figure 4(b)), programmed memory cells above WL_2 hinder pre-charge efficiency by blocking voltage propagation. In MLC memory, program disturbance primarily affects MSB pages by shifting the erase threshold voltage distribution, resulting in higher RBER. The middle layers show the worst-case RBER as they will experience the highest pre-charge voltage drop assuming pre-charge happens from both the BL and the substrate direction.

5.2 Performance Analysis

To explore the performance impact of using the SFWS approach, we use the SSD simulator MQSim-E [11]. We model an SSD using parameters shown in Table 1. Note that we use 2 planes per die, 2 dies per chip, 4 chips per channel, and 8 channels configuration. The simulation is performed using write-only random-access workloads (W_{10} to W_{40}) with a varying "hot" region to emulate real applications with spatial and temporal locality. For instance, W_{10} represents a random write-access pattern, where 90% of the requests target 10% of the address range, hence resulting in the highest access locality. Similarly, W_{20} has 80% of the requests targeting 20% of the address range, and so on. The total volume of data transferred is assumed to be equal, and large enough to trigger GC. We observe that the SFWS approach results in a smaller block size compared to the much larger block size in the conventional SSDs, where LFWS is used, leading to significant performance differences. The total SSD size is kept the same by adjusting the total number of blocks per plane according to the block size.

Table 1: Simulation Parameters

Total SSD Capacity	256 GB
Page Size	8096 Bytes
Pages Per Block	512 (LFWS), 32 (SFWS)
Blocks Per Plane	512 (LFWS), 8,096 (SFWS)
Interface	NVMe x4 lanes at 1.0 GB/s

Figure 5(a) shows the average number of pages copied for each approach during GC. Due to the smaller effective block size, SFWS significantly reduces page copy operations per GC operation with an average reduction of 95.2%. However, the total number of GC operations is significantly higher for smaller blocks. Figure 5(b) shows the total pages copied for all GC operations is still reduced by 9.7% on average against LFWS. Since the total number of host-issued writes is the same, fewer pages copied during GC leads to lower write amplification, improving SSD endurance. Finally, Figure 5(c) shows the average response time for each method, showing an average improvement of 36.0% due to reduced GC latency overhead.

6 CONCLUSION

This paper describes a new SFWS mechanism designed to alleviate the big-block problem in modern 3D NAND flash SSDs. This method coupled with the physical sub-block erase operations improves GC efficiency and reduces write amplification in 3D NAND flash SSDs. Our performance analysis showed how SFWS reduced GC overhead by 9.6% and average response time by 36.0% compared to the traditional LFWS approach. Experiments carried out on 3D MLC NAND flash chips showed that although the RBERs of SFWS are higher than those for LFWS, especially for the pages residing in the middle layers, they can still be easily corrected by standard ECCs. Thus, SFWS is a promising technique to improve performance in 3D NAND flash SSDs.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grant #2403540 and #2346853.

REFERENCES

[1] M. Buddhanoy et al. 2023. Electrostatic Shielding of NAND Flash Memory from Ionizing Radiation. In *2023 IRPS*.
[2] H. Y. Chang et al. 2016. How to enable software isolation and boost system performance with sub-block erase over 3D flash memory. In *2016 CODES+ISSS*.
[3] F. H. Chen et al. 2015. PWL: a progressive wear leveling to minimize data migration overheads for nand flash devices. In *2015 DATE*.
[4] T. Y. Chen et al. 2016. Enabling sub-blocks erase management to boost the performance of 3D NAND flash memory. In *Proc. of 2016 ACM DAC*.
[5] Y. A. Chung et al. 2023. Common Source Line-to-Word Line Short Improvement by Eliminating SLT Sidewall Notch in 3D NAND Deep Trench Patterning. In *2023 ASMC*.
[6] M. A. d'Abreu. 2015. Partial block erase for a three dimensional (3D) memory. US Patent No. 9,036,428, Issued May. 19th., 2015.
[7] M. Dunga et al. 2020. System and method for string-based erase verify to create partial good blocks. US Patent No. 10,535,411, Issued Jan. 14th., 2020.
[8] A. Goda et al. 2012. Scaling directions for 2D and 3D NAND cells. In *2012 IEDM*.
[9] H. Gong et al. 2021. Accelerating Sub-Block Erase in 3D NAND Flash Memory. In *2021 IEEE ICCD*.
[10] J. Kim et al. 2012. Low-energy error correction of NAND Flash memory through soft-decision decoding. *EURASIP J. Adv. Signal Process* (2012).
[11] D. Lee et al. 2022. MQSim-E: An Enterprise SSD Simulator. *IEEE Comput. Archit. Lett. 21* (2022), 13–16.
[12] C. Y. Liu et al. 2018. PEN: Design and Evaluation of Partial-Erase for 3D NAND-Based High Density SSDs. In *USENIX FAST '18*.
[13] Y. Luo et al. 2018. HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature Awareness. In *2018HPCA*.
[14] E. C. Oh et al. 2015. Nonvolatile memory device and sub-block managing method thereof. US Patent No. 20,140,063,938, Issued Feb. 24th., 2015.
[15] M. C. Yang et al. 2014. Garbage collection and wear leveling for flash memory: Past and future. In *2014 SMARTCOMP*.
[16] T. H. Yeh et al. 2017. 3D non-volatile memory array with sub-block erase architecture. US Patent No. 9,721,668, Issued Aug. 1st., 2017.
[17] L. Zuolo et al. 2017. Solid-State Drives: Memory Driven Design Methodologies for Optimal Performance. *Proc. IEEE* (2017).