Constructing Large, Durable and Fast SSD System via Reprogramming 3D TLC Flash Memory

Congming Gao
Chongqing University
University of Pittsburgh

Min Ye
YEESTOR Microelectronics Co., Ltd

Qiao Li
City University of Hong Kong

Chun Jason Xue
City University of Hong Kong

Youtao Zhang
University of Pittsburgh

Liang Shi
East China Normal University

Jun Yang
University of Pittsburgh

ABSTRACT

NAND flash memory based SSDs have been widely studied and adopted. The scaling of SSD has evolved from plannar (2D) to 3D stacking. Compared with 2D SSD, 3D SSD stacks more layers into one block, constructing one block with more flash pages. For reliability and other reasons, technology node in 3D NAND SSD is larger than in 2D, but data density can be increased via increasing bit-per-cell. However, representing multiple bits per cell encounters additional challenges such as endurance and access latency. In this work, we develop a novel reprogramming scheme for TLCs in 3D NAND SSD, such that a cell can be programmed and reprogrammed several times before it is erased. Such reprogramming can reduce the frequency of erases which determines the endurance of a cell, improve the speed of programming, and increase the amount of bits written in a cell per program/erase cycle, i.e., effective capacity. Our work is the first to perform real 3D NAND SSD test to validate the feasibility of the reprogram operation. From the collected data, we derive the restrictions of performing reprogramming due to reliability challenges. Further, a reprogrammable SSD (ReSSD) is designed to structure reprogram operations, and when they should be applied. ReSSD is evaluated in a case study in 3D TLC SSD based RAID 5 system (RSS-RAID). Experimental results show that RSS-RAID can improve the endurance by 30.3%, boost write performance by 16.7%, and increase effective capacity by 7.71%, with negligible overhead compared with conventional 3D SSD based RAID 5 system.

CCS CONCEPTS

• **Hardware → Memory and dense storage**; *Hardware reliability*;
• **Computer systems organization →** *Architectures.*

KEYWORDS

3D TLC SSD, reprogramming, endurance, performance, capacity, reliability, RAID 5

MICRO-52, October 12–16, 2019, Columbus, OH, USA

ACM ISBN 978-1-4503-6938-1/19/10…$15.00
https://doi.org/10.1145/3352460.3358323

ACM Reference Format:
Congming Gao, Min Ye, Qiao Li, Chun Jason Xue, Youtao Zhang, Liang Shi, and Jun Yang. 2019. Constructing Large, Durable and Fast SSD System via Reprogramming 3D TLC Flash Memory. In *The 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-52), October 12–16, 2019, Columbus, OH, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3352460.3358323

1 INTRODUCTION

NAND flash memory based Solid State Drive (SSD) is widely applied in storage systems due to its dramatic performance improvement, low power consumption, and lightweight form factors, compared to Hard Disk Drive (HDD) [39, 51]. To reduce cost per bit, SSD products take two approaches: 1) adopting higher bit density flash cells (MLC, TLC, QLC, PLC); and 2) constructing vertical architecture (3D SSDs [34, 35, 60]). These two approaches can be combined to achieve even higher capacity. Currently, 96-layer 3D TLC SSD with terabits of capacity per chip has been launched [18]. However, storing more bits per cell reduces endurance and operation speed sharply [5, 7, 15]. Although having more than 100 layers per block is on industry agenda, stacking more layers is also getting harder [27] due to limited space for control circuits that span across all layers [41].

While storing more bits per cell effectively multiplies the capacity of an SSD, the challenges in its endurance and performance remain. Recently, a *reprogram scheme* was proposed for 2D NAND SLC SSDs [10, 12, 14], which reprograms an SLC by dividing the large voltage range of an SLC into smaller ranges to represent new data in the cell. This directly increases the SSD capacity, which in turn decreases the occurrences of garbage collections (GC) and benefits both endurance and performance (since GC is both time-consuming and wear-sensitive) of an SSD. However, same mechanism cannot be applied to 2D TLC [15] as voltage range between neighboring states is too narrow to be further divided. However, the concept of reprogramming there was used to mainly increase the read performance. Neither capacity nor endurance was changed.

In this work, we propose a **Re**programmable 3D TLC **SSD** design, termed **ReSSD**. Our goal is to achieve capacity increase, endurance improvement and performance improvement. We design a novel encoding scheme for to-be-stored bits during each reprogram such that a TLC cell can be reprogrammed twice before an erase is necessary. The number of flash pages consumed by the same amount of program operations is decreased, increasing effective SSD capacity.

The benefits of fewer GCs, better endurance and performance naturally follow. As we adopt it in a 3D SSD, there are new challenges in reliability so that reprogramming should be applied with constraints. To the best of our knowledge, our work is the first to test reprogramming on a product 3D NAND SSD to verify its feasibility, and derive its constraints.

We performed a case study on implementing ReSSD design in Redundant Array of Independent Disks (RAID) 5 system (called as **RSS-RAID**). RAID 5 system has been widely adopted due to its good trade-off among performance, fault-tolerance, and redundant capacity consumption [11, 32, 50]. There are two important technologies adopted in RAID 5 system: Stripe and Parity Protection. Stripe is designed to evenly distribute data to multiple disks for performance optimization. Parity Protection is designed to protect user data loss within one stripe by generating additional parity data. However, with the adoption of SSD in RAID 5 system, extra capacity consumption resulted from Parity Protection will quickly increase, which triggers more GCs in the entire storage system, hurting endurance and performance of the RAID 5 system. As ReSSD is most suitable for frequently updated data streams, we applied it on a 3D TLC SSD based RAID 5 system in simulation and measured its effectiveness.

The results reveal that RSS-RAID can reduce capacity consumption and bring fewer GC activities, improving the lifetime and endurance, and increasing the effective capacity of 3D TLC SSD based RAID 5 system. On average, the endurance is increased by 30.3%, write performance is improved by 16.7% and effective capacity is increased by 7.71%. We summarized our contributions as follows.

- We verified the feasibility of reprogram operation on real 3D TLC NAND flash memory, from which we derive several important constraints on applying reprogramming;
- We proposed a reprogrammable 3D TLC SSD design to construct a large, durable and fast SSD system;
- We evaluated proposed ReSSD in 3D TLC SSD based RAID 5 system using a modified 3D TLC SSD based RAID 5 simulator [28]. The experimental results show that RSS-RAID effectively improves capacity, endurance, and performance of 3D TLC SSD based RAID 5 system.

The rest of the work is organized as follows: In Section 2, background of this work is presented. Section 3 describes the feasibility of reprogram operation on 3D TLC SSDs. In Section 4, the design of ReSSD is presented. In Section 5, a case study employing ReSSD is presented. In Section 6, the experiments are presented. In Section 7, related works are discussed. Finally, the work is concluded in Section 8.

2 BACKGROUND

2.1 3D Flash Memory based SSD

Currently, several vertical architectures have been proposed for 3D SSDs, including BiCS, P-BiCS, V-NAND, and TCAT [45]. In this work, Terabit Cell Array Transistor (TCAT) based 3D SSD designed by Samsung is discussed due to its better performance and reliability [30, 45].

Figure 1 shows the organization of a typical TCAT based 3D SSD [30, 41]. Inside SSD, there is a host interface, which is responsible for the connection between SSD and host system. The flash controller contains several important components, which are in charge of Garbage Collection (GC), Data Allocation (DA), mapping between logical address and physical address (Flash Translation Layer, FTL), and some other functions, such as Wear Leveling (WL) [1] and cache [19, 20]. Apart from these components, a flash chip controller is equipped to connect flash chips and flash controller. To boost the performance of SSDs, internal architecture is organized in four levels of parallelism, from channel to chip, to die, and to plane [22–24]. Inside each plane, multiple blocks are maintained. The main difference between 3D SSD and 2D SSD comes from the architecture of flash block.

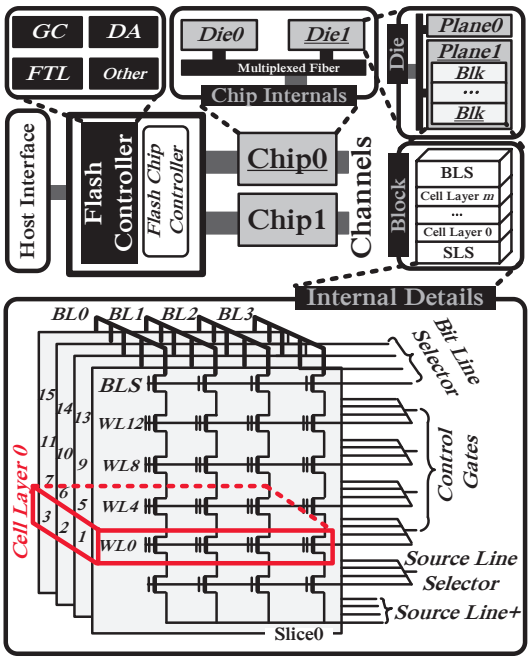


Figure 1: The architecture of 3D SSD.

The bottom of Figure 1 shows the vertical block architecture of 3D SSD, where flash cells are layered. Based on the vertical architecture, a new concept, named layer, is proposed in 3D SSD, which contains several word lines lying on a horizontal level. Each layer consists of multiple word lines and each word line is made up of multiple flash cells. In addition, since current 3D SSD uses larger process technology node (i.e., 30-50nm), one-shot programming (detailed below) is adopted to boost the write performance without making programmed pages error-prone [5]. With the assistance of three write-back registers, one-shot programming writes three pages (TLC based 3D SSD) to a word line at a time [26].

2.2 Voltage Distributions of Triple-Level-Cell

TLC has been widely adopted in 3D SSDs due to its reasonable trade-off among bit density, performance, and reliability. Figure 2 shows a voltage distribution of TLC [15], of which the coding scheme can reduce the number of data sensing iterations for read operation [46]. In this figure, the x-axis represents the possible voltage threshold (Vth) in the flash cell, and the y-axis shows the probability in each state. Through charging different number of electrons into a flash cell, the entire voltage range can be divided into eight states (ER, P1~P7). Each state represents information of

three bits, Least-Significant-Bit (LSB), Central-Significant-Bit (CSB), and Most-Significant-Bit (MSB). With a fixed coding scheme, three-bit values can be represented in a TLC by moving state from "ER" to the right along the x-axis. This voltage state movement can be realized by a program operation, termed "one-shot programming" in 3D SSD [26]. Since there is only one charging process for one-shot programming, three write-back registers per plane are equipped to hold three pages that will be written into the same word line at a time. For read operation, read sensing circuits use seven reference voltages (Vref) to distinguish these eight states, while other unread cells are applied with a pass-through voltage (Vpass) [6].

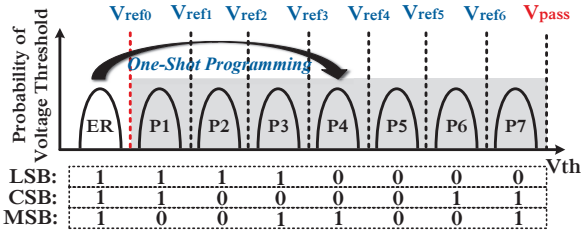


Figure 2: The voltage distribution of TLC [15, 47].

2.3 Garbage Collection and Endurance of SSD

SSD adopts out-of-place update for program operations, meaning updates are written to free pages and original pages are invalidated because a cell can be only written from an erased state ("ER" state in Figure 2). Once free space of SSD drops to a pre-defined threshold, GC is triggered to reclaim invalid pages for serving future program operations. A GC migrates valid pages in a victim block to free space and erases the victim block. The efficiency of GC is highly related to the number of valid page migration [8, 21]. Typical optimization scheme selects the block with largest number of invalid pages to reclaim, so as to minimize the cost of valid page migration [1, 28, 53].

GC not only introduces performance overhead, but also consumes the lifetime of flash cells. The endurance of SSD is quantified by the Program/Erase (P/E) cycle which indicates the manufacturer guaranteed maximum number of program and erase operations executed on flash cells without introducing unrecoverable errors. As cells can only be written once between erases, one effective approach to improve the endurance of SSDs is to reduce the number of erases, or GCs [14, 36, 37]. However, endurance can also be improved if more writes can be performed between erases. As the duration of voltage stress in the program operation is less than the erase operation, previous studies have shown that cells are not significantly further worn in a program operation [14, 31, 49]. Therefore, if cells can be written multiple times before an erase operation, the number of erases or GCs can be reduced and endurance of SSD can be improved.

3 REPROGRAMMING ON 3D TLC SSDS

3.1 Feasibility of Reprogramming Strategy

Reprogramming 2D flash memory has been developed in previous works [10, 14, 36, 37]. The large voltage range between two voltage states in an SLC is leveraged to reprogram a cell by increasing V_{th} multiple times to represent new values in-place before a true erase becomes necessary. In [15], similar approach is adopted on TLC, where the lower voltage state is reprogrammed to a higher voltage state to realize in-place update. However, previous reprogramming schemes cannot be directly applied to 3D TLC SSDs due to two additional restrictions. First, the voltage range between two voltage states in TLC is too narrow to be further divided; Second, reprogramming flash cells will result in poor reliability on 3D TLCs, as will be demonstrated in this paper.

3.2 Reprogram Operation on 3D TLC

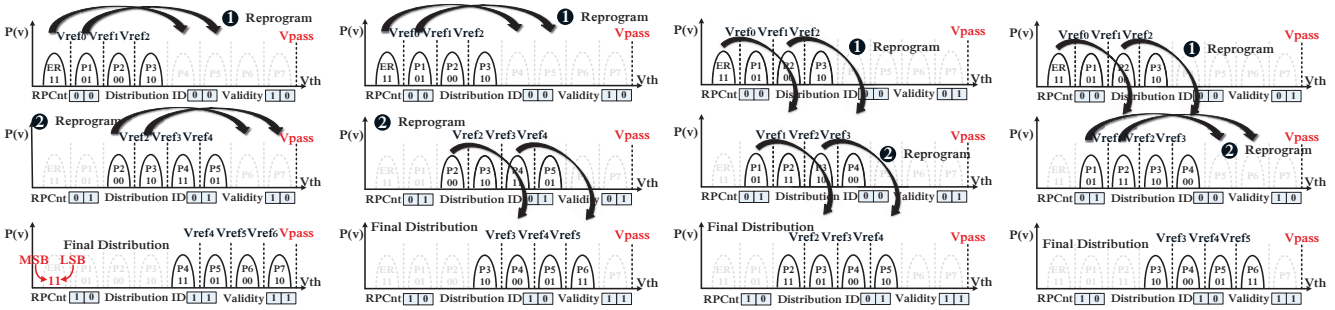
In this work, we develop a novel reprogramming technique to avoid further narrowing the range between voltage states in TLC flash cells. The basic idea is to first program a TLC as a MLC by only using the first four voltage states to represent 2-bit value, as shown in the top of Figure 3(a). Then, as the threshold voltage can keep increasing but not decreasing, low voltage states can be reprogrammed to higher voltage states to represent new values. In this case, a cell can be reprogrammed several times before a true erase is necessary. This is in contrast to the conventional model where a cell is programmed only once and the second program must be preceded by an erase.

In our design, program operation is realized by one-shot programming, which writes two pages at a time from the write-back registers to a word line. In these two pages, the paired bits with the same in-page offset are used to determine the desired state of a cell. One-shot programming injects electrons into each flash cell until its current threshold voltage reaches the desired voltage state which is coded to represent the values of two bits (MSB and LSB) [5]. Once one of the two bits in a cell becomes invalid, a reprogram operation can be carried out to transform the old 2-bit value into a new 2-bit value represented by higher voltage state in the cell. If the old 2-bit value remains valid, reprogramming on this cell is then prohibited. If both of two bits becomes invalid, reprogramming on this cell is performed by assuming one of the two bits still is valid. We create such design on TLC cells, and discovered that to cover all possible bit update sequences with altogether eight valid voltage states, two reprograms can be carried out after the initial program. Hence a cell can be programmed three times. The transitions from old 2-bit states to new 2-bit states are enumerated in Figure 3. In the top of Figure 3, the leftmost four voltage states (ER~P3) are used to represent 2-bit value after the initial program operation. Then, according to which bit is invalidated, a new voltage distribution is constructed by using current or higher voltage state to represent the new 2-bit value. More details are presented as follows.

Table 1: State changes during two reprogram operations.

Initial state	1^{st} reprogram	Middle state	2^{nd} reprogram
11	11 or 10	00	00 or 10
01	01 or 00	10	10 or 00
00	00 or 01	11	11 or 01
10	10 or 11	01	01 or 11

As a 2-bit value can only be updated twice on a cell, depending on which bit is invalidated first, there are four possible reprogramming sequences that can be represented by four voltage distributions, as depicted in the four subfigures of Figure 3. Then, since each bit can be reprogrammed into two possible values, there exist eight transitions as listed in Table 1. Take Figure 3(b) as an example. The two transitions from top to bottom correspond to the invalidation of the LSB and then the MSB, as shown in the left two columns and right two columns in Table 1 respectively. For the first transition that updates LSB, there are eight possible state changes, as listed in the left two columns of Table 1. The voltage distribution needs to be



(a) Reprogram(LSB/LSB) (b) Reprogram(LSB/MSB) (c) Reprogram(MSB/MSB) (d) Reprogram(MSB/LSB)

Figure 3: State movements and voltage distribution arrangements. Reprogram(LSB/MSB) means the LSB of a MLC mode cell is reprogrammed first, followed by reprogramming the MSB.

reprogrammed to higher states and re-encoded because otherwise, there would be two backward state transitions which are impossible without an erase. These two backward transitions are "00"–>"01" and "10"–>"11". Hence, "11" and "01" are mapped to P4 and P5 states respectively, shifting the four states to the middle of the entire voltage range, as depicted in the middle of Figure 3(b). Hence, all LSB updates are represented by increasing voltages, which are viable in one-shot programming. Similarly, as the MSB is updated in the second transition, state P3~P6 are utilized and re-encoded to represent new voltage distribution. Specifically, "00" is represented by P4 and "11" is represented by P6, to avoid backward state changes. Totally, after reprogramming a cell twice, the total number of bit-per-cell can be increased from 3 to 4.

Before reprogramming a cell, current voltage distribution, 2-bit value inside the cell and current validities of LSB and MSB should be read out first to determine next reprogramming voltage state. Then, to distinguish voltage distribution of a cell after reprogram operation, each voltage distribution is assigned with a unique 2-bit ID, termed *Distribution ID*. Specifically, "00" represents voltage distribution from P1 to P4; "01" represents P2 to P5; "10" represents P3 to P6 and "11"represents P4 to P7. Since *Distribution ID* is effective only when the cell has been reprogrammed, another counter (*RPCnt*) is used to record how many times the cell has been reprogrammed. By default, the value of *RPCnt* is set to "00". It is incremented every time the cell is reprogrammed. A value of "10" prohibits future reprogramming on this cell. For read operation, with the awareness of *RPCnt* and *Distribution ID*, the value stored in the reprogrammed cell also can be read out by only applying two sensing operations [4, 6]. Then, current validities of programmed two bits (*Validity*) are used to determine the next voltage distribution before reprogramming the cell. *Validity* requires 2-bit to indicate valid states of LSB and MSB, respectively. In default, the value of *Validity* is set to "11" after the initial program operation, meaning both of LSB and MSB are valid. If one of programmed two bits is invalidated, corresponding bit in *Validity* is set to "0". For example, as shown in top of Figure 3(a), after initial program operation, if LSB is invalidated, *Validity* is set to "10", meaning the cell can be reprogrammed to update the value of LSB. After reprogramming the cell, the value of corresponding bit in *Validity* should be set to "1", meaning the reprogrammed LSB in the cell is valid now and is prohibited to be further reprogrammed until its value is invalidated again. In this work, *RPCnt*, *Distribution ID* and *Validity* are termed as cell status. Note that, since all flash cells in a word line are programmed or reprogrammed simultaneously, *RPCnt*, *Distribution ID* and *Validity* are needed on per word line basis. They are checked before each program or reprogram operation.

We use Figure 3(a) as an example to illustrate the entire hardware procedure of reprogramming with the aid of metadata. Suppose the initial program operation writes ("00") in the TLC. The values of *RPCnt*, *Distribution ID* and *Validity* are set to "00", "00" and "11" by default. As the LSB is invalidated, the value of *Validity* is set to "10". Then, the value of LSB is updated to "1" ("00"–> "01"), the corresponding voltage state of the new value is mapped to P5 state for constructing new voltage distribution as presented in the middle of the figure. Moreover, *RPCnt* and *Distribution ID* are set to "01" and "01" to represent current cell status. After reprogramming the cell, the value of *Validity* is set to "11", indicating current values of two bits in the cell are valid. For the second reprogram operation, with the aid of *Distribution ID*, current 2-bit value ("01") inside the cell is read out first. We assume that the LSB of "01" is invalidated (the value of *Validity* is set to "10") and updated to "0" ("01"–> "00"). Thus a new voltage distribution presented in the bottom of the figure (final distribution) is constructed to represent new value ("00") in the P6 state. After that, *RPCnt* and *Distribution ID* are updated to "10" and "11" as the cell cannot be reprogrammed any more before an erase. And the value of *Validity* is also set to "11" after the second reprogram operation on the cell. In summary, at any point in time, there can be at most two bits inside a cell. However, when a cell is reprogrammed twice, four bits of information are programmed or reprogrammed into a TLC ("00" in first initial program, "1" in first reprogram and "0" in second reprogram). As a result, from the overall SSD perspective, the effective capacity can be increased by reprogramming a cell twice.

3.3 Reprogram Operation Verification

To verify the feasibility of reprogram operations on 3D TLC SSDs, we evaluate them on YEESTOR 9083 flash memory testing platform [59] with Micron B17A Series NAND flash chips [48]. YEESTOR 9083 flash memory testing platform shown in Figure 4, is a high-reliability and high-performance flash memory controller that enables users to implement specified flash memory solutions.

As discussed earlier, reprogram operation is enabled as one of old two bits has been invalidated in a cell. Since there are many word lines in a block in 3D TLC SSDs, flash cells in a word line may be invalidated and rewritten whenever necessary, and not following conventional sequential program order. With the capability of

reprogramming, in our verification design, we relax the restriction of sequential order among word lines through two steps. In the first step, the restriction of sequential order among word lines in a layer is relaxed. All word lines in a layer are reprogrammed in a random order while layers are reprogrammed sequentially. Such a reprogramming scheme is termed "One-Layer Reprogram" or OLR in this work. In the second step, the restriction of sequential order among layers is relaxed by reprogramming word lines in multiple layers in a somewhat random order. We term such scheme "Multi-Layer Reprogram" or MLR. More details about OLR and MLR will be presented in the following subsections. In the verification design, to show the maximal capability of a cell to be reprogrammed, each cell in the following experiments is reprogrammed four times to completely shift the initial voltage distribution to the rightmost side. In real SSD products, each cell only can be reprogrammed at most twice after initial program operation by employing the designed voltage distribution arrangement.

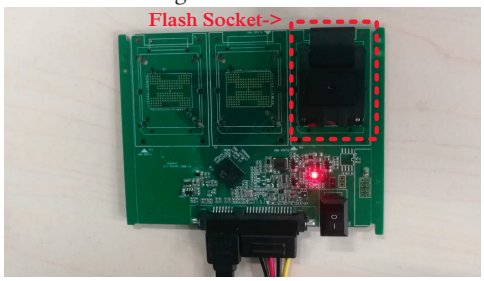


Figure 4: YEESTOR 9083 testing platform.

3.3.1 One-Layer Reprogram. **Evaluation Methodology:** For the OLR scheme, initially, all word lines within each layer are sequentially programmed such that all flash cells in each word line are programmed in a MLC mode. Then, word lines in the layer are randomly selected to reprogram flash cells by moving the first four states to rightmost four states step by step. To evaluate the reliability of OLR, the worst voltage distribution (ER~P7) of a cell is evaluated and presented in Figure 5, where x-axis represents the voltage threshold and y-axis represents the number of flash cells falling into different voltage states. As shown in the figure, eight voltage states (including the ER state) are distributed to different threshold voltage ranges separately.

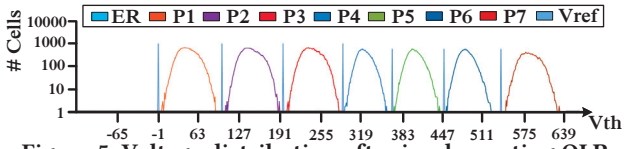


Figure 5: Voltage distribution after implementing OLR.

Reliability Impact of OLR: To analyze the impact of reprogramming on reliability, the maximum and average numbers of bit errors per page are evaluated. There are two types of read methods for reading and checking the raw data in each page. The first read method uses default voltage offsets (DVO) and the second read method adopts voltage calibration that can minimize the number of page bit errors by applying optimized voltage offset (OVO) between every two neighboring states [4, 6, 25, 48], which has been used in SSD products. The results of maximum and average numbers of page bit errors collected by these two read methods are presented in second row of Table 2. As a contrast, the results of conventional program scheme are evaluated as well and presented in the first row of this table. As we can see, with voltage calibration read method, the numbers of maximum and average page bit errors are increased by 4 and 1 respectively from conventional program scheme. Given the strong ECC capability (e.g., 80bits/1KB) in modern SSD products [25, 48]), the increased number of page bit errors resulted from OLR is negligible. Therefore, we conclude that reprogramming flash cells layer by layer in 3D TLC SSDs is effective and reliable.

Table 2: Bit errors of conventional scheme, OLR and MLR.

# of Page Bit Errors	DVO		OVO	
	Max	Mean	Max	Mean
Conventional Program	889	291	6	1
One-Layer Reprogram	1428	432	10	2
Two-Layer Reprogram	1851	438	46	9
Three-Layer Reprogram	2096	349	234	11

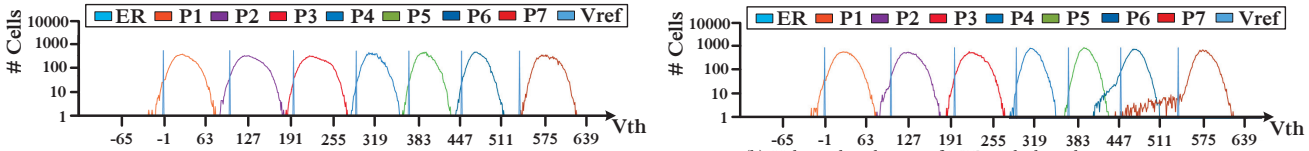
3.3.2 Multi-Layer Reprogram. To relax the restriction of sequential order among layers, in our design, we group multiple physical layers as a reprogrammable super layer. Inside each reprogrammable super layer, physical layers are reprogrammed in a random order, while reprogrammable super layers are programmed in sequence.

Evaluation Methodology: We measured the feasibility of MLR starting with a super layer containing two physical layers. The evaluation methodology is similar to that for the OLR. Initially, all word lines in a super layer are sequentially programmed, and all cells are programmed as a MLC utilizing only the ER~P3 states. Then, reprogram operations are carried out in randomly selected word lines to move the first four states to the rightmost four states step by step. To evaluate the reliability of MLR with two-layer reprogramming, the worst voltage distribution of a cell is presented in Figure 6(a), where eight states can still be clearly distinguished by reference voltages. When MLR is evaluated with three physical layers per super layer, the worst voltage distribution tends to be error-prone, as seen in Figure 6(b). In this figure, the left tails of some voltage states are largely stretched across reference voltages.

Reliability Impact of MLR: Quantitatively, the maximum and average page bit errors per page of MLR with two-layer reprogramming are 46 and 9 respectively, as presented in the last two rows of Table 2, which can still be corrected by ECC. However, as the number of physical layers per super layer increases to three, MLR produces too many bit errors to be correctable (234 in maximum and 11 on average), due to large overlap between adjacent voltage states in the far right, as seen in Figure 6(b). Those errors indicate that MLR with three layers per super page may not be reliable enough, especially in the evaluated 3D TLC flash chips, considering there are other error sources such as P/E cycles induced errors and retention error [4]. When we further increased the size of super layers, the number of bit errors per page is increased to thousands even under read voltage calibration. The voltage distribution becomes worse. ***Therefore, the maximum number of physical layers that can be reprogrammed as a super layer in this work is set to two.***

3.3.3 Reliability Discussion. We first discuss why reprogram operations introduce challenges in maintaining reliability. Then, we give further evaluations on reliability when aggregating all error sources from reprogramming, P/E cycle and retention.

Reliability Challenges: There are two main reasons for reprogram operation to challenge reliability: cell to cell interference [4, 7] and background pattern dependency (BPD) [13, 33, 38]. The former is slightly weaker and the latter is dominant in harming reliability.



(a) Voltage distribution of MLR with two-layer reprogramming

(b) Voltage distribution of MLR with three-layer reprogramming

Figure 6: Worst cases of voltage distributions with implementing MLRs.

First, reprogram operation will aggravate its harm to the reliability due to the impact from cell to cell interference, which is highly related to how many times neighboring cells are programmed or reprogrammed [7]. In the OLR and MLR evaluated above, each cell is programmed once and reprogrammed four more times to fully moving voltage state to the rightmost side, causing four more times cell-to-cell interference to neighboring cells. Put it in another way, each cell receives four more times cell-to-cell interference from reprogramming on top of the cell from program operations in the conventional scheme. However, in real SSD products, with applying voltage distribution arrangement, a cell only can be reprogrammed twice at most so that the impact from cell to cell interference can be further mitigated.

Second, BPD means the current-voltage characteristic of a cell can be affected by whether other cells in the same cell string (a cell string means flash cells sharing the same bit line) have been programmed or reprogrammed. The resistance of a cell string increases as more cells are programmed in the same string [33]. Therefore, BPD makes the electrons hard to be injected into a cell if there are cells atop already programmed [46]. To suppress the reliability impact from BPD, a sequential program order is placed to program word lines from bottom (closest to the source line+) to top (closest to the bit line contact), as shown in Figure 1. However, the proposed OLR and MLR break conventional sequential program order, aggravating the reliability impact from BPD. As a result, some cells cannot be reprogrammed to desired state within the manufacturer guaranteed maximum times of program & verify iterations, creating the left tail of voltage distribution, as depicted in Figure 6. In addition, such a left shift in voltage distribution also shows that BPD is the chief influence on reliability, as cell to cell interference tends to shift voltage distribution to the right side [7].

Reliability Impact from Other Error Sources: Here we quantitatively analyze how reprogram operations worsen reliability when there are already error sources from such as P/E cycles and retention time.

First, we test the reliability of cells when they are stressed under both P/E cycles and reprogramming. An experiment is constructed by reprogramming two blocks with two types of P/Es, the normal P/E and reprogram P/E. The former means a block is repeatedly programmed and erased without reprogram operations, and the latter means a block is worn by repeated MLR with two-layer reprogramming and normal erase operations. After 1K normal and reprogram P/E cycles on two blocks, the maximum and average numbers of bit errors per page are collected and presented in the left two columns of Table 3 (under w/o Retention). Compared to normal P/Es, reprogram P/Es increase the maximum and average number of bit errors by 7 and 4 respectively. Such a slight increase in page bit errors can be easily corrected by ECC. Moreover, the experiment also proves that reprogramming a cell multiple times before an erase operation induces little wear stress on a cell. The reason is that the stress duration of erase operation is much longer than that of program and reprogram operations [31]. That is, the main wear stress on a cell comes from erase operation even when a cell is reprogrammed multiple times.

Second, we test the reliability with further error source from retention when the cells are already stressed under P/E cycles and reprogramming. The above evaluated chip is heated to 120°C for three hours, of which the effective retention time is 1 year under 40°C [2, 3]. The maximum and average numbers of page bit errors per page collected by read voltage calibration are presented in right two columns of Table 3. Compared to normal P/E, reprogram P/E increases the maximum and average numbers of page bit errors by 21 and 15, respectively. In the worst case, after 1K reprogram P/E and 1 year retention under 40°C, there are 121 bit errors per page, which still can be corrected by ECC. Notice that, according the results presented in Table 2 and Table 3, the maximum number of bit errors after 1K reprogram P/E cycles (27) is even smaller than that of MLR with two-layer reprogramming without P/E impact (46). The reason is that the chips evaluated in these two experiments are different but with the same specification. We remark here that the impact of process variation (PV) [52, 54] dominates error rates over reprogramming.

Based on the above results, we conclude that the reprogram operation still is reliable enough even when P/E cycles and retention time are considered.

Table 3: Page bit errors under different retention time and two types of P/E cycles.

# of Page Bit Errors	w/o Retention		w/ Retention	
	Max	Mean	Max	Mean
Normal P/E	20	5	100	34
Reprogram P/E	27	9	121	49

Risk of Capacity Loss: By degenerating TLC to MLC, the number of bit-per-cell is reduced from three to two, causing capacity loss of SSD. On the other hand, if data is frequently updated, reprogramming can write more bits into a cell before an erase, increasing effective capacity. Hence, reprogramming hot data is more beneficial than cold data in terms of capacity. However, if the hotness of programmed data cools down and remains valid, the reprogram operation on this cell will be prohibited, thereby resulting in capacity loss. Moreover, given the sequential order among reprogrammable super layers, previous reprogrammable super layers are not allowed to be reprogrammed again once the next reprogrammable super layer is activated. Thus the capacity may become lost while flash cells in previous reprogrammable super layer have not been reprogrammed. An extreme case is that each reprogrammable layer is programmed into MLC mode but never reprogrammed. Then one third of capacity is lost.

However, such capacity loss does not have lasting effects on SSD, due to GC process that is used inside SSD to reclaim invalid or lost space. In this work, if there exists lost space, GC is activated to move valid data from reprogrammable blocks to normal blocks as cold

data [57], and then, the reprogrammable block is erased and reset to a normal block. Additionally, if only hot data can be reprogrammed, most data in reprogrammable blocks tends to be invalid as each page can only be programmed twice. Hence, reprogrammable blocks are more likely to be GC-friendly, meaning that they require less copies for valid data. Since hot data is frequently invalidated and updated, reprogram operation induced page bit errors will not be propagated by reading and rewriting updated data.

Selective Reprogram: Based on the above analysis, we draw the conclusion that ***reprogramming flash cells in 3D TLC SSDs is feasible*** but should be applied selectively. First, a flash cell can be reprogrammed only when some of the programmed bits are invalidated. Second, the maximum number of physical layers per reprogrammable super layer should not exceed two for reliability reasons (according to the chips we tested). Third, to avoid capacity loss, only hot data that can be updated soon is suggested to be reprogrammed.

4 THE DESIGN OF RESSD SYSTEM

Following the restrictions above, a reprogrammable 3D TLC SSD (ReSSD) design is proposed, as outlined in Figure 7. Inside ReSSD, the hotness of each write request is detected by the Hotness Filter module. Hot write requests are reprogrammed and cold write requests are regularly programmed. *TimeStamp Recording* and *Reprogram Reference* modules are used to assist reprogram operation. Then, another function, termed Fully Invalidated, is added to passively invalidate cooling data in reprogrammable block and make sure all data in reprogrammable block can be invalidated soon.

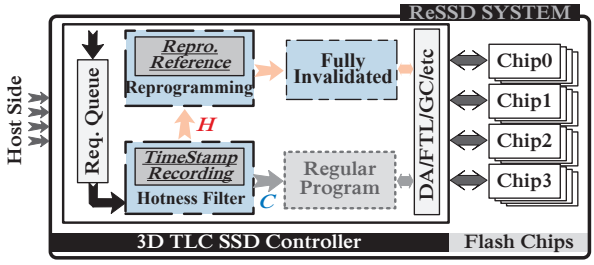


Figure 7: The overview of ReSSD system design.

4.1 Selecting Hot Writes for Reprogramming

To detect hot write requests, Hotness Filter defines the hotness of each write request and determine where hot requests should be written to. The filter follows these two rules:

Rule 1: Only updated requests are selected as hot write requests, as most updated data has high potential to be updated again;

Rule 2: Writes with similar hotness are placed in the same block.

In this work, we categorize the hotness of write requests into four levels based on the update time interval between two continuous writes accessing the same data. In detail, data to be updated either within 30 minutes, between 30 minutes and 60 minutes or between 60 minutes and 120 minutes is defined as hot data. For data that never be updated or to be updated over 120 minutes, it is defined as cold data. Within 120 minutes, most data in server workloads are updated according to [14]. Such update time interval calculation requires to record the arrival time of each write. However, maintaining time stamps of all write requests is space-consuming, thus per-page time stamp is replaced by block level granularity time stamp, which is set as the arrival time of first write request programmed to this block.

Based on the hotness determined by update time interval, write requests are distributed to the blocks containing data with the same hotness. In our design, flash blocks are divided into Normal Blocks for regular TLC programming, and Repro. Blocks for reprogramming. We use a simple example illustrated in Figure 8 to show how data are distributed among those two types of blocks. When the first arriving write ❶ is programmed to *Normal Block 0*, the time stamp of *Normal Block 0* is set to *TS 1*. Later when the same data is updated, it is considered hot by calculating the inter arrival time between *TS 1* and its arrival time, and the update is placed in *Repro. Block 1* and block time stamp of *Repro. Block 1* is set to *TS 2*. Here the first super layer is created and the two updates ❷ and ❸ are allocated to this super layer. Suppose there are some other updates land in this super layer, and it becomes no longer reprogrammable (all data are valid), a second super layer is activated to host future updates. Suppose the same data is updated at the third time (❹), it is now allocated to one word line in the second super layer. Recall that each Repro. Block maintains only one active super layer, the time stamp of *Repro. Block 1* is updated to *TS 4*.

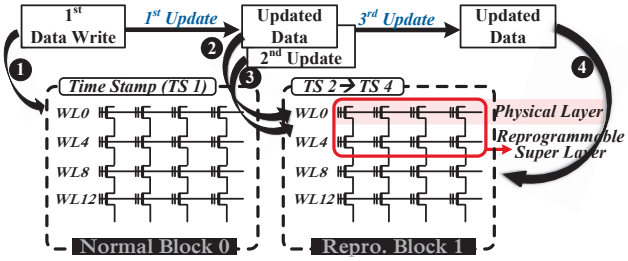


Figure 8: Hotness based data reprogramming.

4.2 Reprogrammable Block

4.2.1 Writing a Super Layer. We use an example to explain how writes are programmed and reprogrammed in the reprogrammable super layer, as illustrated in Figure 9. We assume that there is one physical layer per reprogrammable super layer and each physical layer contains four word lines. Each word line is first programmed utilizing states ER~P3, termed MLC Program, and then reprogrammed twice utilizing remaining states. Figure 9 shows these steps in four columns where the first two represent status of the two bits written in the MLC. At time ❶, there are eight updates arriving, which have the same hotness and they access the same reprogrammable super layer. Each paired writes (e.g.<*P0,P1*>) are first written via MLC programming. After writing the first two pairs, <*p3*> in the third pair invalidates its previous version in *WL1* and writes new data to *WL2*. Similarly, <*p0*> in the fourth pair also invalidates previous version in *WL0* and writes new data to *WL3*. When all word lines in the super layer have been programmed in the MLC mode, reprogramming begins. At time ❷, three more writes arrive, in which <*P6*> is a new write request and <*p4*>, <*p1*> invalidate and update their previous data. Since a previous version of <*P0*> has been invalidated in *WL0*, <*P6*> can be reprogrammed in *WL0*. For the remaining two updates, their previous versions in *WL2* and *WL0* are invalidated first. Then, new data is reprogrammed to *WL1* and *WL2* that still can be reprogrammed. At time ❸, since previous data of <*P1*> in *WL0* has been invalidated

and *WL0* has not been reprogrammed, <*P8*> can be reprogrammed, paring with <*P6*>. Totally, two reprogram operations are executed on *WL0*. Notice that, if <*p1*> is first written to the super layer at time ❷, the previous version of <*p1*> in *WL0* is invalidated, making both of old two pages in *WL0* invalid. To reprogram new data of <*p1*> to *WL0*, we assume there still exists one valid page so as to match one of the four 2-bit value transitions presented in the middle of Figure 3.

After time ❸, each word line in this super layer maintains two valid pages, prohibiting future reprogramming. Thus, we suspend current reprogrammable block as a candidate block, and select another available reprogrammable block. For candidate blocks, they can be switched back to reprogrammable blocks proactively or passively. If valid pages in candidate blocks are proactively invalidated, the corresponding word lines in this block are set to be reprogrammable again. However, if there are no available reprogrammable blocks but still some candidate blocks, the latter should be reused with higher priority to avoid greedy allocation of reprogrammable blocks. In this case, valid pages in a candidate block are migrated to normal TLC blocks as cooling data, making candidate blocks reprogrammable again. This not only avoids capacity loss, but also can reduce GC cost by moving valid pages in advance. Such invalidation also helps to correct errors created by reprograms as they are moved to normal TLC blocks. In addition, since only hot data is selected and reprogrammed, the cooling data that needs to be invalidated and migrated passively only account for a small part of overall reprogrammed data.

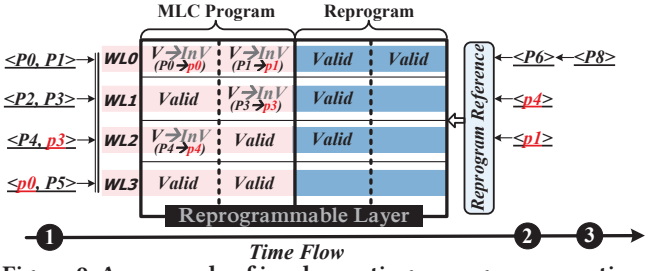


Figure 9: An example of implementing reprogram operation on reprogrammable super layer.

4.2.2 Reclaiming Candidate Blocks. The migration of valid pages during candidate block reclaiming may block host requests To minimize such a conflict, the write-back registers equipped for one-shot programming in 3D TLC SSDs are leveraged. In our design, candidate block with fewest valid pages is selected and its valid pages are read to the write-back registers. To hide the latency of valid page migration, they are programmed to a normal TLC block accompanied with incoming host write requests. At each time, the number of valid pages being read out from a candidate block is decided by the number of current host write requests accessing the write-back registers. One or two valid pages should be read out to form a three-page word line written to a normal TLC block. According to the settings of 3D TLC SSDs in Section 6, within each reprogrammable super layer, there are four word lines in a super layer, meaning at most each word line maintains four valid pages and total eight valid pages in a super layer should be sequentially migrated . Thus the overhead of reclaiming a candidate block is at most two valid page reads per host write request.

4.3 Overhead Analysis

Space for storing metadata. In this work, we assume a 576GB 3D TLC SSD containing 64 planes, 1536 blocks per plane and 384 pages per block. The detailed parameters and configurations can be found in Section 6. The overhead of implementing ReSSD comes from four aspects. *First*, each block maintains a 4-byte time stamp [44, 55], which sum up to 384 KB for the entire SSD. *Second*, in this work, we divide data hotness into four levels and data with top three levels is reprogrammed. Therefore, for each block, a 2-bit tag is added to represent four types of hotness levels. In addition, another 1-bit tag is added to distinguish candidate and reprogrammable block. Totally, 36 KB is required for those information. *Third*, in traditional SSD, each plane is equipped with a pointer to record active block. While proposed ReSSD is employed, additional three pointers per plane are required to record in-plane addresses of three active reprogrammable blocks with different hotness. Based on the total block number of each plane, each pointer needs 11 bits. In addition, to record the address of current reprogrammable word line, three 7-bit pointers are required per plane. Totally, additional 432 bytes are needed for the entire 3D TLC SSD. *Fourth*, the status of each word line should be maintained, which requires 2-bit to record *RPCnt*, 2-bit to record *Distribution ID* and 2-bit to record *Validity*. Therefore, the space overhead of word line status is 9 MB. In total, less than 10 MB additional space for each 3D TLC SSD is required. On the other hand, for reprogrammable blocks, since each word line only contains two pages at any time, the space cost of address mappings can be reduced to 2/3 of a normal TLC block. Since the space cost for each mapping entry (recording the paired addresses from logical address to physical address) is 7 bytes, the saved space of mappings per block reaches to 128×8 bytes=896 bytes as total 128 mapping entries are saved. That is, if there are twelve blocks being reprogrammed, the additional space cost can be covered by the saved space of mappings (12×896 bytes=10.5 MB).

Latency of a reprogram operation. Finally, reprogramming a word line is carried in two steps. First, the word line status information including *RPCnt* and *Distribution ID* are checked ("Reprogram Reference" in Figure 7), and the data is read out from the word line. Those information are used to determine how many times the word line has been reprogrammed and the location of the current state, which is used to determine how to shift state in the reprogramming. Therefore, compared with conventional program operation, the reprogram operation experiences an additional read time. However, benefiting from the fewer voltage states covered by arranged voltage distribution, a cell is reprogrammed with fewer charging cycles, resulting actual smaller latency than a full program operation [15]. As a result, the total latency including the read overhead is still smaller than that of the conventional program operation (detailed parameters are shown in Table 4.

5 CASE STUDY: RESSD IN RAID 5

The ideal application of ReSSD design should be update-intensive, which quickly invalidates previous data in reprogrammable block. In this work, 3D TLC SSD based RAID 5 systems, which requires significant capacity consumption and frequent updates to parity data, is selected as a typical application to verify the effectiveness of ReSSD design in RAID 5 systems [11, 32, 50].

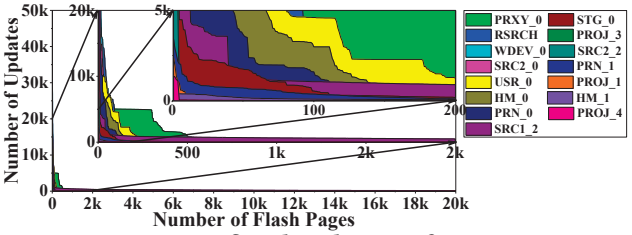


Figure 10: Zipfian distribution of IO requests.

In RAID 5 systems, the adopted Stripe and Parity Protection can introduce significant write amplification, which will accelerate capacity consumption and trigger frequent GCs. To verify the severity of this problem, first, the number of parity accompanied by user data writes should be evaluated. The RAID 5 setting is configured based on */drivers/md/* module of Linux Kernel *5.0.2* Version, where a maximum stripe count is set as the maximum cached stripe number to realize the cache function for reducing the update frequency of parity data [16, 29, 40]. On average, the number of parity writes, which is normalized to total user writes, reaches 88.1%. Such a large parity writes will occupy a lot of physical pages, causing frequent GC activities and negative impact on performance and endurance. Second, the average parity update time interval is also evaluated and normalized to user data update time interval. On average, the parity update time interval is reduced by 32.3%. The update time interval decrease comes from the design principle of parity protection, which requires parity to be updated while any user data in the same stripe is written or updated. In addition, the update frequency of all write operations (including user writes and parity writes) is evaluated and presented in the form of zipfian distribution as shown in Figure 10 [56]. The x-axis shows the count of logical addresses accessed by write operations at flash page granularity and y-axis represents the number of update operations on each logical address. This figure shows that only a small part of logical addresses tend to be frequently accessed by update requests while the rest are barely updated. On average, 73.9% of update operations occur in 20% of logical addresses (80/20 rule [17]). Therefore, to mitigate the impact of frequent parity update operation, the proposed ReSSD design is employed in 3D TLC SSD based RAID 5 system, called as **Re**programmable 3D TLC **SS**D based RAID 5 system (**RSS-RAID**).

6 EXPERIMENT

6.1 Experimental Setup

Simulated SSD based RAID 5 System: A workload driven flash memory simulator, SSDSim [28], is used in this work. To match the characteristics of 3D TLC SSD based RAID 5 system, three modifications are made in SSDSim. First, the parameters of simulated single SSD are configured based on a state-of-art 3D TLC SSD, as shown in Table 4. Second, one-shot programming is added in SSDSim. Third, multiple modified 3D TLC SSDs employing ReSSD design are organized as a RAID 5, where the RAID 5 controller and its configurations are implemented based on */drivers/md/* module of Linux Kernel *5.0.2* Version. The bottom of Table 4 shows the configurations of RAID 5. In addition, other default management mechanisms are also implemented in each SSD controller, including page mapping based flash translation layer [1], static wear leveling [9], static data allocation [28], and greedy based GC [21]. The evaluated latencies of reprogram and read operations are presented in the middle two rows of Table 4. The results show that program and reprogram latencies on reprogrammable block are slightly smaller than natural program latency of 3D TLC SSD. This is because the voltage distance of each program or reprogram operation is reduced. Similarly, the read latency is reduced as well thanks to the reduced number of sensing operations in each reprogrammed cell.

For the hotness determination, three update time points that divides hotness into four levels are set to 30, 60 and 120 minutes, respectively. Within 120 minutes, most data in selected server workloads are updated according to [14]. For the baseline, host write requests are distributed to blocks based on their hotness as well.

Workloads: A set of MSR Cambridge workloads [51] are used as user data workloads in RAID 5 system. For parity requests, they are generated based on */drivers/md/* module of Linux Kernel *5.0.2* Version. The characteristics of workloads are presented in Table 5. Note that, W Pct, U Pct, WS, RS, WV and RV indicate the percentages of write requests, the percentages of update requests, average write/read request size and write/read data volume.

Table 4: Parameters of Simulated 3D TLC SSD based RAID 5 System [14, 48, 59].

Parameters	Value	Parameters	Value
Number of Channels	4	Layers per Block	64
Chips per Channel	4	T_r (in Page)	66 us
Dies per Chip	2	T_p (in Page)	3000 us
Planes per Die	2	T_e (in Block)	10 ms
Blocks per Plane	1536	Data Transfer Rate	400 MB/s
Pages per Block	384	GC Threshold	8%
Page Size	16 KB	Initial Data	92%
Reprogram Scheme	Value	Reprogram Scheme	Value
$T_{regular\ p}$ (in Page)	2675 us	T_{rep} (in Page)	2705 us
T_r (in Page)	53 us		
RAID 5 Configuration	Value	RAID 5 Configuration	Value
Number of SSDs	4	Chunk per Stripe	4
Pages per Chunk	64	Max Stripe Count	256

Table 5: The Characteristics of Evaluated Workloads.

Workloads	W Pct	U Pct	WS(KB)	RS(KB)	WV(GB)	RV(GB)
HM_0	75.1%	91.7%	11.2	11.7	7.95	2.74
HM_1	3.1%	74.3%	22.9	18.1	0.35	8.54
PRN_0	89.3%	81.0%	14.0	26.6	11.77	2.69
PRN_1	31.1%	61.1%	13.8	17.7	4.05	11.54
PROJ_1	9.4%	25.3%	22.2	43.2	1.97	36.93
PROJ_3	9.8%	36.6%	30.1	15.1	2.78	12.87
PROJ_4	3.7%	61.7%	14.8	10.3	0.51	9.37
RSRCH	90.9%	97.7%	12.7	15.7	10.90	1.35
SRC1_2	84.5%	98.3%	44.9	16.9	35.81	2.48
SRC2_0	86.4%	95.2%	11.0	12.6	8.99	1.63
SRC2_2	71.8%	51.6%	57.8	88.5	39.18	23.56
STG_0	76.7%	97.3%	12.7	33.6	9.19	7.39
WDEV_0	80.4%	96.6%	12.1	16.5	9.21	3.05
PRXY_0	97.0%	97.9%	6.3	9.7	5.75	0.27
USR_0	63.1%	95.0%	13.6	47.1	8.07	16.42

6.2 Experimental Results

6.2.1 Flash Page Consumption Analysis. Physical Page Consumption (PPC) is evaluated to show the number of flash pages being written during run time. Results are presented in Figure 11, where the number of PPC of RSS-RAID 5 system is evaluated and normalized to the baseline. On average, compared with the baseline, PPC of RSS-RAID is reduced to 70.1%, which can be broken into three parts, including write requests induced PPC (flash page consumption caused by normal write requests and reprogram based write requests), GC

writes (flash page consumption caused by GC induced valid page writes) induced PPC and Fully Invalidated writes induced PPC (Fully Invalidated Writes refer to the process of moving cooling pages from candidate blocks to normal TLC blocks). On average, these three pages account 34.5%, 51.1% and 14.4% of total PCC of RSS-RAID, respectively.

Compared with the baseline, write requests induced PPC of RSS-RAID is reduced 68.4%. As Fully Invalidated writes can reduce GC cost by moving valid pages in advance, the total PPC resulted from Fully Invalidated writes and GC writes in RSS-RAID is reduced to 72.1%, compared with the baseline. Based on the results, one can see that RSS-RAID can achieve significant PPC decreases. The reasons come from two aspects: First, reprogram operation is able to increase the number of pages written in a word line. Second, reprogram operation can also reduce GC count and cost by storing more data into each block and erasing a block with less valid page migrations (details can be found in Section 6.2.2.)

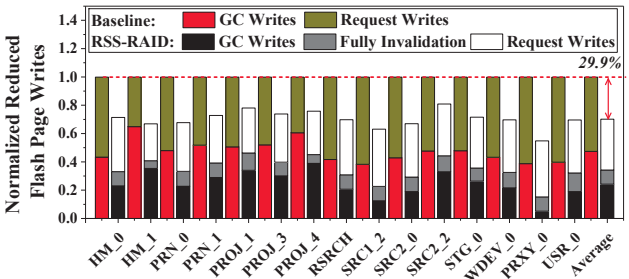


Figure 11: Normalized physical flash page decrease.

6.2.2 GC Evaluation and Analysis. In this subsection, first, total GC numbers of the baseline and RSS-RAID are evaluated. There are two types of GCs in flash memory, including GC with valid page migration and direct GC (erasing a block without valid page inside). If there are more blocks that can be reclaimed by direct GCs, the performance can be improved due to less valid page migration cost. On average, the total GC number of RSS-RAID is reduced by 30.6%, of which the number of GC with valid page migration is reduced by 49.4% and the number of direct GC is increased by 10.9 times. The reasons are as follows: First, reprogram operation can reduce physical page consumption, triggering fewer GC activities; Second, reprogrammable blocks can be erased by direct GCs when cooling valid pages in blocks have been moved out in advance by Fully Invalidation.

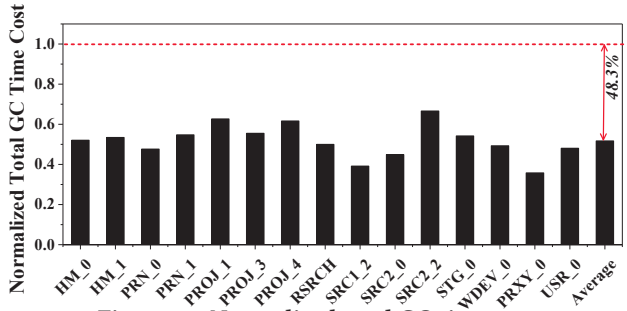


Figure 12: Normalized total GC time cost.

As the total number of GC decreases significantly, total GC time cost can also be reduced significantly. As shown in Figure 12, on average, total GC time cost is reduced by 48.3%. This is because the total GC count is reduced and more GCs are processed without valid page migration. As a result, the effectiveness of GC can be improved by adopting RSS-RAID.

6.2.3 Endurance Improvement Analysis. Endurance of SSD is quantified by P/E cycles. As additional reprogram operation induced voltage stress is quite small, a metric, termed Page Writes per Erase (PWE), is used to identify the endurance of flash memory [14]. PWE is used to show the number of page writes between erases. If PWE is increased, then total number of erase operations of RSS-RAID system can be reduced, improving the endurance. The evaluated results are presented in Figure 13, where PWE of RSS-RAID is increased by 30.3% on average. The reason is that, RSS-RAID not only increases the number of bits stored in a cell, but also significantly reduces total erase count of GC activities. With the increases of PWE, the endurance of RSS-RAID is improved.

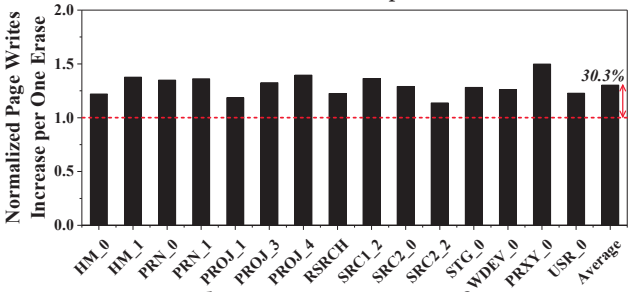


Figure 13: Endurance improvement of RSS-RAID.

6.2.4 Performance Impact Analysis. In order to reveal reprogram operation induced impact on read and write performance of 3D TLC SSD based RAID 5 system, the average read and write latencies are evaluated and presented in Figure 14. In this figure, evaluated latency is normalized to the baseline and one can observe that the read performance of RSS-RAID is slightly reduced compared with the baseline. This is because only two read sensing operations are required for reading a reprogrammed page.

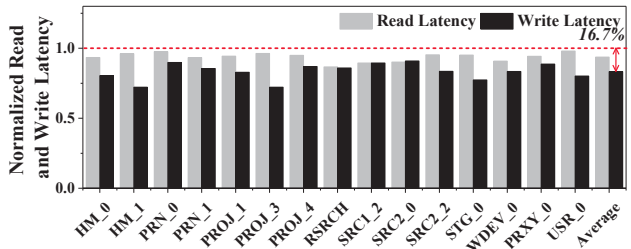


Figure 14: Normalized read/write latency of RSS-RAID.

For write performance, on average, the write latency of RSS-RAID is reduced by 16.7% compared with the baseline. Although program and reprogram operations executed in reprogrammable blocks can slightly reduce the latency, the benefit from one-shot programming is reduced due to less data being programmed simultaneously, hindering the write performance. However, the benefit from GC optimization can recover this negative impact, bringing better write performance of RSS-RAID.

6.2.5 Increased Effective Capacity. In this subsection, after running each workload, the effective capacity represented by the number of free flash pages in RAID 5 system is evaluated. Notice that, in RSS-RAID scheme, if a block is reprogrammable, all word lines in this block are regarded as MLC word line that only contains two

pages. In Figure 15, the number of free flash pages of RSS-RAID is normalized to that of the baseline. The number of free flash pages of RSS-RAID is increased by 7.71% on average, compared with the baseline. Furthermore, RSS-RAID can achieve at least 8% free page number increases for more than half of workloads (8 out of 15). This is because each flash cell is able to store more bits before an erase operation.

However, for other workloads, such as PROJ_4, the number of free flash pages of RSS-RAID is only increased by 1.25%, on average. The reason is that, since the numbers of total write requests and update requests in PROJ_4 are small, only a few number of reprogrammable blocks can be frequently accessed and reprogrammed to achieve effective capacity increases while other blocks are barely accessed.

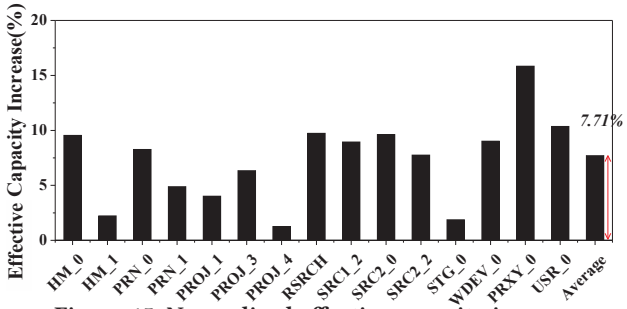


Figure 15: Normalized effective capacity increase.

6.2.6 Evaluation of MLC mode RAID. In this subsection, conventional 3D TLC SSD is used as 3D MLC mode SSD by degenerating TLC to MLC. The most direct impact is that the capacity of 3D MLC mode SSD is reduced by 33.3% as only two bits are used per cell. The performance and endurance of MLC mode SSD can be improved because of its advantages in faster program speed and larger P/E cycles [47, 48].

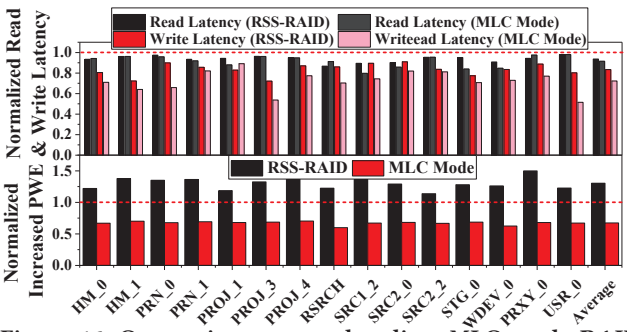


Figure 16: Comparison among baseline, MLC mode RAID and RSS-RAID.

First, the latency of programming MLC page via one-shot programming is set to 2200 us. Results are presented in the top of Figure 16, where the read and write latencies of RSS-RAID and MLC mode RAID are normalized to the baseline. On average, the write latency of MLC mode RAID is reduced by 27.8% while the write latency of RSS-RAID is reduced by 16.7%, compared with the baseline. The main reason is that the latency of program operation on MLC is smaller than that of conventional 3D TLC SSD and RSS-RAID 5 system. However, since each block in MLC mode SSD contains fewer flash pages, GC may be triggered more frequently, delaying more requests and increasing request latency. Therefore, in some workloads (e.g., PROJ_1), the write latency of MLC mode SSD is even larger than that of RSS-RAID. For read performance, MLC mode RAID and RSS-RAID achieve 8.5% and 6.3% read latency reduction on average compared with the baseline.

Second, the results of PWE are evaluated and presented in the bottom of Figure 16. Compared with the baseline and RSS-RAID, average PWE of MLC mode SSD is reduced by 32.7% and 48.4%, respectively. The reason comes from that when only two bits are used in a cell, the number of erase operation increases in MLC mode SSD. Although MLC mode SSD has better endurance, RSS-RAID can slow down the wearing process of TLC based SSD by reducing the number of erase operations.

7 RELATED WORKS

Several previous works have been proposed to write more data in a word line. They can be divided into three types: First, Kim et al. [37] and Kim et al. [36] proposed a subpage program method, which can program a large flash page multiple times with small write operation so that data can be stored in flash page at a finer granularity. Second, multiple previous works adopt write-once memory (WOM) code [42, 43, 58] to enlarge the capacity of SSDs. However, WOM code requires the assistance of ECC space for holding additional information, which will weaken the capability of ECC. Third, [14] and [10] are proposed to reprogram SLC by narrowing voltage margin between two states. In this case, each cell can be reprogrammed multiple times by invalidating previous values in the same cell. These works are proposed oriented 2D SLC SSDs, where a large voltage margin exist for serving reprogram operation. For 2D TLC SSDs, [15] shows that TLC can be reprogrammed as well. The main drawback of reprogramming 2D TLC SSDs is the reliability, which comes from the cell to cell interference among flash cells. To solve this problem, refresh operations in SSDs are applied. The characteristics of reprogramming 3D TLC SSDs has not been discussed in all previous works, which has completely different characteristics compared to 2D SSDs.

8 CONCLUSION

In this work, the feasibility of reprogram operation in 3D TLC SSDs is verified. Then, a reprogrammable SSD design is proposed in 3D TLC SSDs, which aims to reduce capacity consumption, improve the endurance and performance of 3D TLC SSDs. By applying the proposed approach, a case study for 3D TLC SSD based RAID 5 system is evaluated. As a result, the proposed approach can improve the endurance by 30.3%, boost write performance by 16.7% and increase effective capacity by 7.71%, respectively, which is realized with negligible overhead compared with conventional SSD based RAID 5 system.

ACKNOWLEDGMENTS

This work is supported in part by NSF No. 1910413,1617071,1725657 and 1718080, NSFC No. 61572411, 61772092. We also thank anonymous reviewers for their constructive feedback and Tyler Garrett for his preliminary work. This work was carried out while Congming Gao was visiting the University of Pittsburgh on a CSC scholarship.

REFERENCES

[1] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D Davis, Mark S Manasse, and Rina Panigrahy. 2008. Design tradeoffs for SSD performance.. In *Annual Technical Conference*. USENIX.

[2] Svante Arrhenius. 1889. Über die Reaktionsgeschwindigkeit bei der Inversion von Rohrzucker durch Säuren. In *Zeitschrift für physikalische Chemie*. De Gruyter Oldenbourg.

[3] JEDEC SOLID STATE TECHNOLOGY ASSOCIATION. 2011. *JEDEC STANDARD:Stress-Test-Driven Qualification of Integrated Circuits JESD47H.01*. JEDEC.

[4] Yu Cai, Saugata Ghose, Erich F Haratsch, Yixin Luo, and Onur Mutlu. 2017. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. In *Proceedings of the IEEE*. IEEE.

[5] Yu Cai, Saugata Ghose, Yixin Luo, Ken Mai, Onur Mutlu, and Erich F Haratsch. 2017. Vulnerabilities in MLC NAND flash memory programming: experimental analysis, exploits, and mitigation techniques. In *International Symposium on High Performance Computer Architecture*. IEEE.

[6] Yu Cai, Yixin Luo, Saugata Ghose, and Onur Mutlu. 2015. Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery. In *Dependable Systems and Networks*. IEEE.

[7] Yu Cai, Onur Mutlu, Erich F Haratsch, and Ken Mai. 2013. Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation. In *The 31st International Conference on Computer Design*. IEEE.

[8] Li-Pin Chang, Tei-Wei Kuo, and Shi-Wu Lo. 2004. Real-time garbage collection for flash-memory storage systems of real-time embedded systems. In *Transactions on Embedded Computing Systems*. ACM.

[9] Yuan-Hao Chang, Jen-Wei Hsieh, and Tei-Wei Kuo. 2007. Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design. In *Design Automation Conference*.

[10] Yu-Ming Chang, Yung-Chun Li, Ping-Hsien Lin, Hsiang-Pang Li, and Yuan-Hao Chang. 2016. Realizing erase-free SLC flash memory with rewritable programming design. In *Hardware/Software Codesign and System Synthesis*. IEEE.

[11] SZ Chen and Don Towsley. 1993. The design and evaluation of RAID 5 and parity striping disk array architectures. In *booktitle of Parallel and Distributed Computing*. Elsevier.

[12] Tseng-Yi Chen, Yuan-Hao Chang, Yuan-Hung Kuan, and Yu-Ming Chang. 2017. VirtualGC: Enabling erase-free garbage collection to upgrade the performance of rewritable SLC NAND flash memory. In *Annual Design Automation Conference*. ACM.

[13] Wei-Chen Chen, Hang-Ting Lue, Kuo-Pin Chang, Yi-Hsuan Hsiao, Chih-Chang Hsieh, Yen-Hao Shih, and Chih-Yuan Lu. 2014. Study of the programming sequence induced back-pattern effect in split-page 3D vertical-gate (VG) NAND flash. In *Proceedings of Technical Program-2014 International Symposium on VLSI Technology, Systems and Application*. IEEE.

[14] Wonil Choi, Mohammad Arjomand, Myoungsoo Jung, and Mahmut Kandemir. 2017. Exploiting data longevity for enhancing the lifetime of flash-based storage class memory. In *Proceedings of the Measurement and Analysis of Computing Systems*. ACM.

[15] Wonil Choi, Myoungsoo Jung, and Mahmut Kandemir. 2018. Invalid Data-Aware Coding to Enhance the Read Performance of High-Density Flash Memories. In *Annual IEEE/ACM International Symposium on Microarchitecture)*. IEEE.

[16] Ching-Che Chung and Hao-Hsiang Hsu. 2014. Partial parity cache and data cache management method to improve the performance of an SSD-based RAID. In *IEEE Transactions on Very Large Scale Integration Systems*. IEEE.

[17] Wikipedia contributors. 2019. Pareto principle. https://en.wikipedia.org/w/index.php?title=Pareto_principle&oldid=904233270.

[18] Western Digital. 2017. Western Digital Announces Industry's First 96-Layer 3D NAND Technology. https://www.westerndigital.com/company/newsroom/press-releases/2017/2017-06-27-western-digital-announces-industrys-first-96-layer-3d-nand-technology.

[19] Congming Gao, Yejia Di, Aosong Deng, Duo Liu, Cheng Ji, Jason Chun Xue, and Liang Shi. 2018. F2FS Aware Mapping Cache Design on Solid State Drives. In *Non-Volatile Memory Systems and Applications Symposium*. IEEE.

[20] Congming Gao, Liang Shi, Yejia Di, Qiao Li, Chun Jason Xue, and Edwin H.-M. Sha. 2018. An Efficient Cache Management Scheme for Capacitor Equipped Solid State Drives. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*. ACM.

[21] Congming Gao, Liang Shi, Yejia Di, Qiao Li, Chun Jason Xue, Kaijie Wu, and Edwin Sha. 2018. Exploiting Chip Idleness for Minimizing Garbage Collection Induced Chip Access Conflict on SSDs. In *Transactions on Design Automation of Electronic Systems*. ACM.

[22] Congming Gao, Liang Shi, Cheng Ji, Yejia Di, Kaijie Wu, Chun Jason Xue, and Edwin H-M Sha. 2018. Exploiting parallelism for access conflict minimization in flash-based solid state drives. In *Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE.

[23] Congming Gao, Liang Shi, Jason Chun Xue, Cheng Ji, Jun Yang, and Youtao Zhang. 2019. Parallel all the time: Plane Level Parallelism Exploration for High Performance SSDs. In *Mass Storage Systems and Technologies*. IEEE.

[24] Congming Gao, Liang Shi, Mengying Zhao, Chun Jason Xue, Kaijie Wu, and Edwin H-M Sha. 2014. Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives. In *Mass Storage Systems and Technologies*. IEEE.

[25] Erich F. Haratsch. 2017. Controller Technologies for Managing 3D NAND Flash Memories. In *Seagate*. Flash Memory Summit.

[26] Chien-Chung Ho, Yung-Chun Li, Yuan-Hao Chang, and Yu-Ming Chang. 2018. Achieving defect-free multilevel 3D flash memories with one-shot program design. In *Design Automation Conference*. IEEE.

[27] Joel Hruska. 2018. Samsung Now Mass Producing 96-Layer 3D NAND. https://www.extremetech.com/computing/273167-samsung-now-mass-producing-96-layer/-3d-nand.

[28] Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Hao Luo, and Shuping Zhang. 2011. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity. In *Proceedings of the international conference on Supercomputing*. ACM.

[29] Soojun Im and Dongkun Shin. 2011. Flash-aware RAID techniques for dependable and high-performance flash memory SSD. In *Transactions on Computers*. IEEE.

[30] Jaehoon Jang, Han-Soo Kim, Wonseok Cho, Hoosung Cho, Jinho Kim, Sun Il Shim, Jae-Hun Jeong, Byoung-Keun Son, Dong Woo Kim, Jae-Joo Shim, et al. 2009. Vertical cell array using TCAT (Terabit Cell Array Transistor) technology for ultra high density NAND flash memory. In *Symposium on VLSI Technology*. IEEE.

[31] Jaeyong Jeong, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim. 2014. Life-time improvement of NAND flash-based storage systems using dynamic program and erase scaling.. In *Conference on File and Storage Technologies*. USENIX.

[32] Timothy J Johnson. 1998. Raid-5 parity generation and data reconstruction. US Patent 5,805,788.

[33] Tae-Sung Jung, Young-Joon Choi, Kang-Deog Suh, Byung-Hoon Suh, Jin-Ki Kim, Young-Ho Lim, Yong-Nam Koh, Jong-Wook Park, Ki-Jong Lee, Jung-Hoon Park, et al. 1996. A 117-mm/sup 2/3.3-V only 128-Mb multilevel NAND flash memory for mass storage applications. In *IEEE booktitle of Solid-State Circuits*. IEEE.

[34] C. Kim, J. Cho, W. Jeong, I. Park, H. Park, D. Kim, D. Kang, S. Lee, J. Lee, W. Kim, J. Park, Y. Ahn, J. Lee, J. Lee, S. Kim, H. Yoon, J. Yu, N. Choi, Y. Kwon, N. Kim, H. Jang, J. Park, S. Song, Y. Park, J. Bang, S. Hong, B. Jeong, H. Kim, C. Lee, Y. Min, I. Lee, I. Kim, S. Kim, D. Yoon, K. Kim, Y. Choi, M. Kim, H. Kim, P. Kwak, J. Ihm, D. Byeon, J. Lee, K. Park, and K. Kyung. [n. d.]. 11.4 A 512Gb 3b/cell 64-stacked WL 3D V-NAND flash memory. In *International Solid-State Circuits Conference*. IEEE.

[35] Hyunsuk Kim, Su-Jin Ahn, Yu Gyun Shin, Kyupil Lee, and Eunseung Jung. 2017. Evolution of NAND Flash Memory: from 2D to 3D as a storage market leader. In *Memory Workshop*. IEEE.

[36] Jung-Hoon Kim, Sang-Hoon Kim, and Jin-Soo Kim. 2015. Subpage programming for extending the lifetime of nand flash memory. In *Design, Automation & Test in Europe Conference & Exhibition*. IEEE.

[37] Myungsuk Kim, Jaehoon Lee, Sungjin Lee, Jisung Park, and Jihong Kim. 2017. Improving performance and lifetime of large-page NAND storages using erase-free subpage programming. In *Design Automation Conference*. IEEE.

[38] Seungjae Lee, Jin-yub Lee, Il-han Park, Jongyeol Park, Sung-won Yun, Min-su Kim, Jong-hoon Lee, Minseok Kim, Kangbin Lee, Taeeun Kim, et al. 2016. 7.5 A 128Gb 2b/cell NAND flash memory in 14nm technology with tPROG= 640μs and 800MB/s I/O rate. In *International Solid-State Circuits Conference*. IEEE.

[39] Sang-Won Lee, Bongki Moon, Chanik Park, Jae-Myung Kim, and Sang-Woo Kim. 2008. A case for flash memory ssd in enterprise database applications. In *SIGMOD international conference on Management of data*. ACM.

[40] Chu Li, Dan Feng, Yu Hua, and Fang Wang. 2016. Improving raid performance using an endurable SSD cache. In *International Conference on Parallel Processing*. IEEE.

[41] Chun-yi Liu, Jagadish Kotra, Myoungsoo Jung, and Mahmut Kandemir. 2018. PEN: design and evaluation of partial-erase for 3D NAND-based high density SSDs. In *Conference on File and Storage Technologies*. USENIX.

[42] Fabio Margaglia and André Brinkmann. 2015. Improving MLC flash performance and endurance with extended P/E cycles. In *Symposium on Mass Storage Systems and Technologies*. IEEE.

[43] Fabio Margaglia, Gala Yadgar, Eitan Yaakobi, Yue Li, Assaf Schuster, and André Brinkmann. 2016. The Devil Is in the Details: Implementing Flash Page Reuse with WOM Codes.. In *Conference on File and Storage Technologies*.

[44] Fei Meng, Li Zhou, Xiaosong Ma, Sandeep Uttamchandani, and Deng Liu. 2014. vCacheShare: Automated Server Flash Cache Space Management in a Virtualization Environment.. In *Annual Technical Conference*. USENIX.

[45] Rino Micheloni et al. 2016. *3D Flash memories*. Springer.

[46] Rino Micheloni, Luca Crippa, and Alessia Marelli. 2010. *Inside NAND flash memories*. Springer Science & Business Media.

[47] Rino Micheloni, Alessia Marelli, and Kam Eshghi. 2013. *Inside solid state drives (SSDs)*. Springer.

[48] Micron. 2018. MT29F1T08EEHAFJ4-3R 3D TLC NAND Flash. https://www.micron.com/products/nand-flash/3d-nand/part-catalog/mt29f1t08eehafj4-3r.

[49] Neal Mielke, Hanmant Belgal, Ivan Kalastirsky, Pranav Kalavade, Andrew Kurtz, Qingru Meng, Nick Righos, and Jie Wu. 2004. Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling. In *Transactions on Device and Materials Reliability*. IEEE.

[50] Sangwhan Moon and A. L. Narasimha Reddy. 2013. Don't Let RAID Raid the Lifetime of Your SSD Array. In *Workshop on Hot Topics in Storage and File Systems*. USENIX.

[51] Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron. 2009. Migrating server storage to SSDs: analysis of tradeoffs. In *European conference on Computer systems*. ACM.

[52] Pravin Prabhu, Ameen Akel, Laura M Grupp, S Yu Wing-Kei, G Edward Suh, Edwin Kan, and Steven Swanson. 2011. Extracting device fingerprints from flash memory by exploiting physical variations. In *International Conference on Trust and Trustworthy Computing*. Springer.

[53] Narges Shahidi, Mohammad Arjomand, Myoungsoo Jung, Mahmut T Kandemir, Chita R Das, and Anand Sivasubramaniam. 2016. Exploring the potentials of parallel garbage collection in ssds for enterprise storage systems. In *International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press.

[54] Liang Shi, Yejia Di, Mengying Zhao, Chun Jason Xue, Kaijie Wu, and Edwin H-M Sha. 2015. Exploiting process variation for write performance improvement on NAND flash memory storage systems. In *Transactions on Very Large Scale Integration (VLSI) Systems*. IEEE.

[55] Jonathan Tjioe, Andrés Blanco, Tao Xie, and Yiming Ouyang. 2012. Making garbage collection wear conscious for flash SSD. In *Networking, Architecture and Storage*. IEEE.

[56] Catriona Tullo and James Hurford. 2003. Modelling Zipfian distributions in language. In *Proceedings of language evolution and computation workshop/course at ESSLLI*.

[57] Benny Van Houdt. 2013. Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data. In *Performance Evaluation*. Elsevier.

[58] Gala Yadgar, Eitan Yaakobi, and Assaf Schuster. 2015. Write Once, Get 50% Free: Saving SSD Erase Costs Using WOM Codes. In *Conference on File and Storage Technologies*.

[59] YEESTOR. 2018. YS9083XT/YS9081XT SSD Platform. http://www.yeestor.com/index-product-info-id-946-cid-33-pid-3-infopid-33.html.

[60] Jung H Yoon, Ranjana Godse, Gary Tressler, and Hillery Hunter. 2017. 3D-NAND scaling & 3D-SCM–Implications to Enterprise Storage. In *Flash Memory Summit*.