A Reliability-based Coding Strategy to Suppress Lateral Charge Migration for TLC NAND Flash Memory

Chin-Hsien Wu
Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology
Taipei, Taiwan
chwu@mail.ntust.edu.tw

Yan-Qi Liao
Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology
Taipei, Taiwan
M10902412@mail.ntust.edu.tw

Yung-Cheng Huang
Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology
Taipei, Taiwan
M11002148@mail.ntust.edu.tw

ABSTRACT

In recent years, TLC NAND flash memory has emerged as the dominant storage medium, primarily due to its high storage density, cost-effectiveness, and widespread adoption in various applications. However, despite these advantages, TLC NAND flash memory suffers from significant reliability challenges, especially when data is retained for an extended period. One of the most critical issues is retention errors, which tend to accumulate over time, eventually surpassing the error correction capabilities of ECC (Error Correction Code). This problem is particularly severe in the early stages of usage, when the memory has undergone fewer program/erase (P/E) cycles and is highly vulnerable to retention errors caused by lateral charge migration between adjacent cells. In this paper, we propose a reliability-based coding strategy to suppress the lateral charge migration and minimize retention errors during the early lifespan of TLC NAND flash memory. By leveraging an optimized coding approach, the proposed method effectively enhances data reliability without introducing significant computational overhead. Experimental results demonstrate that our strategy significantly outperforms previous methods, providing a substantial reduction in retention errors while maintaining efficient storage performance.

CCS CONCEPTS

• **Information systems** → **Flash memory**; • **Software and its engineering** → **Secondary storage**; • **Computer systems organization** → *Embedded software*;

KEYWORDS

NAND Flash Memory, Reliability, Retention Errors, Lateral Charge Migration

1 INTRODUCTION

In recent years, solid-state drives (SSDs) have become the dominant storage medium, primarily due to their adoption of TLC NAND flash memory based on charge trap (CT) technology. With the continuous increase in cell storage capacity, NAND flash memory has evolved into several types, including Single-Level-Cell (SLC), Multi-Level-Cell (MLC), and Triple-Level-Cell (TLC). Among these, TLC NAND flash memory is the most widely used due to its optimal balance between storage density and cost-effectiveness. In TLC NAND flash memory, each cell is capable of storing three bits of data, corresponding to eight distinct voltage states. However, as cell density increases, the probability of errors also rises, leading to various reliability concerns [8, 17, 25]. Common issues include retention errors [4, 16, 20, 24] and read disturb errors [1, 4, 6], both of which can degrade SSD reliability and, in severe cases, cause permanent data loss. Among these, retention errors are recognized as the primary cause of NAND flash reliability degradation [16, 20]. These errors occur when programmed flash cells store data for extended periods, leading to charge leakage [4, 24]. In TLC NAND flash memory with charge trap technology, charge leakage can be categorized into two types: lateral charge migration and vertical charge de-trap. Notably, lateral charge migration has a significant impact in the early stages of SSD usage, making it a crucial challenge to address in newly deployed TLC NAND flash-based products. Therefore, developing effective techniques to suppress lateral charge migration is essential to improving NAND flash reliability and ensuring long-term data integrity.

In this paper, we propose a reliability-based coding strategy specifically designed to suppress lateral charge migration in TLC NAND flash memory during its early usage phase when program/erase (P/E) cycles are still low. The primary objective of this strategy is to effectively minimize retention errors caused by lateral charge migration, which is a critical factor in NAND flash reliability. This is especially important when a system needs to store a large volume of crucial data for extended periods, ensuring data integrity and preventing early degradation. Lateral charge migration has the most significant impact when low-potential states are positioned adjacent to high-potential states across neighboring word lines, leading to increased error rates. To address this issue, the proposed strategy integrates a novel coding mechanism along with an enhancement skill, ensuring that data is structured in a way that significantly reduces the likelihood of charge interference. This approach provides the following key contributions:

- We propose a coding concept designed to minimize the need for additional read operations while efficiently determining appropriate cell states at suitable encoding positions. The core idea of this coding concept is to strategically arrange data by placing low-potential states toward the left and high-potential states toward the right as consistently as possible. By following this structured arrangement, the method aims to reduce the high potential differences between adjacent

word lines, thereby significantly mitigating the effects of lateral charge migration. This design not only enhances data reliability but also helps in extending the operational lifespan of TLC NAND flash memory by reducing unnecessary charge redistribution effects.

- Then, we systematically analyze three distinct sequences—the linear sequence, the Fibonacci sequence, and the exponential sequence—to thoroughly investigate their respective impacts on assigning appropriate weights to different cell states at various encoding positions. Since each of these sequences exhibits unique mathematical properties, they provide different weighting strategies that influence the encoding process. Additionally, because these sequences can be precomputed, their corresponding weights can be efficiently stored in a compact weight table, making them highly suitable for resource-constrained embedded systems. This approach ensures that weight assignments are both computationally efficient and memory-friendly, enabling seamless integration into low-power NAND flash memory architectures.
- Finally, we incorporate an enhancement skill designed to further mitigate retention errors, ensuring a significant improvement in data reliability. This enhancement skill seamlessly integrates with the proposed coding concept, particularly in scenarios where minimizing retention errors takes precedence over space efficiency. By effectively eliminating high potential states, this enhancement skill reduces charge migration, which is a primary cause of retention errors in TLC NAND flash memory. As a result, the combined approach of the proposed coding concept and the enhancement skill leads to a substantial reduction in retention errors, thereby greatly enhancing the overall reliability and longevity of NAND flash memory systems.

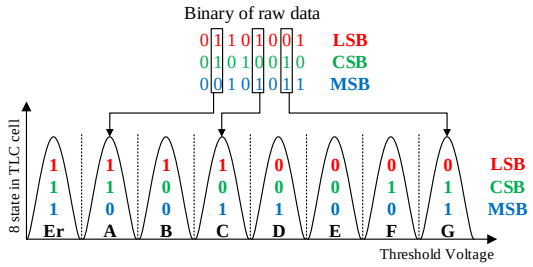
According to the experimental results, the proposed coding concept effectively reduces retention errors by approximately 70% on average. Furthermore, when combined with the enhancement skill, the proposed approach achieves an even greater reduction, lowering retention errors by about 85.3% on average.

The remainder of this paper is structured as follows. Section 2 presents the necessary background knowledge and discusses related work. Section 3 outlines the research motivation behind this study. In Section 4, we introduce a reliability-based coding strategy to suppress the lateral charge migration for TLC NAND flash memory. Section 5 evaluates the performance of the proposed strategy through experimental analysis. Finally, Section 6 concludes the paper with a summary of key findings and potential directions for future research.

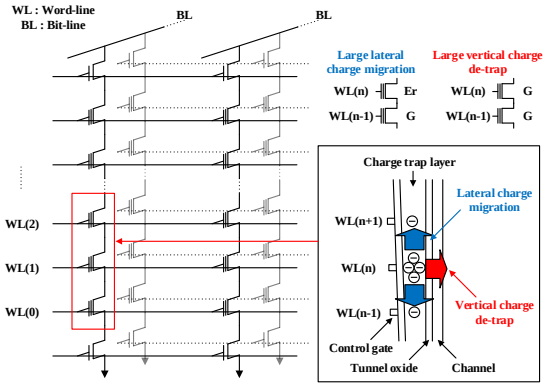
2 BACKGROUND KNOWLEDGE AND RELATED WORK

2.1 TLC NAND Flash Memory

In TLC (triple-level cell) NAND flash memory, each cell is capable of storing 3 bits of data, which are represented by the following states: 111 (Er state), 110 (A state), 100 (B state), 101 (C state), 001 (D state), 000 (E state), 010 (F state), and 011 (G state). Typically, a word-line serves as the fundamental unit for programming operations and consists of three pages: the least significant bit (LSB) page, the central significant bit (CSB) page, and the most significant bit (MSB) page. Since each TLC cell within a word-line represents 3 bits of data, it contributes to these three pages collectively. Fig. 1(a) illustrates how the binary representation of raw data is distributed across the LSB, CSB, and MSB pages through TLC cells, demonstrating the relationship between the stored data and the corresponding voltage levels of the memory cells.



(a) LSB, CSB and MSB pages are represented by a TLC cell



(b) Two types of charge leakage: lateral charge migration and vertical charge de-trap

Figure 1: A TLC cell and its two types of charge leakage.

2.2 Retention Errors

When TLC NAND flash memory cells are programmed and stored for an extended period, they are susceptible to retention errors due to charge leakage [4, 24]. This charge leakage causes shifts in the threshold voltage of memory cells, with higher voltage states (e.g., G state) being more vulnerable to voltage shifts compared to lower voltage states (e.g., Er state). Furthermore, regardless of the program/erase (P/E) endurance of NAND flash memory using charge trap material (CT), retention errors remain the predominant type of errors in NAND flash memory [16]. Additionally, retention errors are a critical concern in various operational scenarios. Studies have shown that they become dominant even under moderate usage conditions, such as below 10K P/E cycles, after just three days of retention, and following 900K read operations [20]. These observations highlight the significance of addressing retention errors to enhance the reliability of TLC NAND flash memory.

In TLC NAND flash memory utilizing charge trap material (CT), charge leakage can be categorized into two primary types: lateral charge migration and vertical charge de-trap [21, 31], as illustrated in Fig. 1(b). Lateral charge migration occurs when there is a significant potential difference between adjacent word-lines, such as the pair of Er and G states. This phenomenon is particularly pronounced even when the NAND flash memory has undergone relatively few program/erase (P/E) cycles, making it a major concern during the early usage stage. On the other hand, vertical charge de-trap primarily occurs at higher P/E cycles, where high potential states (e.g., G and G states) between neighboring word-lines lead to gradual charge loss [23, 27]. Consequently, retention errors remain a significant reliability issue even when TLC NAND flash memory is in its early usage phase with low P/E cycles, primarily due to lateral charge migration. Addressing this challenge is critical to ensuring the longevity and stability of NAND flash memory.

2.3 Error Mitigation

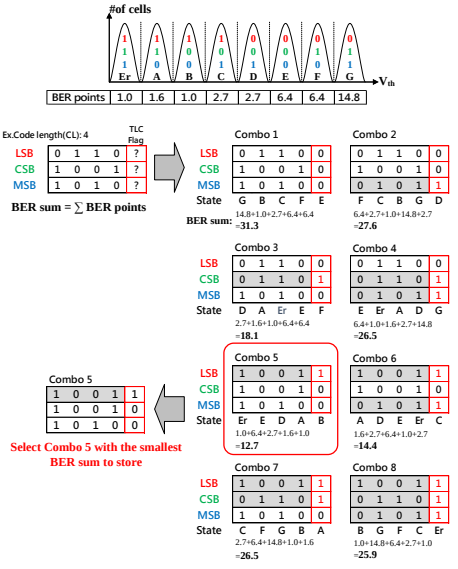


Figure 2: WBVM uses a bit-flip method to flip or non-flip each code length and calculates the smallest BER sum (i.e., $\sum BER\ points$) as the best store combination according to each state's BER points [11].

The conventional approaches for mitigating the bit error rate (BER) in NAND flash memory can be categorized into four primary techniques: error correction coding (ECC), data rewriting, refreshing and refilling, reference voltage shifting, and data coding. ECC, such as error detection [17], BCH and LDPC codes [9, 18, 19, 30], is widely employed to detect and correct errors by utilizing redundant bits, enhancing overall data integrity. Data rewriting, refreshing and refilling techniques [8, 32] periodically relocate frequently accessed (read-hot) pages from a victim block to another flash block with fewer read cycles or just refilling a cell, thereby minimizing errors. Reference voltage shifting [14] dynamically adjusts the read reference voltage to compensate for threshold voltage shifts, reducing the probability of misread cells. Additionally, data coding techniques [11, 22, 23, 27, 28] optimize data storage patterns to mitigate BER by distributing data in a manner that minimizes charge leakage and disturbance effects. These methods collectively enhance the reliability and longevity of NAND flash memory, making them essential for improving the stability of flash-based storage systems.

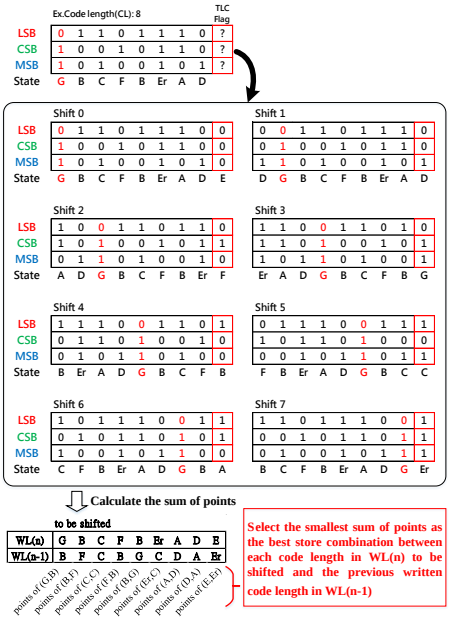


Figure 3: VN can select the smallest sum of points as the best store combination between each code length in WL(n) to be shifted and the previous written code length in WL(n-1) to decrease the high potential difference between any neighbor word-lines [23].

Data coding is an effective approach to reducing the bit error rate (BER) in NAND flash memory by minimizing the occurrence of error-prone states, as seen in methods such as AC [28], WBVM [11], VN [23], DVDS [27], and LRPER [22]. As illustrated in Fig. 2, WBVM employs a bit-flip technique to determine whether each code length should be flipped or left unchanged. It then calculates the smallest BER sum (i.e., $\sum BER\ points$) to find the optimal storage combination based on the BER points assigned to each state. Higher potential states, such as F and G, exhibit greater BER points and are more susceptible to errors, whereas lower potential states, like Er and A, have fewer BER points. For example, in Fig. 2, Combo 5 achieves the lowest BER sum (12.7) by flipping only the LSB bit, with a TLC flag of 100. WBVM effectively reduces retention errors by limiting the occurrence of high potential states like F and G. LRPER follows a similar principle but operates as a lightweight version of WBVM. It employs a single flag bit to flip or preserve the LSB bit of each code length, thereby reducing the space overhead of flag cells. However, its ability to lower BER is less pronounced.

As illustrated in Fig. 3, VN mitigates lateral charge migration by shifting the code lengths of a current word-line (e.g., WL(n)) while considering the adjacent code lengths of the previously written word-line (e.g., WL(n-1)). VN calculates the potential difference between corresponding states in WL(n) and WL(n-1), assigning

a higher penalty to larger differences. It then selects the storage combination that yields the smallest cumulative potential difference, thereby minimizing high-voltage disparities between adjacent word-lines. DVDS operates similarly to VN but instead of shifting, it selects the optimal storage combination by flipping or not flipping each code length. Modern flash controllers also incorporate scramblers [5, 7] to randomize the distribution of 0s and 1s in raw data, preventing exploitation by malicious applications. Consequently, data coding methods are primarily applied to randomized data at the controller level.

3 MOTIVATION

Data coding helps prevent data from being stored in error-prone states, thereby mitigating retention errors. By reducing the likelihood of errors, it also lowers the execution frequency of other error recovery mechanisms, such as ECC, data rewriting, refreshing and refilling, and reference voltage shifting. Consequently, this allows the use of low-cost and computationally efficient ECC techniques (e.g., BCH or LDPC), which can further enhance performance and effectively restore erroneous data. While previous data coding methods like WBVM and LRPER aim to reduce errors by minimizing the occurrence of low or high potential states, they fail to adequately address the issue of high potential differences between adjacent word-lines. This limitation makes them less effective in mitigating lateral charge migration, particularly in TLC NAND flash memory during its early usage phase with low P/E cycles. Similarly, although VN and DVDS consider lateral charge migration, they require additional read operations to retrieve previously written data from neighboring word-lines in order to compute the optimal storage combination. These extra read operations introduce additional overhead, which may impact performance and energy efficiency.

The above observations serve as the primary motivation for this paper. In this work, we propose a reliability-based coding strategy to suppress lateral charge migration for TLC NAND flash memory, particularly during its early usage stage when P/E cycles are low. The key distinction between the proposed strategy and existing data coding techniques (such as AC, WBVM, VN, DVDS, and LRPER) is its ability to strategically encode specific states that should not be adjacent to one another, ensuring they are stored at different encoding positions whenever possible. By distributing these unsuitable states apart, we significantly reduce the likelihood of high potential differences—such as those formed by specific state pairs—between neighboring word-lines. This strategy proves to be more effective than previous data coding techniques, which primarily rely on flipping, non-flipping, or shifting mechanisms. Encoding each word-line with this proposed strategy substantially decreases the probability of certain unsuitable state pairs occurring together, thereby further minimizing retention errors. Here, we will answer three important issues in the following:

- How to prevent the high potential difference (e.g., the pair of Er and G states) between any neighbor word-lines to reduce the impact of the lateral charge migration.
  To address this issue, we propose a coding concept that strategically controls the placement of appropriate states at suitable encoding positions, ensuring that high potential differences are minimized. Notably, this coding concept does not require additional read operations, making it efficient and well-suited for resource-limited environments.
- How to assign proper weights to different states at different encoding positions if an assignment mechanism is used to control the appropriate states at the suitable encoding positions.
  To address this issue, we explore the impact of different weights by analyzing the linear sequence, the Fibonacci sequence, and the exponential sequence. By systematically assigning proper weights to different states at suitable encoding positions, we aim to optimize data placement, minimize high potential differences between adjacent word-lines, and further suppress retention errors caused by lateral charge migration.
- How to further reduce the retention errors due to the lateral charge migration when compared to the previous methods.
  To address this issue, we introduce an enhancement skill designed to further suppress retention errors. This enhancement skill seamlessly integrates with the proposed coding concept, providing an additional layer of error reduction. If minimizing retention errors is prioritized over space efficiency, the enhancement skill can be applied to completely eliminate high potential states, significantly improving the reliability of NAND flash memory while maintaining data integrity.

4 A RELIABILITY-BASED CODING STRATEGY TO SUPPRESS LATERAL CHARGE MIGRATION

4.1 System Architecture

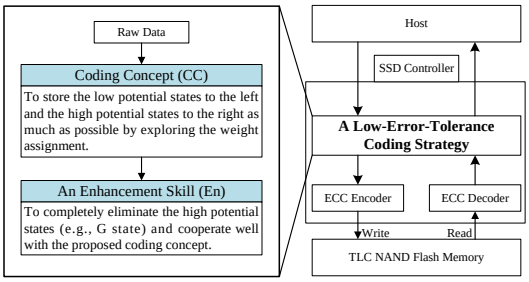


Figure 4: System architecture.

We propose a reliability-based coding strategy to suppress the lateral charge migration for TLC NAND flash memory, particularly during its early usage stage when P/E cycles are low. Before raw data from the host is written to NAND flash memory, the proposed strategy is applied to optimize data storage and enhance reliability. The overall system architecture is illustrated in Fig. 4. This strategy consists of two core components: a coding concept detailed in Section 4.2, which addresses the optimal placement of cell states to minimize high potential differences, and an enhancement skill discussed in Section 4.3, which further reduces retention errors. These two components collectively tackle the three key issues outlined in

Section 3, ensuring improved endurance and data integrity in TLC NAND flash memory.

To mitigate high potential differences—such as those between Er and G states—between neighboring word-lines and suppress lateral charge migration, the proposed coding concept prioritizes storing low-potential states toward the left and high-potential states toward the right whenever possible, as detailed in Section 4.2. To effectively assign proper weights to different states at various encoding positions and ensure optimal state placement, we explore the linear sequence, Fibonacci sequence, and exponential sequence in Section 4.2.2. Additionally, we analyze the relationship between the BER (bit error rate) effects of different paired states and the impact of weight assignments in Section 4.2.3. To further mitigate retention errors resulting from lateral charge migration, we introduce an enhancement skill in Section 4.3 that works in conjunction with the proposed coding concept. This enhancement skill aims to completely eliminate high-potential states (e.g., G state) and minimize high-potential differences (e.g., Er-G state pairs), prioritizing retention error reduction over storage efficiency when necessary. Given that coding methods are implemented at the SSD controller level, their design must remain simple and computationally efficient to ensure practical deployment.

4.2 Coding Concept

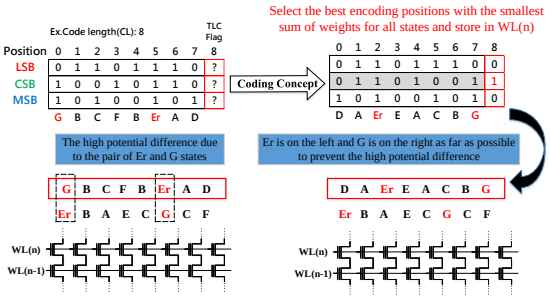


Figure 5: Coding concept.

4.2.1 *Encoding Process*

Since a word-line (WL) consists of numerous cells, we divide it into a fixed number of code lengths (CLs) for efficient encoding, where each CL contains a predetermined number of cells. The key idea of the proposed coding concept is to encode each code length within a word-line such that low-potential states (such as Er and A states) are positioned toward the left, while high-potential states (such as F and G states) are placed toward the right as much as possible. This strategy minimizes high potential differences (e.g., the Er-G state pair) between neighboring word-lines, effectively mitigating lateral charge migration.

To achieve this, we assign different weights to states depending on their encoding positions. States in suitable positions receive lower weights, while states in unsuitable positions receive higher weights. This concept differs from traditional weight tables in prior works [10, 11], as it considers both state types and their positions rather than focusing only on individual states. Unlike previous methods that do not consider encoding positions or only optimize a single state's placement, the proposed strategy accounts for both high potential differences and weight distribution.

The core design principle ensures that specific states (based on their voltage thresholds) are assigned minimal weight at ideal encoding positions, with weights gradually increasing as they move away. This results in states being positioned in opposite or dispersed directions within each code length, reducing the probability of forming undesirable state pairs (e.g., Er-G) between neighboring word-lines. Additionally, because this coding concept encodes only the current word-line without referencing previously written word-lines, it avoids extra read operations, significantly reducing the overhead compared to VN and DVDS, which require reading previously stored data to determine optimal encoding positions.

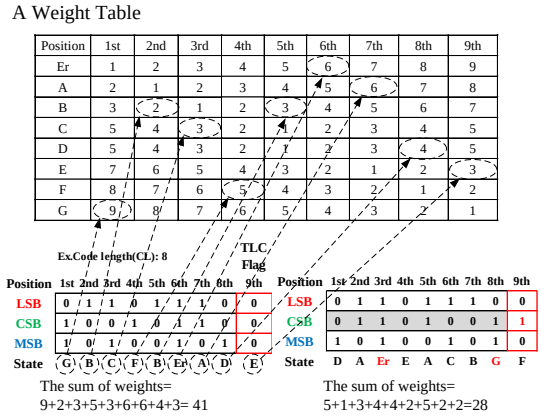


Figure 6: The proposed coding concept uses a weight table to calculate the sum of weights for a code length.

To ensure that low-potential states are positioned to the left of a code length while high-potential states are placed to the right as much as possible, we propose a simple yet effective bit-flip method. This method modifies cell states by selectively flipping or retaining corresponding bits to determine the most suitable combination. As illustrated in Fig. 5, each TLC NAND flash memory cell represents three bits, and a flag cell can control up to eight possible code length combinations through different flipping configurations. In this example, assume that the current word-line, WL(n), is about to be programmed, while its neighboring word-line, WL(n-1), has already been written. Consider a code length in WL(n) containing eight TLC cells with an initial state sequence of (G, B, C, F, B, Er, A, D) and a flag cell of (000). In this case, the high-potential G state appears on the far left, while the low-potential Er state is positioned toward the right, making this an undesirable arrangement. This is because the G and Er states in WL(n) can pair with Er and G states in WL(n-1), respectively, increasing the likelihood of retention errors caused by lateral charge migration.

By applying a bit-flip operation using a flag cell configuration of (010), the state sequence transforms into (D, A, Er, E, A, C, B, G). In this new arrangement, Er is now positioned to the left while G is placed to the right, significantly improving the encoding balance. Moreover, the previously written word-line, WL(n-1), already follows a similar pattern, where Er appears on the left and G is on

the right. This adjustment reduces the probability of Er-G pairings between adjacent word-lines, thereby suppressing lateral charge migration. The crucial challenge is determining the best flag cell configuration—such as why (010) is preferable over (000). To address this, the proposed coding concept employs a weight table that assigns different weights to states based on their encoding positions. The objective is to prioritize low-potential states in leftmost positions and high-potential states in rightmost positions within a code length.

As shown in Fig. 6, when the flag cell is set to E state (000), the original code length state sequence (G, B, C, F, B, Er, A, D) yields a total weight sum of 41 (calculated as 9+2+3+5+3+6+6+4+3). In contrast, when the flag cell is set to F state (010), the updated sequence (D, A, Er, E, A, C, B, G) produces a lower weight sum of 28 (i.e., 5+1+3+4+4+2+5+2+2). Since the configuration with the smaller weight sum (010) aligns more closely with the intended encoding strategy—positioning Er to the left and G to the right—it is deemed the better encoding choice. By ensuring that all code lengths within a word-line adhere to this coding concept, the probability of Er-G pairs appearing between adjacent word-lines can be minimized. This ultimately reduces retention errors caused by lateral charge migration. Furthermore, the proposed coding concept only requires a small weight table and does not need to reference previously written neighbor word-lines, thereby lowering computational complexity. In the following section, we will further explore the methodology for assigning appropriate weights to different states at varying encoding positions to optimize the encoding process.

4.2.2 Weight Assignment

The proposed coding concept aims to strategically position states within their optimal encoding locations by assigning weights that reflect their suitability. Specifically, a state receives a lower weight when placed in a favorable encoding position and a higher weight when positioned unfavorably. This weighting mechanism allows us to guide the placement of specific states, preventing undesirable pairings (such as Er and G states) between adjacent word-lines, which would otherwise contribute to retention errors due to lateral charge migration. For example, consider a weight assignment where the Er state has its lowest weight at the leftmost position of a code length, with its weight progressively increasing toward the right. Conversely, the G state has its lowest weight at the rightmost position, with its weight increasing toward the left. Using this structured weight table, we can systematically arrange Er and G states in opposite directions within a code length. This approach significantly reduces the probability of Er and G states appearing adjacent to each other across neighboring word-lines, thereby mitigating the impact of high potential differences that exacerbate charge migration.

By enforcing a well-structured weight assignment, we ensure that states naturally align with positions that minimize error-prone conditions. The effectiveness of this approach hinges on the careful selection of weight values, as they directly influence encoding outcomes. As a result, the process of assigning appropriate weights to different states at various encoding positions plays a critical role in ensuring that each state is placed optimally. Then, we explore the linear sequence, the Fibonacci sequence, and the exponential sequence to investigate their impact on assigning incremental weights to different states in different directions. Since these three sequences have distinct characteristics and can be precomputed, the weights can be recorded in a small weight table, making them suitable for resource-limited embedded systems.

For the linear, Fibonacci, and exponential sequences, Linear(n) represents a linear number of n, Fib(n) represents the n-th Fibonacci number, and Exp(n) represents an exponential value raised to the n-th power of 2 (i.e., 2^n). Using these sequences, we create three weight tables that guide encoding decisions. Assume that a code length contains eight cells along with a flag cell, the weights for the Er state using the linear sequence starting from Linear(n) follow the pattern: (Linear(n), Linear(n+1), Linear(n+2), ..., Linear(n+8)). Similarly, the weights for Er using the Fibonacci sequence follow the pattern: (Fib(n), Fib(n+1), Fib(n+2), ..., Fib(n+8)). For the exponential sequence, the pattern is: (Exp(n), Exp(n+1), Exp(n+2), ..., Exp(n+8)). Conversely, the G state follows a reversed weight pattern since G should have the smallest weight on the right side of a code length.

For example, the weight values for a code length using the linear, Fibonacci, and exponential sequences, starting from Linear(2), Fib(2), and Exp(2) for the Er state, are: (2, 3, 4, 5, 6, 7, 8, 9, 10) (linear), (1, 2, 3, 5, 8, 13, 21, 34, 55) (Fibonacci), (2, 4, 8, 16, 32, 64, 128, 256, 512) (exponential). For the G state, the corresponding reversed weight values are: (10, 9, 8, 7, 6, 5, 4, 3, 2) (linear), (55, 34, 21, 13, 8, 5, 3, 2, 1) (Fibonacci), (512, 256, 128, 64, 32, 16, 8, 4, 2) (exponential). Additionally, weight assignments must account for the flag cell as the ninth encoding position. This consideration is necessary because if two adjacent flag cells between neighboring word-lines store Er and G states, they could still lead to a high potential difference. Properly assigning weights ensures that undesirable state pairings, such as Er and G, are minimized, thereby reducing the impact of lateral charge migration.

In addition to the Er and G states, the remaining six states (A, B, C, D, E, and F) do not create excessive potential differences between neighboring word-lines. However, due to their varying voltage thresholds, these states should still have carefully assigned weights to ensure optimal encoding positions within a code length. Each of these six states should have its smallest weight at a specific encoding position, with weights increasing in both directions according to the linear sequence, Fibonacci sequence, or exponential sequence. This approach is necessary because different voltage thresholds require different preferred encoding positions. Specifically, states with higher voltage thresholds should be positioned further to the right within a code length.

As shown in Table 1, the A state is assigned the smallest weight (weight = 1) at the second encoding position of a code length. From this position, its weight increases in both directions. The corresponding weight distributions for the linear sequence, Fibonacci sequence, and exponential sequence are: (2, 1, 2, 3, 4, 5, 6, 7, 8) (linear), (2, 1, 2, 3, 5, 8, 13, 21, 34) (Fibonacci), (2, 1, 2, 4, 8, 16, 32, 64, 128) (exponential). Similarly, as the voltage thresholds of B, C, D, E, and F states progressively increase, they are assigned their smallest weights at the 3rd, 5th, 5th, 7th, and 8th encoding positions, respectively. The full weight distributions for each state, based on the three sequences, are detailed in Table 1. It is important to emphasize that the weight assignment presented in this paper is a general concept. The optimal weight values can be dynamically adjusted based on the actual bit error rate (BER) effects observed

Table 1: Three weight tables according to the linear sequence, the Fibonacci sequence and the exponential sequence.

A weight table according to the linear sequence

Position	1st	2nd	3rd	4th	5th	6th	7th	8th	9th (a flag cell)
Er	Linear(n)	Linear(n+1)	Linear(n+2)	Linear(n+3)	Linear(n+4)	Linear(n+5)	Linear(n+6)	Linear(n+7)	Linear(n+8)
A	2	1	2	3	4	5	6	7	8
B	3	2	1	2	3	4	5	6	7
C	5	4	3	2	1	2	3	4	5
D	5	4	3	2	1	2	3	4	5
E	7	6	5	4	3	2	1	2	3
F	8	7	6	5	4	3	2	1	2
G	Linear(n+8)	Linear(n+7)	Linear(n+6)	Linear(n+5)	Linear(n+4)	Linear(n+3)	Linear(n+2)	Linear(n+1)	Linear(n)

A weight table according to the Fibonacci sequence

Position	1st	2nd	3rd	4th	5th	6th	7th	8th	9th (a flag cell)
Er	Fib(n)	Fib(n+1)	Fib(n+2)	Fib(n+3)	Fib(n+4)	Fib(n+5)	Fib(n+6)	Fib(n+7)	Fib(n+8)
A	2	1	2	3	5	8	13	21	34
B	3	2	1	2	3	5	8	13	21
C	8	5	3	2	1	2	3	5	8
D	8	5	3	2	1	2	3	5	8
E	21	13	8	5	3	2	1	2	3
F	34	21	13	8	5	3	2	1	2
G	Fib(n+8)	Fib(n+7)	Fib(n+6)	Fib(n+5)	Fib(n+4)	Fib(n+3)	Fib(n+2)	Fib(n+1)	Fib(n)

A weight table according to the exponential sequence

Position	1st	2nd	3rd	4th	5th	6th	7th	8th	9th (a flag cell)
Er	Exp(n)	Exp(n+1)	Exp(n+2)	Exp(n+3)	Exp(n+4)	Exp(n+5)	Exp(n+6)	Exp(n+7)	Exp(n+8)
A	2	1	2	4	8	16	32	64	128
B	4	2	1	2	4	8	16	32	64
C	16	8	4	2	1	2	4	8	16
D	16	8	4	2	1	2	4	8	16
E	64	32	16	8	4	2	1	2	4
F	128	64	32	16	8	4	2	1	2
G	Exp(n+8)	Exp(n+7)	Exp(n+6)	Exp(n+5)	Exp(n+4)	Exp(n+3)	Exp(n+2)	Exp(n+1)	Exp(n)

for different state pairings. In the following section, we will investigate the relationship between BER effects and weight assignment, refining the proposed strategy to enhance NAND flash memory reliability further.

4.2.3 Relationship between BER Effects and Weight Assignment

Table 2: BER (Bit Error Rate) effects.

BER Effects	BER of a 5-state-gap pair	BER of a 6-state-gap pair	BER of a 7-state-gap pair
DVDS	21.74%	28.26%	50%
LRPER	18.05%	26.89%	55.06%
VN	21.04%	26.62%	52.34%
Synthetic BER-90%	5%	5%	90%
Synthetic BER-80%	10%	10%	80%
Synthetic BER-70%	15%	15%	70%
Synthetic BER-60%	20%	20%	60%
Synthetic BER-50%	25%	25%	50%
Synthetic BER-40%	30%	30%	40%

We will examine the relationship between bit error rate (BER) effects of different paired states and their weight assignments, specifically using the linear sequence, Fibonacci sequence, and exponential sequence. Based on prior research [22, 23, 27], the three most error-prone state pairings are those with a 7-state-gap, 6-state-gap, and 5-state-gap, where an n-state-gap refers to the voltage threshold difference of n between two states in a pair. Among these, the (Er, G) pair is the only 7-state-gap pair, making it the most vulnerable to errors. The 6-state-gap pairs include (Er, F) and (A, G), while the 5-state-gap pairs include (Er, E), (A, F), and (B, G). Table 2 presents the BER effects of these three state-gap pairs as measured in DVDS [27], LRPER [22], and VN [23]. It is evident that the 7-state-gap pair contributes the highest BER, followed by the 6-state-gap pairs and then the 5-state-gap pairs. For example, measurements from LRPER [22] indicate that a 7-state-gap pair accounts for 55.06% of total errors, while 6-state-gap and 5-state-gap pairs contribute 26.89% and 18.05%, respectively.

To ensure comprehensive evaluation, we also consider synthetic BER effects that simulate different distribution scenarios. For example, a synthetic BER-90% model assumes that 90% of total errors originate from 7-state-gap pairs, while the remaining 10% are evenly distributed between 6-state-gap and 5-state-gap pairs. Other synthetic models, such as BER-80%, BER-70%, BER-60%, BER-50%, and BER-40%, follow a similar distribution pattern. It is important to note that even with the use of scramblers/randomizers to evenly distribute paired states, the total occurrences of 7-state-gap, 6-state-gap, and 5-state-gap pairs remain significant. These pairs may still account for approximately 3.12%, 6.25%, and 9.37% of all paired states, respectively. As a result, they continue to pose a risk of

high potential differences that can exacerbate retention errors. In systems with low tolerance for retention errors, it is essential to minimize the occurrence of 7-state-gap, 6-state-gap, and 5-state-gap pairs as much as possible to improve reliability.

Table 3: Comparison of weight tables according to the linear sequence (i.e., $Table_{Linear(n)}$), the Fibonacci sequence (i.e., $Table_{Fib(n)}$) and the exponential sequence (i.e., $Table_{Exp(n)}$). Note that * represents that this weight table is the best of all.

BER Effects	Well-Performed Weight Tables
DVDS	$Table_{Fib(5)}$*, $Table_{Fib(4)}$, $Table_{Fib(6)}$
LRPER	$Table_{Fib(5)}$*, $Table_{Fib(4)}$, $Table_{Fib(6)}$
VN	$Table_{Fib(5)}$*, $Table_{Fib(4)}$, $Table_{Fib(6)}$
Synthetic BER-90%	$Table_{Fib(6)}$*, $Table_{Fib(7)}$, $Table_{Exp(5)}$
Synthetic BER-80%	$Table_{Fib(6)}$*, $Table_{Fib(7)}$, $Table_{Exp(4)}$, $Table_{Exp(5)}$
Synthetic BER-70%	$Table_{Fib(5)}$*, $Table_{Fib(6)}$
Synthetic BER-60%	$Table_{Fib(5)}$*, $Table_{Fib(4)}$, $Table_{Fib(6)}$
Synthetic BER-50%	$Table_{Fib(5)}$*, $Table_{Fib(4)}$, $Table_{Fib(6)}$
Synthetic BER-40%	$Table_{Linear(9)}$*, $Table_{Fib(4)}$, $Table_{Fib(5)}$, $Table_{Linear(10)}$, $Table_{Linear(11)}$

In Table 3, we evaluate the impact of different weight tables based on the linear sequence, Fibonacci sequence, and exponential sequence under various BER (bit error rate) effects. The experiments encode approximately 190MB of random raw data and compare the performance of different weight assignments across DVDS, LRPER, VN, and several synthetic BER models (i.e., Synthetic BER-90%, BER-80%, BER-70%, BER-60%, BER-50%, and BER-40%). For example, $Table_{Fib(n)}$ represents a weight table derived from the Fibonacci sequence, where the smallest weight for Er and G begins at Fib(n), while the other six states (A, B, C, D, E, and F) have their smallest weights assigned at their respective encoding positions, as shown in Table 1. Similarly, $Table_{Linear(n)}$ and $Table_{Exp(n)}$ follow the same principle but start from Linear(n) and Exp(n), respectively. Once a weight table is created (e.g., $Table_{Fib(5)}$), the proposed coding concept utilizes it to encode each code length within a word-line. The goal is to assign appropriate states to their optimal encoding positions, ensuring the smallest sum of weights for all states in each code length.

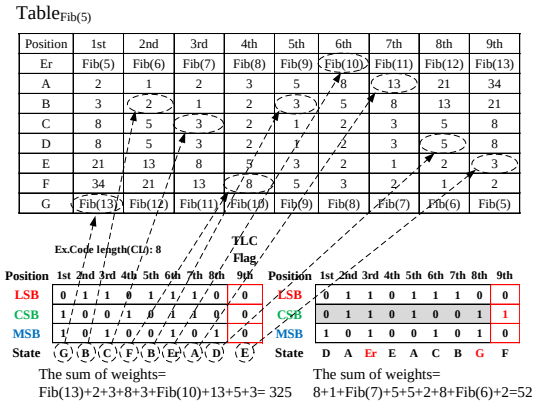


Figure 7: The proposed coding concept uses $Table_{Fib(5)}$) to calculate the sum of weights for a code length.

For example, as illustrated in Fig. 7, assume that a code length consists of eight cells along with a flag cell. When the flag cell is in state E (000), the initial state sequence of the code length is (G, B, C, F, B, Er, A, D). The proposed coding concept calculates its total weight sum as 325 (i.e., Fib(13)+2+3+8+3+Fib(10)+13+5+3=325) using $Table_{Fib(5)}$. However, if the flag cell is flipped to state F (010), the state sequence transforms into (D, A, Er, E, A, C, B, G), and the new total weight sum is significantly reduced to 52 (i.e., 8+1+Fib(7)+5+5+2+8+Fib(6)+2) from $Table_{Fib(5)}$. This comparison highlights that encoding with flag cell (010) is significantly more effective than flag cell (000) because Er is positioned to the left, G is placed to the right, and the other six states are assigned to their optimal encoding positions. By applying the same evaluation process to the remaining flag cells (001, 011, 100, 101, 110, 111), we can determine the best encoding configuration—the one that results in the smallest sum of weights among all eight possible combinations. This process ensures that low-potential states are placed to the left and high-potential states to the right as much as possible. Finally, we measure the total BER by accumulating the BER effects of all 7-state-gap pairs, 6-state-gap pairs, and 5-state-gap pairs after encoding all code lengths in a word-line using the procedure described above.

After evaluating the total bit error rates (BERs) using different weight tables (i.e., $Table_{Linear(n)}$, $Table_{Fib(n)}$, and $Table_{Exp(n)}$), we can identify well-performing weight tables that effectively support the proposed coding concept in reducing the overall BER for random raw data.

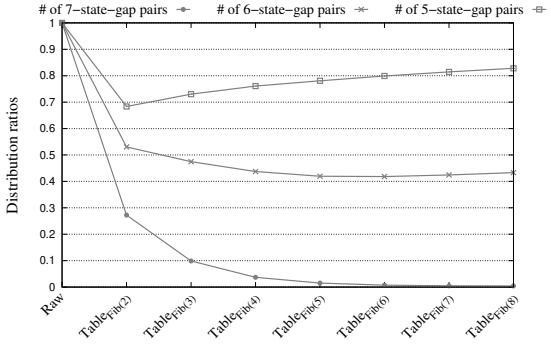


Figure 8: The distribution ratios of the total number of the 7-state-gap pairs, the 6-state-gap pairs and the 5-state-gap pairs under $Table_{Fib(2)}$~$Table_{Fib(8)}$.

According to the results in Table 3, we observe that $Table_{Linear(n)}$ and $Table_{Fib(n)}$ exhibit similar performance when the BER effects of 7-state-gap, 6-state-gap, and 5-state-gap pairs are relatively close (e.g., Synthetic BER-40%). However, when the BER impact of a 7-state-gap pair is more dominant, both $Table_{Fib(n)}$ and $Table_{Exp(n)}$ outperform $Table_{Linear(n)}$. Interestingly, $Table_{Fib(5)}$ consistently achieves the lowest total BER across various BER scenarios. This effectiveness stems from its ability to significantly reduce the total number of 7-state-gap pairs while also moderately decreasing 6-state-gap pairs when applied to random raw data. As shown in Fig. 8, applying $Table_{Fib(2)}$ through $Table_{Fib(8)}$ effectively minimizes error-prone state-gap pairs. In particular, when the BER effect of a 7-state-gap pair is extremely high (e.g., Synthetic BER-80% or Synthetic BER-90%), $Table_{Fib(6)}$ may provide better results than $Table_{Fib(5)}$. This is because prioritizing a greater reduction in 7-state-gap pairs yields more significant BER improvements than the marginal increases in 5-state-gap and 6-state-gap pairs.

Overall, $Table_{Fib(n)}$ proves to be more effective in assisting the proposed coding concept to achieve a lower BER compared to $Table_{Exp(n)}$. This is because the Fibonacci sequence exhibits a growth pattern that is similar to the exponential sequence, but less abrupt, making it a better-balanced choice for controlling state placement and minimizing high-potential differences between neighboring word-lines.

4.3 An Enhancement Skill

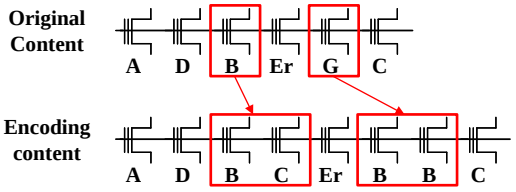
Table 4: An Encoding Table: G state is encoded to BB states and B state is encoded to BC states.

State	Encoding state
Er	Er
A	A
B	B, C
C	C
D	D
E	E
F	F
G	B, B

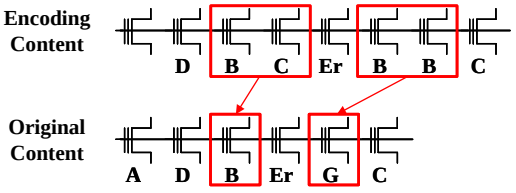
Although the proposed coding concept effectively positions Er and G states in opposite directions and minimizes the high potential difference (e.g., the Er-G state pair) between neighboring word-lines, it cannot completely eliminate instances where Er and G states appear together. Additionally, high potential states (such as the G state) are inherently more susceptible to voltage shifts than low potential states (such as the Er state), making them more prone to retention errors over extended data retention periods. To further mitigate retention errors, we introduce an enhancement skill that completely eliminates the G state. The core idea behind this enhancement skill is to encode the G state into a combination of two low potential states (selected from A, B, or C states) since low potential states are inherently more stable and less susceptible to retention errors.

As shown in Table 4, the enhancement skill encodes a G state into two consecutive B states (BB) and modifies a B state into BC to prevent confusion between BB and BC states. For example, as illustrated in Fig. 9(a), the original state sequence (A, D, B, Er, G, C) is encoded into (A, D, B, C, Er, B, B, C) using the enhancement skill. Similarly, as shown in Fig. 9(b), the encoded sequence (A, D, B, C, Er, B, B, C) can be successfully decoded back to (A, D, B, Er, G, C) without ambiguity, as BB and BC states are uniquely distinguishable. Additionally, it helps mitigate vertical charge de-trap effects, which become more pronounced at high P/E cycles in TLC NAND flash memory [27]. Notably, the enhancement skill can function independently but achieves the lowest BER (bit error rate) when combined with the proposed coding concept, as confirmed by experimental results.

Although the proposed coding concept with the enhancement skill achieves the lowest BER, it introduces extra space overhead due to longer encoding (e.g., G state → BB states, B state → BC states). When using a scrambler/randomizer [5, 7] to randomize state distributions in raw data, the additional space overhead is approximately 25%. To minimize this space overhead, we propose two optimization conditions: Selective Encoding and Optimized State Selection. Selective Encoding: If a word-line does not contain a G state, the enhancement skill is not applied, thereby avoiding unnecessary space expansion. Optimized State Selection: When encoding the G state, we select the least frequent low-potential state in the word-line as its replacement. For example, if the B state appears the least frequently, the encoding is optimized as G → BB, B → BC, reducing the overall extra space cost. Even with these optimizations, the enhancement skill inevitably increases space usage due to longer encoding. However, if minimizing retention errors takes priority over space efficiency, the enhancement skill remains a valuable approach. This is particularly critical in modern NAND flash memory, where higher cell density leads to increased error rates. Given that future NAND flash memory designs will continue pushing for higher storage capacity per cell, systems with low error tolerance and long-term data storage needs will greatly benefit from prioritizing retention error reduction over space efficiency.



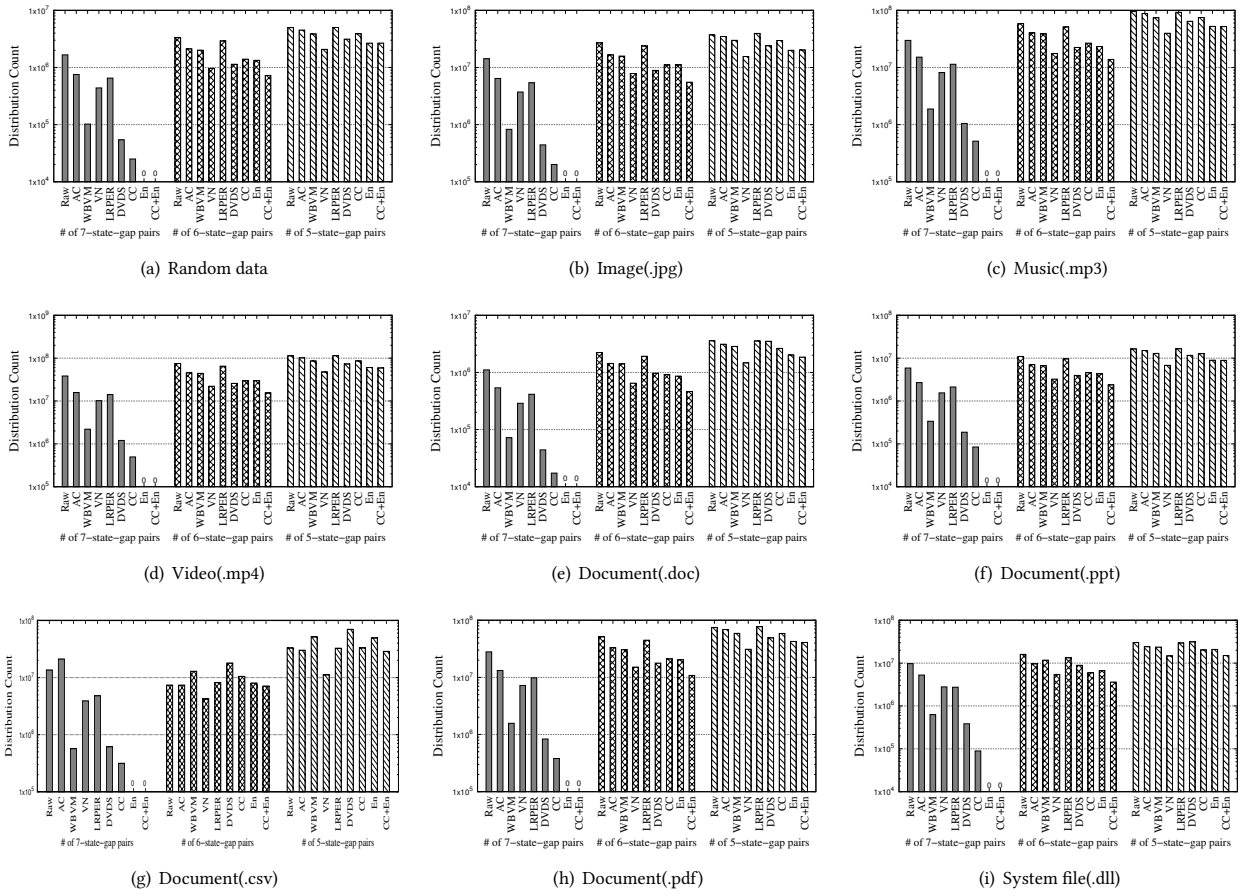
(a) Encoding process: G state is encoded to "BC" states and B state is encoded to "BB" states.



(b) Decoding process: "BC" states are decoded to G state and "BB" states are decoded to B state.

Figure 9: Encoding process and decoding process of the enhancement skill.

In addition to the extra space required for storing longer encoded data, another overhead arises from the need to access this additional space during read and write operations—a challenge that previous encoding methods have also encountered. This overhead exists because a single word-line of original data expands in length after encoding. To mitigate this issue, we can leverage the multi-channel architecture of modern SSDs. Today's SSDs employ multiple I/O channels for NAND flash memory to parallelize operations and enhance access speed. By splitting the longer encoded data into smaller variable-length segments and distributing them across different SSD channels, we can offset the access overhead. The address mapping for these fragmented data segments can be efficiently managed using existing flash translation layer (FTL) techniques such as DFTL and ZFTL [15, 26].



(a) Random data (b) Image(.jpg) (c) Music(.mp3)

(d) Video(.mp4) (e) Document(.doc) (f) Document(.ppt)

(g) Document(.csv) (h) Document(.pdf) (i) System file(.dll)

Figure 10: Distribution count of 7-state-gap pairs, 6-state-gap pairs and 5-state-gap pairs.

Furthermore, the proposed coding concept introduces low computational complexity and reduces read operations compared to previous methods (e.g., VN and DVDS). This efficiency gain compensates for the extra access overhead introduced by the enhancement skill, making it a viable trade-off. Ultimately, the enhancement skill remains an optional feature—it can be seamlessly integrated with the proposed coding concept when minimizing retention errors takes priority over space efficiency. However, even when only the proposed coding concept is applied (without the enhancement skill), it still achieves a lower BER than existing methods, assuming equal space overhead from flag cells.

Table 5: Workloads characteristics.

File type	Number of Files	Total File Size
Random data	1	19MB
Image (.jpg)	7,197	146MB
Music (.mp3)	1,000	394MB
Video (.mp4)	450	446MB
Document (.doc)	303	22.4MB
Document (.ppt)	45	68.4MB
Document (.csv)	68	371MB
Document (.pdf)	119	300MB
System Files (.dll)	407	172MB

5 PERFORMANCE EVALUATION

5.1 Experimental Data

We evaluate retention BER (bit error rate) using two types of workloads: randomized data and daily life files, as summarized in Table 5. The daily life files encompass a diverse range of file types, including images, music, videos, documents, and system files. For image files, we select bird images [2] in the JPG format. For music files, we use speech accent archive files [29] in the MP3 format. For video files, we choose real-life situation recordings [12] in the MP4 format. For document files, we include a variety of document types, such as administrative forms (DOC), course PowerPoints (PPT), U.S. Education Datasets: Unification Project (CSV) [13], and science memos (PDF) [3]. For system files, we analyze dynamic-link libraries (DLL) from Windows OS. In our experiments, we compare the proposed strategy—including CC (coding concept), En (enhancement skill), and CC+En (both combined)—against raw data and several state-of-the-art methods such as AC [28], WBVM [11], VN [23], LRPER [22],

and DVDS [27]. The raw data category includes both randomized data and daily life files to provide a comprehensive evaluation.

5.2 Distribution Count of 7-state-gap Pairs, 6-state-gap Pairs and 5-state-gap Pairs

In Fig. 10, we analyze the worst top-three distribution counts of 7-state-gap pairs (i.e., (Er, G) pairs), 6-state-gap pairs (i.e., (Er, F) and (A, G) pairs), and 5-state-gap pairs (i.e., (Er, E), (A, F), and (B, G) pairs) across all workloads. These high-state-gap pairs are particularly prone to retention errors due to the high potential difference, especially in TLC NAND flash memory with low P/E cycles. Our results indicate that CC (coding concept) significantly reduces the 7-state-gap pairs across all workloads when compared to raw data, achieving reductions of: 98.5% for random data (Fig. 10(a)), 98.6% for JPG images (Fig. 10(b)), 98.3% for MP3 files (Fig. 10(c)), 98.7% for MP4 videos (Fig. 10(d)), 98.5% for DOC documents (Fig. 10(e)), 98.6% for PPT presentations (Fig. 10(f)), 97.7% for CSV datasets (Fig. 10(g)), 98.7% for PDF files (Fig. 10(h)), 99.1% for DLL system files (Fig. 10(i)). Moreover, CC consistently outperforms AC, WBVM, VN, LRPER, and DVDS, exhibiting a lower number of 7-state-gap pairs across all workloads. Additionally, En (enhancement skill) and CC+En (combined approach) completely eliminate 7-state-gap pairs, while CC+En further reduces both 6-state-gap pairs and 5-state-gap pairs more effectively than other methods. For systems that store a large volume of important data over extended periods, the proposed strategy significantly reduces retention errors, thereby minimizing the need for additional error recovery mechanisms (e.g., ECC, data rewriting, refreshing and refilling, and reference voltage shifting) and enhancing overall system performance.

5.3 Retention BER

In Fig. 11, we measured the retention BER (bit error rate) across all workloads based on the BER effects of 7-state-gap pairs, 6-state-gap pairs, and 5-state-gap pairs, as used in DVDS, LRPER, and VN. The raw data serves as the baseline for comparison. Our results show that CC (coding concept) significantly reduces retention BER across all workloads by: 70.0% for random data (Fig. 11(a)), 71.1% for JPG images (Fig. 11(b)), 68.5% for MP3 files (Fig. 11(c)), 71.6% for MP4 videos (Fig. 11(d)), 70.7% for DOC documents (Fig. 11(e)), 71.2% for PPT presentations (Fig. 11(f)), 66.3% for CSV datasets (Fig. 11(g)), 71.9% for PDF files (Fig. 11(h)), 75.4% for DLL system files (Fig. 11(i)). Moreover, when CC and En (enhancement skill) are combined (CC+En), the retention BER reduction is even more significant, achieving reductions of: 85.3% for random data, 86.2% for JPG images, 84.5% for MP3 files, 86.1% for MP4 videos, 85.9% for DOC documents, 85.9% for PPT presentations, 76.8% for CSV datasets, 86.2% for PDF files, 86.3% for DLL system files.

While En performs slightly better than CC in most cases (except for DLL files), CC has lower space overhead because En requires a longer encoding. Therefore, if only one method can be used, we recommend CC, as it provides a similar reduction in retention BER while being more space-efficient. However, if space cost is not a primary concern, using CC and En together (i.e., CC+En) provides the lowest retention BER across all workloads. This remarkable reduction is due to the fact that CC+En completely eliminates 7-state-gap pairs and significantly reduces 6-state-gap and 5-state-gap pairs more effectively than any other method. For systems where reducing retention errors is more critical than minimizing space cost, CC+En is the optimal choice. It not only reduces retention errors but also lowers the overhead of other error recovery mechanisms such as ECC, data rewriting, refreshing and refilling, and reference voltage shifting, ultimately enhancing overall performance.

5.4 BER Reduction Ratio and Space Overhead

Table 6: Comparison of raw data, AC, WBVM, VN, LRPER, DVDS, CC, En, CC+En for randomized data.

	Avg. BER Reduction Ratio	Space Overhead
Raw Data	Baseline	N/A
AC	35.7%	12.5%
WBVM	58.7%	12.5%
VN	62.8%	12.5%
LRPER	38.1%	≈3.8%
DVDS	60.4%	12.5%
CC	70%	12.5%
En	74.2%	≈25%
CC+En	85.3%	≈35.5%

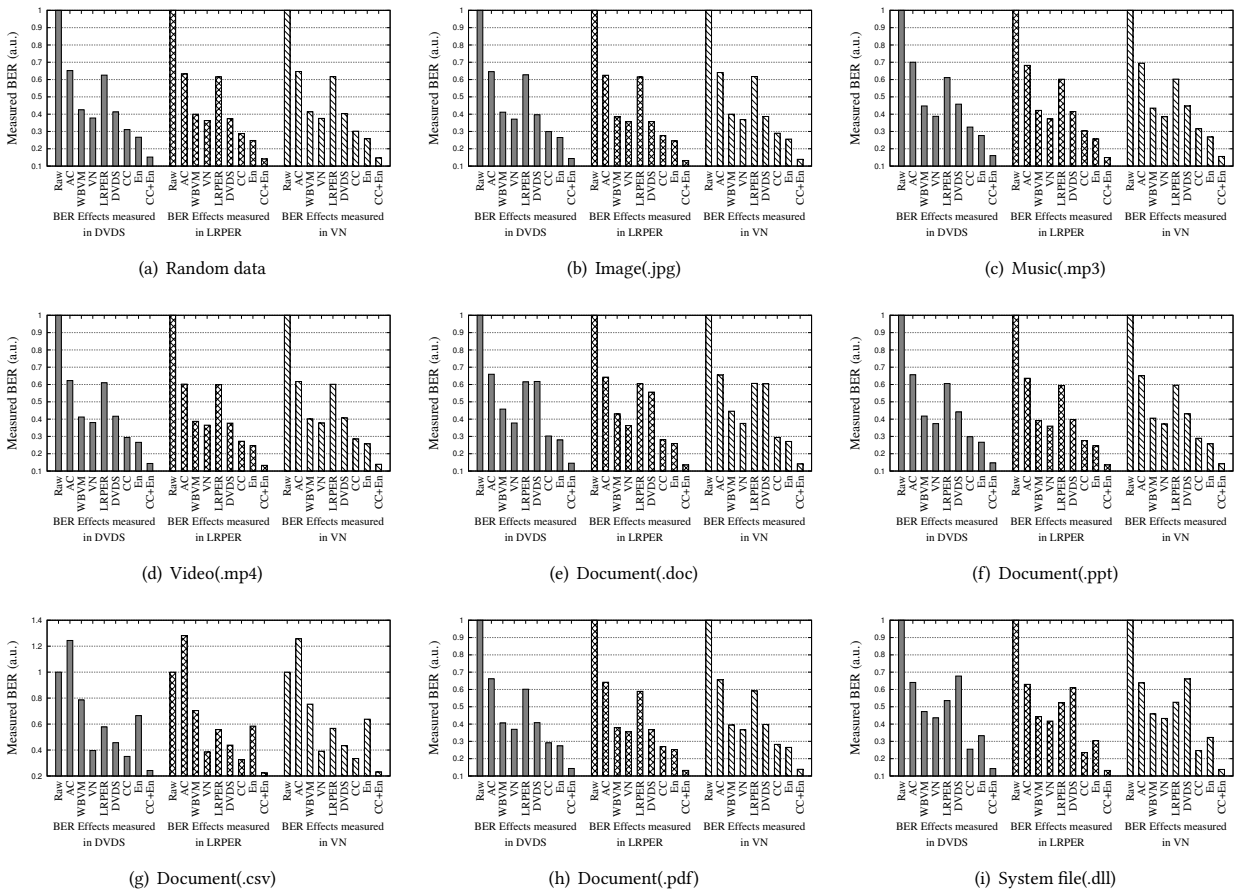
In Table 6, we summarize the BER reduction ratio and space overhead for various methods, including raw data, AC, WBVM, VN, LRPER, DVDS, CC, En, and CC+En, using randomized data as the workload. The raw data serves as the baseline (i.e., no encoding is applied).

CC achieves a lower retention BER than LRPER, and it further outperforms AC, WBVM, VN, and DVDS, even when all methods have the same space overhead (caused by flag cells). CC offers an optimal balance between space cost and BER reduction, achieving a lower bit error rate than other methods while maintaining low computation complexity. Compared to VN and DVDS, CC requires fewer read operations during encoding, which helps offset the extra space access overhead introduced by En (enhancement skill). If minimizing retention errors is the top priority, CC+En is the best choice, as it provides the lowest BER. If space overhead is a concern, CC alone is still highly effective, achieving a lower retention BER than other methods while maintaining a reasonable space cost. By integrating CC+En, systems that store large volumes of important data over extended periods can significantly reduce retention errors while optimizing the efficiency of error recovery mechanisms such as ECC, data rewriting, refreshing and refilling, and reference voltage shifting.

6 CONCLUSION

In this paper, we propose a reliability-based coding strategy, which consists of a coding concept (CC) and an enhancement skill (En), to effectively mitigate retention errors caused by lateral charge migration when NAND flash memory is in its early use stage and has low P/E cycles. We mainly answer three important issues to suppress the lateral charge migration in the following:

- To mitigate lateral charge migration, we aim to prevent high potential differences (e.g., between low and high potential states) between adjacent word-lines. The proposed coding concept achieves this by encoding each code length in a



(a) Random data (b) Image(.jpg) (c) Music(.mp3)

(d) Video(.mp4) (e) Document(.doc) (f) Document(.ppt)

(g) Document(.csv) (h) Document(.pdf) (i) System file(.dll)

Figure 11: The retention BER (bit error rate) of all workloads according to the BER effects of a 7-state-gap pair, a 6-state-gap pair and a 5-state-gap pair.

word-line such that low potential states (e.g., Er and A) are positioned toward the left, while high potential states (e.g., F and G) are positioned toward the right as much as possible. This strategic placement minimizes the occurrence of high potential differences, thereby reducing retention errors. Additionally, unlike previous methods such as VN and DVDS, the proposed coding concept only considers the current word-line during encoding. It does not require access to the previously written neighbor word-line, eliminating the need for additional read operations. This significantly reduces read overhead, making the proposed strategy more efficient compared to prior techniques.

- To control state placement at optimal encoding positions, we assign appropriate weights to different states at various positions. We explored three weight assignments—the linear sequence, Fibonacci sequence, and exponential sequence—as they each exhibit distinct characteristics and allow weights to be precomputed and stored efficiently in a small weight table. Furthermore, we analyzed the relationship between BER (bit error rate) effects of different paired states and the three sequences by systematically comparing weight tables derived from them. This analysis enabled us to identify well-performing weight tables that effectively assist the proposed coding concept in achieving a lower BER and improving reliability of NAND flash memory.
- We can achieve a further reduction in retention errors caused by lateral charge migration, surpassing previous methods, by incorporating an enhancement skill (En). This skill completely eliminates high-potential states (specifically, the G state) when minimizing retention errors is a higher priority than space efficiency. Moreover, we demonstrate that En integrates seamlessly with the proposed coding concept, achieving the lowest BER (bit error rate) among all methods. While En introduces additional space overhead due to its longer encoding scheme, the low computational complexity of the proposed coding concept helps offset this cost by reducing the need for additional read operations, thereby improving overall efficiency.

According to the experimental results, when TLC NAND flash memory is in its early usage stage with low P/E cycles, the proposed coding concept (CC) reduces the bit error rate (BER) by approximately 70%, while CC combined with the enhancement

skill (CC+En) further lowers the retention BER by about 85.3% for randomized data. CC outperforms LRPER and achieves even lower retention BER compared to AC, WBVM, VN, and DVDS, given the same space overhead. This demonstrates that CC effectively balances space efficiency and error reduction, making it a superior choice for minimizing retention errors. Additionally, CC can reduce read operations during encoding more efficiently than VN and DVDS, helping to offset the extra space access required by En. For systems that prioritize maximum retention error reduction over space efficiency, CC+En offers the best performance, achieving the lowest retention BER. However, if space overhead is a critical concern, using CC alone still provides significant error reduction, outperforming all other methods while maintaining a reasonable space cost.

For the future work, we will extend CC and En to QLC NAND flash memory, as QLC technology offers greater storage capacity and is the mainstream standard. However, due to its higher density, QLC NAND flash memory is more prone to charge leakage, making retention error mitigation an even more pressing challenge.

REFERENCES

[1] H. Aihara, K. Maeda, S. Suzuki, and K. Takeuchi. Extremely biased error correction method to reduce read disturb errors of 3d-tlc nand flash memories by 60%. In *2020 IEEE International Memory Workshop (IMW)*, pages 1–4, 2020.

[2] Ailurophile. Bird species. https://www.kaggle.com/discussions/general/126169, Mar 2020.

[3] T. Bozsolik. Data science cheat sheets, Feb 2020.

[4] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. *Proceedings of the IEEE*, 105(9):1666–1704, 2017.

[5] Y. Cai, S. Ghose, Y. Luo, K. Mai, O. Mutlu, and E. F. Haratsch. Vulnerabilities in mlc nand flash memory programming: Experimental analysis, exploits, and mitigation techniques. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 49–60, 2017.

[6] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu. Read disturb errors in mlc nand flash memory: Characterization, mitigation, and recovery. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 438–449, 2015.

[7] J. Cha and S. Kang. Data randomization scheme for endurance enhancement and interference mitigation of multilevel flash memory devices. *ETRI Journal*, 35(1):166–169, 2013.

[8] K.-C. Chiang, Y.-C. Li, W.-C. Wang, W.-K. Shih, and C.-C. Ho. Enhancing data integrity with efficient retention-refilling programming schemes. *SIGAPP Appl. Comput. Rev.*, 24(3):37–50, Oct. 2024.

[9] H. Choi, W. Liu, and W. Sung. Vlsi implementation of bch error correction for multilevel cell nand flash memory. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(5):843–847, 2010.

[10] Y. Deguchi, T. Nakamura, A. Hayakawa, and K. Takeuchi. 3-d nand flash value-aware ssd: Error-tolerant ssd without eccs for image recognition. *IEEE Journal of Solid-State Circuits*, 54(6):1800–1811, 2019.

[11] Y. Deguchi and K. Takeuchi. Word-line batch vth modulation of tlc nand flash memories for both write-hot and cold data. In *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 161–164, 2017.

[12] M. Elesawy. Real life violence situations dataset. https://www.kaggle.com/mohamedmustafa/real-life-violence-situations-dataset, Apr 2019.

[13] R. Garrard. U.s. education datasets: Unification project. https://www.kaggle.com/noriuk/us-education-datasets-unification-project, Apr 2020.

[14] A. Grossi, L. Zuolo, F. Restuccia, C. Zambelli, and P. Olivo. Quality-of-service implications of enhanced program algorithms for charge-trapping nand in future solid-state drives. *IEEE Transactions on Device and Materials Reliability*, 15:363–369, 06 2015.

[15] A. Gupta, Y. Kim, and B. Urgaonkar. Dftl: A flash translation layer employing demand-based selective caching of page-level address mappings. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, pages 229–240, New York, NY, USA, 2009. ACM.

[16] Y. Kong, M. Zhang, X. Zhan, R. Cao, and J. Chen. Retention correlated read disturb errors in 3-d charge trap nand flash memory: Observations, analysis, and solutions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4042–4051, 2020.

[17] T.-W. Kuo, P.-C. Huang, Y.-H. Chang, C.-L. Ko, and C.-W. Hsueh. An efficient fault detection algorithm for nand flash memory. *SIGAPP Appl. Comput. Rev.*, 11(2):8–16, Mar. 2011.

[18] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu. Self-adaptive system for addressing permanent errors in on-chip interconnects. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(4):527–540, 2010.

[19] Q. Li, L. Shi, C. J. Xue, K. Wu, C. Ji, Q. Zhuge, and E. H.-M. Sha. Access characteristic guided read and write cost regulation for performance improvement on flash memory. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*, pages 125–132, Santa Clara, CA, Feb. 2016. USENIX Association.

[20] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu. Improving 3d nand flash memory lifetime by tolerating early retention loss and process variation. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(3), dec 2018.

[21] A. Maconi, A. Arreghini, C. M. Compagnoni, G. Van den bosch, A. S. Spinelli, J. Van Houdt, and A. L. Lacaita. Impact of lateral charge migration on the retention performance of planar and 3d sonos devices. In *2011 Proceedings of the European Solid-State Device Research Conference (ESSDERC)*, pages 195–198, 2011.

[22] K. Maeda, K. Mizoguchi, and K. Takeuchi. Less reliable page error reduction for 3d-tlc nand flash memories with data overhead reduction by 40 In *2019 Silicon Nanoelectronics Workshop (SNW)*, pages 1–2, 2019.

[23] K. Mizoguchi, S. Kotaki, Y. Deguchi, and K. Takeuchi. Lateral charge migration suppression of 3d-nand flash by vth nearing for near data computing. In *2017 IEEE International Electron Devices Meeting (IEDM)*, pages 19.2.1–19.2.4, 2017.

[24] K. Mizoguchi, T. Takahashi, S. Aritome, and K. Takeuchi. Data-retention characteristics comparison of 2d and 3d tlc nand flash memories. In *2017 IEEE International Memory Workshop (IMW)*, pages 1–4, 2017.

[25] D. Park, Y. J. Nam, B. Debnath, D. H. C. Du, Y. Kim, and Y. Kim. An on-line hot data identification for flash-based storage using sampling mechanism. *SIGAPP Appl. Comput. Rev.*, 13(1):51–64, Mar. 2013.

[26] Y. Park and J.-S. Kim. zftl: Power-efficient data compression support for nand flash-based consumer electronics devices. *IEEE transactions on consumer electronics*, 57(3):1148–1156, 2011.

[27] S. Suzuki, Y. Deguchi, T. Nakamura, and K. Takeuchi. Endurance-based dynamic vthdistribution shaping of 3d-tlc nand flash memories to suppress both lateral charge migration and vertical charge de-trap and increase data-retention time by 2.7x. In *2018 48th European Solid-State Device Research Conference (ESSDERC)*, pages 150–153, 2018.

[28] S. Tanakamaru, C. Hung, and K. Takeuchi. Highly reliable and low power ssd using asymmetric coding and stripe bitline-pattern elimination programming. *IEEE Journal of Solid-State Circuits*, 47(1):85–96, 2012.

[29] R. Tatman. Speech accent archive, Nov 2017.

[30] T. Tokutomi, S. Tanakamaru, T. O. Iwasaki, and K. Takeuchi. Advanced error prediction ldpc for high-speed reliable tlc nand-based ssds. In *2014 IEEE 6th International Memory Workshop (IMW)*, pages 1–4, 2014.

[31] K.-K. Yong and L.-P. Chang. Error diluting: Exploiting 3-d nand flash process variation for efficient read on ldpc-based ssds. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3467–3478, 2020.

[32] M. Zhao, J. Li, Z. Cai, J. Liao, and Y. Shi. Block attribute-aware data reallocation to alleviate read disturb in ssds. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1096–1099, 2021.

ABOUT THE AUTHORS:

Chin-Hsien Wu received his B.S. degree in computer science from National Chung-Cheng University in 1999 and his M.S. and Ph.D. degrees in computer science from National Taiwan University in 2001 and 2006, respectively. He is currently a full professor in the Department of Electronic and Computer Engineering at National Taiwan University of Science and Technology. He is also a member of ACM and IEEE. His research interests include embedded software, (non-volatile) Memory Systems, and flash-memory storage systems.

Yan-Qi Liao received his Master degree in the electronic and computer engineering from National Taiwan University of Science and Technology in 2022. His research interests include embedded systems and flash-memory storage systems.

Yung-Cheng Huang is currently a graduate student in the department of electronic and computer engineering in National Taiwan University of Science and Technology. His research interests include embedded systems and flash-memory storage systems.