CeSR: A Cell State Remapping Strategy to Reduce Raw Bit Error Rate of MLC NAND Flash

Yutong Zhao, Wei Tong‡, Jingning Liu, Dan Feng, and Hongwei Qin
Wuhan National Lab for Optoelectronics
Huazhong University of Science and Technology, Wuhan, China
Email:{ytzhao, Tongwei, jnliu, dfeng, glqhw}@hust.edu.cn

***Abstract*—Retention errors and program interference errors have been recognized as the two main types of NAND flash errors. Since NAND flash cells in the erased state which hold the lowest threshold voltage are least likely to cause program interference and retention errors, existing schemes preprocess the raw data to increase the ratio of cells in the erased state. However, such schemes do not effectively decrease the ratio of cells with the highest threshold voltage which are most likely to cause program interference and retention errors. In addition, we note that the dominant error type of flash varies with data hotness. Retention errors are not too much of a concern for frequently updated hot data while cold data that is rarely updated need to worry about the growing retention errors as P/E cycles increase. Furthermore, the effects of these two types of errors on the same cell partially counteract each other. Given the observation that retention errors and program interference errors are both cell-state-dependent, this paper presents a cell state remapping (CeSR) strategy based on the error tendencies of data with different hotness. For different types of data segments, CeSR adopts different flipping schemes to remap the cell states in order to achieve the least error-prone data pattern for written data with different hotness. Evaluation shows that the proposed CeSR strategy can reduce the raw bit error rates of hot and cold data by up to 20.30% and 67.24%, respectively, compared with the state-of-the-art NRC strategy.**

***Index Terms*—NAND flash memory, flash reliability, retention error, program interference error, cell state remapping, RBER**

I. Introduction

NAND flash memory is widely used because of its high storage density, non-volatility and fast read/write speed. With the continuing semiconductor process scaling and the introduction of muti-level storage technology, the density of NAND flash has been greatly increased. However, the aggressive increase of NAND flash memory density comes at the cost of shrunk lifetime and reduced reliability.

NAND flash cells are prone to errors. After writing data to flash cells, the fraction of incorrect bits is referred to as the raw bit error rate (RBER). A common approach to handling NAND flash errors is to use error correction codes (ECCs). The bit error rate after applying ECCs is referred as the uncorrectable bit error rate (UBER) which is an important reliability metric of NAND flash. As flash errors continuingly increase, LDPC codes with promising error correction performance have attracted much attention and are used to meet the demand of error correction capability [1] [2]. However, when the RBER is high, using LDPC codes will lead to a noticeable increase in read latency. In addition, UBER is highly related to RBER and strength of ECCs. Therefore, reducing RBER is crucial as it can improve the primal reliability of flash and improve the performance of ECCs.

A MLC cell can store 2 bits according to its threshold voltage (V_{th}) [3]. Specifically, data value of one MLC NAND flash memory cell can be represented as "11", "10", "00" and "01", from the lowest to the highest V_{th} state. Retention errors and program interference errors are the two dominant error types in NAND flash, which will change a cell's V_{th} and may alter the state of the cell, that is, causing data errors. Retention errors are mainly caused by electron leakage of programmed flash cells over time which decrease V_{th}. Program interference errors are mainly caused by excessive electron injections during programming one cell, which increase V_{th} of its neighboring cells. It has been shown that both these two errors are cell-state-dependent [4]. Specifically, retention errors mainly occur in the two states with higher V_{th} (i.e., "01", "00") in MLC NAND flash. In addition, programming a cell with higher V_{th} (i.e., "01", "00") will cause more program interference to its neighboring cells.

Programming NAND flash with randomized data is one of the most efficient ways to make the probability of the worst-case data pattern statistically negligible [5]. The on-chip randomizer of NAND flash can randomize input data from an external device at programming such that ratios of 1s and 0s in the raw data are equal in probability.

As mentioned above, the RBER is mainly attributed to retention errors and program interference errors. There are a lot of works devoted to reducing RBER of flash in different ways. For example, many prior works optimize the read reference voltage according to the error-related factors [6]–[9]. In addition, flash refresh techniques limit the number of retention errors to achieve flash reliability improvement [10]–[12]. Furthermore, many existing works preprocess the raw data to decrease flash errors [13]–[15]. In the paper, we also focus on preprocessing the raw data.

According to the above analysis of these two errors, reducing cells with higher V_{th} can decrease both types of errors. In particular, increasing the ratio of erased state "11" which

‡ Co-corresponding author:
Wei Tong (Tongwei@hust.edu.cn)

holds the lowest V_{th} among four states of MLC NAND flash will decrease these two errors more effectively. From the perspective of preprocessing the raw data, existing works propose different strategies which aim at increasing the ratio of state "11" by raising 1's ratio in the raw data. Guo *et al.* [13] develop a data-pattern-aware (DPA) error protection strategy which uses decorrelation and scrambling techniques on the raw data to increase the ratio of 1s. However, for the weakly correlated data, the complex polynomial fitting operations will bring extra power consumption and delay. The asymmetric coding (AC) scheme [14] intercepts a data segment to compare the ratios of 0s and 1s. If 0's ratio is higher than 1's, then flip the data segment and the flip flag bit "1" is appended to the data segment. Otherwise, the data segment is not flipped and the flag bit is "0". However, the per-segment flip flags will bring high storage overhead. Besides, they may cause the data length to be out of alignment with the page size, which may exacerbate write amplification in flash memory [16]. Wei *et al.* [15] propose a nibble remapping coding (NRC) method which counts the ratios of all nibbles and then remaps the nibbles based on the ratios to increase 1s in the raw data. Although the NRC strategy can effectively reduce the RBER, the access to the remapping table will cause additional delay and remapping table errors are unavoidable.

Since MLC NAND flash cells with lower threshold voltage are less likely to cause program interference and less vulnerable to retention errors, the common logic behind DPA, AC and NRC is to increase the ratio of erased state "11" which holds the lowest V_{th} by increasing the ratio of 1s in the raw data. Their results show that increasing the ratio of state "11" in flash cells can effectively decrease these two errors. However, we argue that there are three important points that should also be considered.

(1) Decreasing the ratio of state "01" which holds the highest V_{th} among the four states is also important for reducing flash errors, especially for retention errors.
(2) The effects of these two errors partly counteract each other because they cause V_{th} to move in the opposite direction. However, existing strategies only consider these two types of errors separately despite the fact that they exist simultaneously in reality. According to our experiments, when considering both types of errors simultaneously, simply increasing the ratio of 1s cannot effectively decrease the RBER. In some cases, they may even increase the RBER. We will deeply discuss this topic in Section II-B.
(3) The implementation of these strategies must disable the randomizer of NAND flash because they increase the ratio of 1s in the data. Consequently, they may lead to a decrease in flash reliability, which will be shown in Section IV-D.

In this paper, we propose a novel cell state remapping (CeSR) strategy based on error tendency analysis of cells in different states to enhance the reliability of MLC NAND flash. The rule of CeSR is to achieve the least error-prone data pattern for written data with different hotness. Specifically, CeSR chooses different flipping schemes for different types of data segments according to 1's ratio and hotness of the data segment, thereby adjusting the ratios of four states due to error tendencies, so as to reduce the RBER ultimately. Different from the existing schemes, CeSR does not increase the ratio of 1s in the raw data directly, but focuses on error characteristics of each state. In addition, the existing strategies ignore the partial counteractions between these two errors. In contrast, our proposed CeSR strategy leverages program interference to mitigate retention errors.

The major contributions beyond existing works are as follows:

(1) We achieve decreasing the ratio of state "01" while increasing the ratio of state "11" for hot data.
(2) We consider counteractions between these two errors and leverage program interference to mitigate retention errors for cold data.
(3) We can decrease the RBER more effectively than existing works under the most unfavorable situation for our proposed CeSR strategy.
(4) We can achieve reliability improvement of NAND flash despite disabling the on-chip randomizer.

The rest of this paper is organized as follows. Section II introduces some basics of NAND flash and analyzes the effect of increasing 1s in the raw data. Section III describes the proposed CeSR strategy and its implementation. The simulation results and comparison of CeSR with the existing strategies are given in Section IV. Finally, we conclude our paper in Section V.

II. Background and Motivation

In this section, we introduce the NAND flash error model used in our work and the on-chip randomizer. After that we analyze the change of each state and the RBER of flash memory as the ratio of 1s in the raw data increases. We then summarize our views based on the analysis and present our motivation of this work.

A. Error Model of MLC NAND Flash

In MLC NAND flash, one flash cell can contain two bits of data. A MLC NAND flash cell can be divided into four logical states (i.e., "11", "10", "00" and "01") according to its V_{th}. The left bit belongs to the least significant bit (LSB) page and the right bit belongs to the most significant bit (MSB) page [9]. Pages within a block that share the same flash cells are referred to as shared pages. Each LSB shared page must be programmed before the MSB shared page of that pair can be programmed. For hot data, Program/Erase (P/E) stress is dominant and program interference errors are more obvious. For cold data that are infrequently updated, retention errors become dominant as data retention time increases [17].

The formation process of threshold voltage distribution of NAND flash memory can be modeled in Fig. 1 [18]. Ideally data erasing and programming result in the ideal threshold voltage distribution, as shown in Fig. 2(a), then the threshold

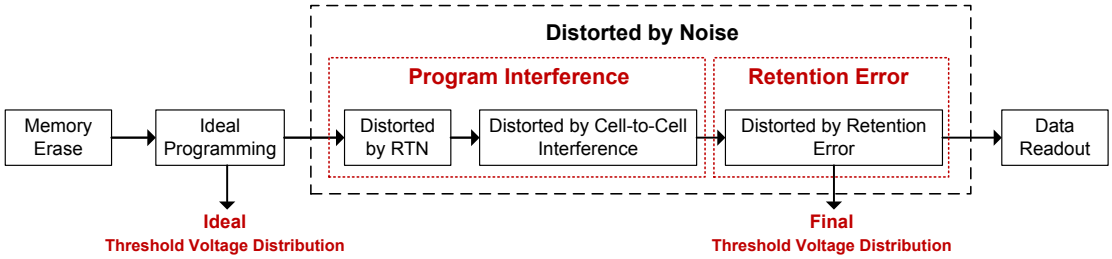
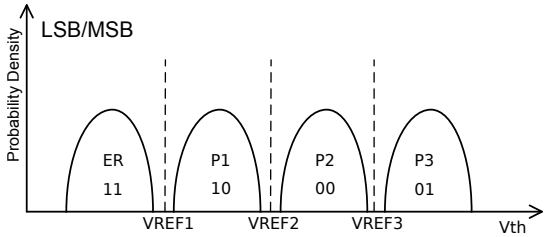
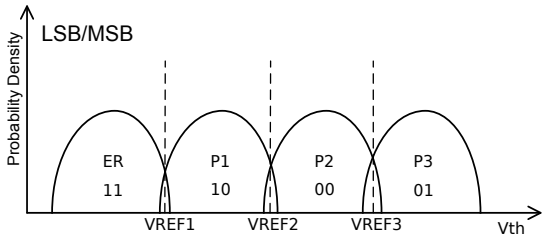


Fig. 1. Error Model of NAND Flash [18]



(a) The ideal threshold voltage distribution .



(b) The final threshold voltage distribution.

Fig. 2. Threshold voltage distribution before and after distorted

voltage distribution will be significantly distorted by noise and the final distribution is formed which is shown in Fig. 2(b). During read operations, a cell's threshold voltage can be determined to be between the most recent two read reference voltages. Fig. 2 shows the three default read reference voltages of MLC NAND flash (i.e., VERF1, VREF2 and VREF3). Noises lead to V_{th} shift, which causes voltage windows to overlap. When reading the data, it is difficult to determine which voltage window the default reference voltage falls into, therefore bit flip occurs.

One flash cell must be erased before it is programmed, and its V_{th} will be set to the lowest voltage window (i.e., erased state "11"). It is well known that the threshold voltage of erased cells tends to have a wide Gaussian-like distribution [19]. Hence, the threshold voltage distribution of the erased state can be approximately modeled as

$$p_e(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{-\frac{(x-\mu_e)^2}{2{\sigma_e}^2}}. \tag{1}$$

where μ_e and σ_e are the mean and standard deviation of the erased state threshold voltage, respectively.

When programing flash cells, a tight threshold voltage control is typically realized by using incremental step pulse program (ISPP) [20], i.e., all the flash cells on the same word-line are recursively programmed using a program-and-verify approach with a stair case program word-line voltage V_{pp}. Let ΔV_{pp} denote the incremental program step voltage. For the kth programmed state with the verify voltage $V_p^{(k)}$, ideally ISPP program results in a uniform threshold voltage distribution:

$$p_p^{(k)}(x) = \begin{cases} \frac{1}{\Delta V_{pp}}, & if\ V_p^{(k)} \le x \le V_p^{(k)} + \Delta V_{pp} \\ 0, & else \end{cases} \tag{2}$$

For MLC NAND flash, the threshold voltage distribution of each state can be expressed as:

$$p(x) = \begin{cases} p_e(x), & k = 0 \\ p_p^{(k)}(x), & k = 1, 2, 3 \end{cases} \tag{3}$$

Unfortunately, the above ideal threshold voltage distribution can be significantly distorted by noise in practice. Previous works identified program interference and retention errors as two major noises that cause V_{th} distortion.

*1) **Program Interference**:* Program interference comes from the combined effect of random telegraph noise (RTN) and cell-to-cell interference. RTN is caused by electrons capture and emission during P/E cycling, which directly results in flash cell V_{th} fluctuation. Assume that λ represents the mean value of V_{th} shift ΔV_{RTN}. The probability density function of ΔV_{RTN} can be modeled by

$$p_{\Delta V_{RTN}} = \frac{1}{2\lambda} e^{-\frac{|\Delta V_{RTN}|}{\lambda}}. \tag{4}$$

Here N denotes the P/E cycles. λ scales with N in an approximate power-law distribution. We set the parameter $\lambda = K_\lambda \cdot N^{0.5}$ according to Reference [18].

Parasitic capacitance-coupling effect causes cell-to-cell interference, which incurs V_{th} shift of the victim cell when its adjacent cells are programmed. In emerging 3D NAND flash, one cell has adjacent cells in three directions, i.e., X-direction, Y-direction and Z-direction. In the all-bit-line (ABL) flash architecture where all cells on the same word-line are read and programmed at the same time, there is no interference in X-direction or even in diagonal direction. So, interference mainly comes from the Y-direction and Z-direction. Each LSB page is always programmed prior to its corresponding MSB page, so the effects of cell-to-cell interference after programing LSB pages and MSB pages are related to page distribution in MLC NAND flash. Fig. 3 shows an example of page construction, it can be seen that the numbers of a LSB page and its corresponding MSB page are not consecutive. The effects of cell-to-cell interference for LSB pages and MSB

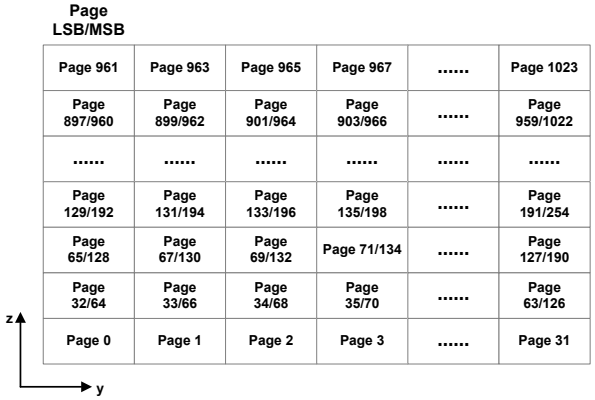


Fig. 3. Y-Z view of flash page construction [21]

pages are considered separately. The V_{th} shift ΔV_{CCI} caused by cell-to-cell interference in the ABL structure after LSB program is described as [21]

$$\Delta V_{CCI}(n,m) = \gamma_y\left(\Delta V_{m+1}^{lsb}\right) + \gamma_y\left(\Delta V_{m-1}^{msb}\right) \quad (5)$$

After MSB program it is described as

$$\Delta V_{CCI}(n,m) = \gamma_y\left(\Delta V_{m+1}^{msb}\right) + \gamma_z\left(\Delta V_{n+1}^{lsb}\right) + \gamma_z\left(\Delta V_{n+1}^{msb}\right) \quad (6)$$

where m and n are the position of one flash page in the Y-direction and Z-direction, respectively. γ_y and γ_z represent the floating gate coupling ratios in the Y-direction and Z-direction, respectively. ΔV_{m+1}^{lsb} and ΔV_{m+1}^{lsb} represent V_{th} shifts of the adjacent m+1 interfering cell whose LSB page or MSB page is programmed after the victim cell, respectively.

The effect of cell-to-cell interference is cell-state-dependent. The largest V_{th} distortion of the victim cell occurs when the interfering cell is programmed to states with higher V_{th}.

*2) **Retention Error**:* Retention errors result from electron escaping, leading to V_{th} decrease. As retention time increases, retention errors become more serious. The V_{th} shift ΔV_{RET} can be modeled by the normal distribution $N\left(\mu_d, {\sigma_d}^2\right)$, μ_d and ${\sigma_d}^2$ can be expressed by

$$\begin{cases} \mu_d = K_s\left(x - x_0\right) K_d N^{0.5} ln\left(1 + \frac{t}{t_0}\right) \\ \sigma_d^2 = K_s\left(x - x_0\right) K_m N^{0.6} ln\left(1 + \frac{t}{t_0}\right) \end{cases} \quad (7)$$

Here x_0 is the V_{th} of state "11" and x is the initial V_{th} after programming. K_s, K_d and K_m are constants. N is the number of P/E cycles and t is retention time. t_0 is an initial time and can be set as 1 hour. Equation shows that higher initial V_{th} incurs larger V_{th} shift. Hence, cells in state "01" is most vulnerable to retention errors.

B. Analysis of Increasing Ratio of 1s

The DPA, AC and NRC strategies are dedicated to increasing 1's ratio in the raw data so as to increase the ratio of state "11", thus they can decrease program interference errors and retention errors to some extent. We can get how the ratio of each state varies as 1's ratio increases through Monte Carlo simulation. It can be seen from Fig. 4(a) that the

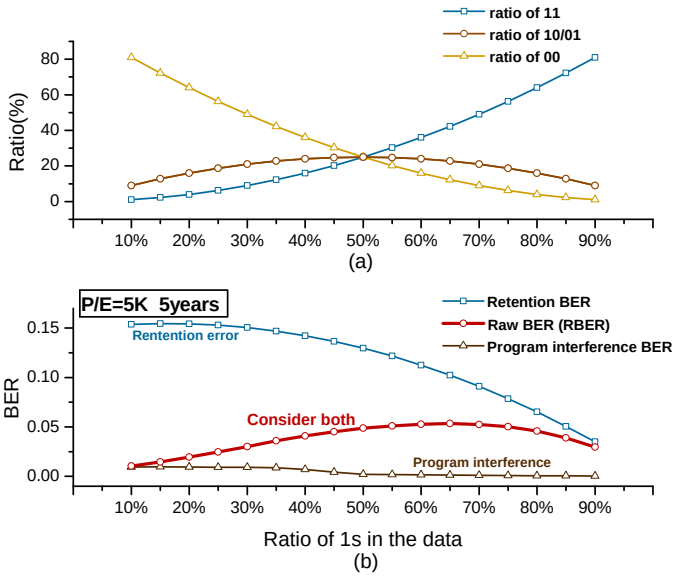


Fig. 4. Analysis of increasing ratio of 1s

ratio of state "11" increases significantly, the ratios of state "10" and "01" are equal, and the ratio of state "00" gradually decreases as 1's ratio increases. However, as can be seen from Fig. 4(b), retention errors for cold data (e.g., P/E cycles=5K, retention time is 5 years) are still serious since the ratio of state "01" where retention errors are most likely to occur doesn't decrease significantly. In the case of comprehensively considering program interference and retention errors, increasing 1's ratio does not always reduce the RBER since these two errors partly counteract. Therefore, we consider how to adjust the ratio of each state and minimize the ratio of state "01" which is most likely to cause program interference and most prone to retention errors, and reduce the RBER of flash memory more effectively in consideration of these two errors accordingly.

C. On-Chip Randomizer of NAND Flash

It has been noted that flash memory manufacturers apply on-chip data randomizer to equalize the ratios of 1s and 0s to guarantee flash reliability. As mentioned earlier, programming NAND flash with randomized data can make the probability of the worst-case data pattern statistically negligible. Specifically, on-chip randomizer simply inserts exclusive OR (XOR) gate to data-in/out path to get randomized data.

However, the precondition of AC, DPA, NRC and our CeSR strategy is to disable the randomizer because they all unbalance the ratios of 1s and 0s in the raw data. How will the reliability of NAND flash be affected after disabling the randomizer? We will answer this question specifically in IV-D.

D. Motivation

Data retention and program interference errors have been identified as the two most common errors in NAND flash, and they cause V_{th} to move in the opposite direction. Previous work based on preprocessing the raw data focused on directly increasing the ratio of 1s in the data, thereby increasing the

ratio of state "11" that doesn't cause program interference and is least prone to retention errors. However, they don't effectively reduce the ratio of state "01" with the highest V_{th}. Moreover, they separately evaluate the bit error rates of program interference and data retention without taking into account the counteraction between these two errors. In this paper, we propose a CeSR strategy based on the characteristics of the errors in each cell to adjust the ratio of each state, thereby reducing the probability of data errors and decreasing the RBER. In addition, the CeSR strategy leverages the counteraction between program interference errors and retention errors, which is discussed in depth in section III.

III. The Proposed Cell State Remapping Strategy

In this section, we present the proposed CeSR strategy based on the error characteristics of each state.

A. Design Goal of CeSR

For hot data that are programmed frequently, program interference errors are dominant. MLC NAND flash cells with higher V_{th} are more likely to cause program interference and more vulnerable to retention errors. Since state "11" ("01") holds the lowest (highest) V_{th} among the four states, we hope to increase the ratio of the state "11" as much as possible while reducing the ratio of state "01".

For cold data, data retention errors are dominant and the states with higher V_{th} have lower reliability. A decrease of state "01" can effectively reduce the RBER which is dominated by retention errors. However, unlike hot data, we attempt to increase the ratio of state "10" instead of state "11". This is because data retention errors and program interference errors cause the V_{th} to move in the opposite direction, the larger ratio of state "10" will cause more serious program interference which can mitigate retention errors caused by electron leakage. However, previous strategies don't consider program interference errors when retention errors are dominant.

B. Proposed Cell State Remapping Strategy

The architecture of CeSR is depicted in Fig.5. The CeSR strategy consists of three parts: 1) Data splitting; 2) Data flipping and 3) Flag bits compressing.

*1) **Data splitting**:* The programming data of one page will be split into N data segments. For a data segment, the length of it is calculated in Equation 8.

$$Data\ Segment\ Length = \frac{Page\ Size}{N} \tag{8}$$

The maximum data segment length is equal to the page size (i.e., N=1). When N is greater than 1, the length of data segment is smaller than the page size, and N in Equation 8 must be a positive integer.

*2) **Data flipping**:* Data segment is the process unit of CeSR. For each data segment, flip the data according to different flipping sub-schemes prior to programming them into a LSB or MSB page. As shown in Table I, for different kinds of data segments, we present four flipping sub-schemes, namely H0, H1, C0 and C1, each of which is used for one type of data segment. For example, H0 is used for **H**ot data segment with more **0**s than 1s.

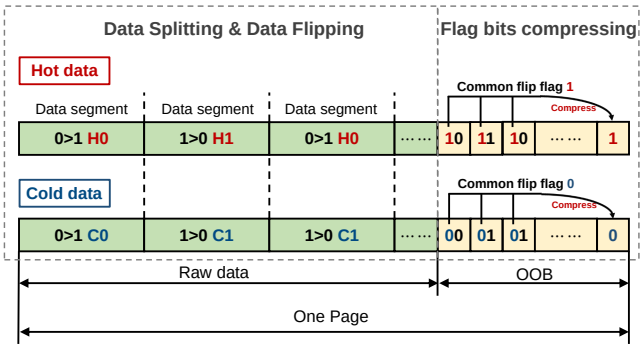


Fig. 5. System architecture of CeSR

TABLE I
SUB-SCHEMES OF THE CESR STRATEGY

Four sub-schemes		*Data hotness*	
		Hot data	Cold data
1's ratio	<50%	**H0**	**C0**
	>50%	**H1**	**C1**

For a data segment, the four sub-schemes process the data segment which will be written to a LSB page or MSB page in different ways. The brief processing method is shown in Table II. Detailed description is as follows:

(1) **Sub-scheme H0**: For a hot data segment of which 1's ratio is less than 50%, if the data will be written to a LSB page, flip them. Otherwise flip the bits of which the corresponding LSB is 1.

(2) **Sub-scheme H1**: For a hot data segment of which 1's ratio is greater than 50%, if the data will be written to a MSB page, flip the bits of which the corresponding LSB is 0, otherwise do not process them.

(3) **Sub-scheme C0**: For a cold data segment of which 1's ratio is less than 50%, if the data will be written to a LSB page, flip them, otherwise do not process them.

(4) **Sub-scheme C1**: For a cold data segment of which 1's ratio is greater than 50%, if the data will be written to a MSB page, flip them, otherwise do not process them.

Each LSB page shares the same flash cells with its corresponding MSB page, thus there are four combinations of flipping sub-schemes for hot data and cold data, respectively. Fig. 6 shows the four combinations for hot data. They are

TABLE II
PROCESSING METHOD OF THE FOUR SUB-SCHEMES

	H0	H1	C0	C1
witten to a LSB page	flip	/	flip	/
witten to a MSB page	flip (if LSB=1)	flip (if LSB=0)	/	flip

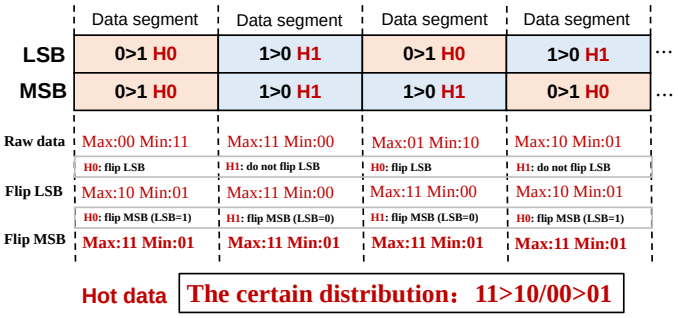


Fig. 6. Combination of flipping sub-scheme for hot data[1]

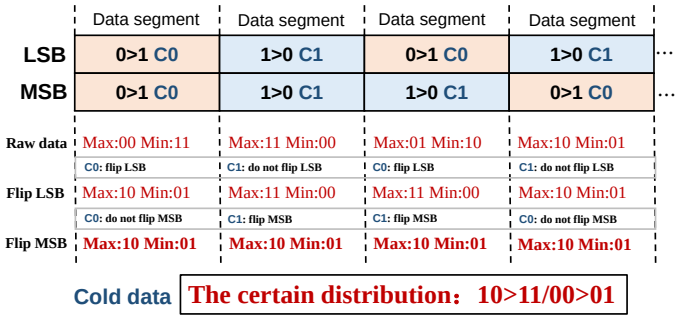


Fig. 7. Combination of flipping sub-scheme for cold data

H0H0, H1H1, H0H1 and H1H0, each of which is one type of flipping sub-scheme combination. For example, H0H1 represents that the flipping sub-scheme is H0 for the data segment which will be written to the LSB page, and H1 for the data segment which will be written to the corresponding MSB page. Next, we will analyze the results of these four combinations for hot data respectively.

(1) **Combination H0H0**: When 0's ratios of the data segments which will be written to the LSB and MSB page are both greater than 1's, the number of cells in state "00" is the maximum and that in state "11" is the minimum statistically. After flipping the data segment which will be written to the LSB page, the number of cells in state "10" is the maximum and that in state "01" is the minimum. Then flip the data segment which will be written to the MSB page when the corresponding LSB is 1, thus the number of cells in state "11" is the maximum and that in state "01" is the minimum.

(2) **Combination H1H1**: When 1's ratios of the data segments which will be written to the LSB and MSB page are both greater than 0's, the number of cells in state "11" is the maximum and that in state "00" is the minimum statistically. According to sub-scheme H1, only flip the data segment which will be written to the MSB page when the corresponding LSB is 0, thus the number of cells in state "11" is the maximum and that in state "01" is the minimum.

(3) **Combination H0H1**: When 0's ratios of the data segments which will be written to the LSB and MSB page are greater and less than 1's, respectively, the number of cells in state "01" is the maximum and that in state "10" is the minimum statistically. After flipping the data segment which will be written to the LSB page, the number of cells in state "11" is the maximum and that in state "00" is the minimum. Then flip the data segment which will be written to the MSB page when the corresponding LSB is 0, thus the number of cells in state "11" and is the maximum and that in state "01" is the minimum.

(4) **Combination H1H0**: When 1's ratios of the data segments which will be written to the LSB and MSB page are greater and less than 0's, respectively, the number of cells in state "10" is the maximum and that in state "01" is the minimum statistically. Flip the data segment which will be written to the MSB page when the corresponding LSB is 1, thus the number of cells in state "11" is the maximum and that in state "01" is the minimum.

For cold data, there are also four flipping sub-scheme combinations which are C0C0, C1C1, C0C1 and C1C0, respectively. Similarly, we can obtain the results after CeSR for each combination.

(1) **Combination C0C0**: When 0's ratios of the data segments which will be written to the LSB and MSB page are both greater than 1's, the number of cells in state "00" is the maximum and that in state "11" is the minimum statistically. After flipping the data segment which will be written to the LSB page, the number of cells in state "10" is the maximum and that in state "01" is the minimum. And there is no additional processing for the data segment which will be written to the MSB page.

(2) **Combination C1C1**: When 1's ratios of the data segments which will be written to the LSB and MSB page are both greater than 0's, the number of cells in state "11" is the maximum and that in state "00' 'is the minimum statistically. According to sub-scheme C1, only flip the data segment which will be written to the MSB page, thus the number of cells in state "10" is the maximum and that in state "01" is the minimum.

(3) **Combination C0C1**: When 0's ratios of the data segments which will be written to the LSB and MSB page are greater and less than 1's, respectively, the number of cells in state "01" is the maximum and that in state "10" is the minimum statistically. Flip the data segment which will be written to the LSB page and MSB page, thus the number of cells in state "10" is the maximum and that in state "01" is the minimum.

(4) **Combination C1C0**: When 1's ratios of the data segment which will be written to the LSB and MSB page are greater and less than 0's, respectively, the number of cells in state "10" is the maximum and that in state "01" is the minimum statistically. In this case, there is no need to process the data segments.

In summary, for hot data, the ratio of state "11" is the largest and the ratio of state "01" is the smallest after flipping.

[1]In order to clearly describe the strategy, in the figure, we use "flip LSB (MSB)" as flipping the data segment which will be written to LSB (MSB) pages, and it is the same in Fig. 7.

As is shown in Fig. 6, the distribution of four states is "11>10/00>01" for hot data, and it has nothing to do with the combination of data segments. Different from hot data, the ratio of state "10" is the largest and the ratio of state "01" is the smallest after flipping for cold data. As is shown in Fig. 7, the distribution of four states is "10>11/00>01" for cold data. We are delighted to see that the distributions of data patterns for hot/cold data are consistent with our design goals.

*3) **Flag bits compressing**:* We use two flags "11", "10", "00" and "01" to represent the four sub-scheme H0, H1, C0 and C1, respectively, as shown in Table III. The flags are used to choose the corresponding scheme when recovering the raw data and they should be stored in the out-of-band (OOB) area of each page. We can find that there is a common flag bit 1 (0) for hot data (cold data). For each page, the flag bits can be compressed by deleting the left common bit and append it to the end of all flags, as shown in Fig. 5.

For one page, the number of flag bits is N+1. For example, when the length of the data segment is equal to the page size, the flip flag bits stored in OOB only requires 2 bits (1+1=2 bits) of memory space. When the length of a data segment is one quarter of the page size, that is, N=4, the flip flag bits require 5 bits (4+1=5 bits). It can be seen that storing the flip flags only consumes negligible extra space.

C. Analysis

DPA, AC and NRC directly increase the ratio of 1s of the total raw data, thus increasing the ratio of state "11" statistically. Different from their strategies, we focus on flash states and process data at page or finer granularity based on the error tendencies of flash states. In addition, for cold data, we try to maximize the ratio of state "10" to offset more retention errors. However, the other strategies do not take into account the counteractions between program interference and retention errors.

D. Implementation

1) Implementation of CeSR: The architecture design of CeSR is shown in Fig. 8. For hot data, when writing data to a LSB page, the data need to be cached in the buffer before writing its corresponding MSB page. Then put the data segment into the counting circuit and flipping circuit. When writing data to MSB pages, flip the data segment according to the corresponding LSB cached. For cold data, there is no need to cache the data which have been written to the corresponding

TABLE III
FLIP FLAGS FOR FOUR FLIPPING SUB-SCHEMES

	Sub-scheme	Flag	Common flag
Hot data	H0	**1**1	**1**
	H1	**1**0	
Cold data	C0	**0**0	**0**
	C1	**0**1	

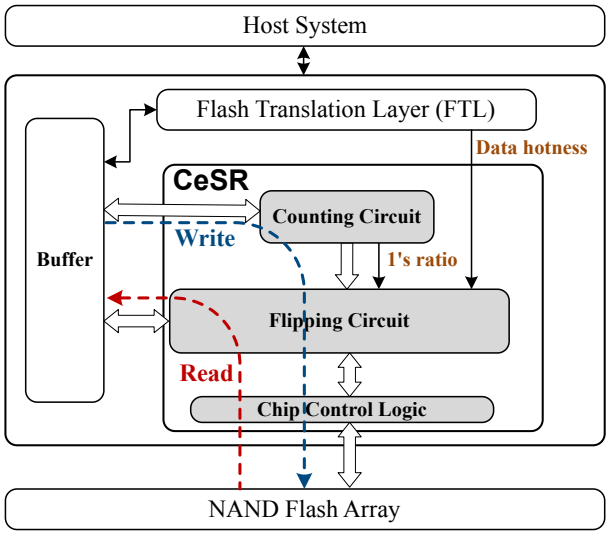


Fig. 8. An illustration of CeSR design

LSB pages. First get 1's ratio of the data segment through the counting circuit, and then flip the data before writing to LSB/MSB pages.

The flags are used to choose the corresponding reversed scheme when reading the data. Reading one MSB page of hot data requires reading the corresponding LSB page at the same time, which is not necessary in other cases.

When the length of one data segment is reduced, that is, N is increased, pipeline processing can be used in the counting and flipping circuits to reduce writing time.

2) Overhead analysis:

- Device memory overhead: For hot data, the data which have been written to LSB pages need to be cached before the corresponding MSB pages are programmed. According to the technical reference manual from Micron [22], for one pair of shared pages of Micron chips, the LSB page and MSB page numbers are not consecutive, as shown in Fig. 3. For example, LSB page 32 and MSB page 64 share the same cell. The maximum gap is 63 (e.g., LSB page 65 and MSB page 128). Before writing to a MSB page, the LSB pages numbered between the MSB page and its shared LSB page have to be cached, that is, up to 63 LSB pages need to be cached. Assume that there are 4 channels and each channel contains 8 dies in an SSD of 16KB-page-size flash, the required extra cost of buffer space is: $63\times32\times16\text{KB}/1024=31.5\text{MB}$.
- Latency overhead: According to the technical reference manual from Micron [22], the time it takes to read data from the flash array pages to the data register is 78μs, and from the data register to the cache register is 11μs. When reading MSB page of hot data, the corresponding LSB page needs to be read at the same time, causing read latency. The READ PAGE CACHE RANDOM command can read the MSB page into the data register while the corresponding LSB page is output from the cache register to reduce the latency. Therefore, the time to read a MSB

page of hot data is: 78+11+11=100μs, and the time is 78μs in other cases. As can be seen from Table II, two of the eight cases have the read latency. Assume that the amounts of hot data and cold data are equal, thus a quarter of the total raw data have 22μs (i.e., 100-78=22) read latency. The read latency increases approximately 7% (22μs/78μs$\div$4$\times$100%=7%).

IV. Evaluation

In this section, we simulate the flash channel using the introduced NAND flash error model by leveraging the parameters in Reference [18] [21]. The parameters used are listed in Table IV. The page size used in our test is 16KB. We test common types of files including different ratios of 1s in matlab environment in the following experiment. We note that .dll is the system file type of the largest amount under current operating system of Windows, .doc, .ppt and .pdf files are typical office files, and .eml is the mail file type that is often used in office work.

A. Test of Hot Data

1) Data pattern comparison: For hot data, the goal of CeSR is to increase the ratio of "11" while reducing the ratio of "01". Fig. 9 shows an example of how data pattern changes after CeSR. Specifically, in Fig. 9, the ratio of 1s in the raw hot data (P/E cycles=5K, retention time is 1 day) is 20% and the process unit is one page (i.e., N=1). We can get ratios of the four states through Monte Carlo simulation. In the raw data, "11", "10", "00" and "01" occupy 4%, 16%, 64% and 16% of the total data, respectively, and the ratio is 1:4:16:4. After CeSR, ratios of state "11", "10", "00" and "01" are 64%, 16%, 16% and 4%, respectively, and the ratio is 16:4:4:1. In a word, CeSR realizes remapping of the cell states which is in line with our design goal. It can be clearly seen that V_{th} distortion after CeSR becomes smaller, thus the RBER will decrease through qualitative analysis.

2) Test under different P/E cycles: Since these all types of errors are highly correlated with P/E cycles, to evaluate the effectiveness of CeSR on hot data under different degrees of program interference, we test the variation of RBERs of four

TABLE IV
Parameters of MLC NAND flash error model

Parameter	Value	Parameter	Value
μ_e	1.4	γ_z	0.038
σ_e	0.35	K_s	0.333
ΔV_{pp}	0.3	x_0	1.4
$V_p{}^{(1)}$	2.85	K_d	4×10^{-4}
$V_p{}^{(2)}$	3.55	K_m	2×10^{-6}
$V_p{}^{(3)}$	4.25	VREF1	2.65
K_λ	4×10^{-4}	VREF2	3.35
γ_y	0.033	VREF3	4.05

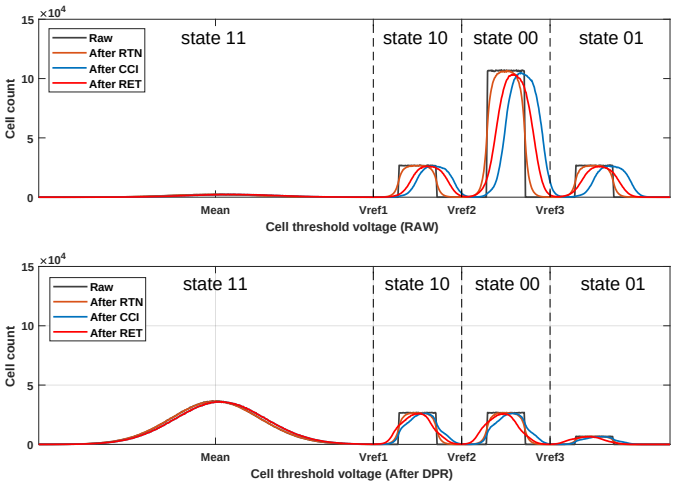


Fig. 9. Data patterns of hot data (1's ratio:20%) before and after CeSR

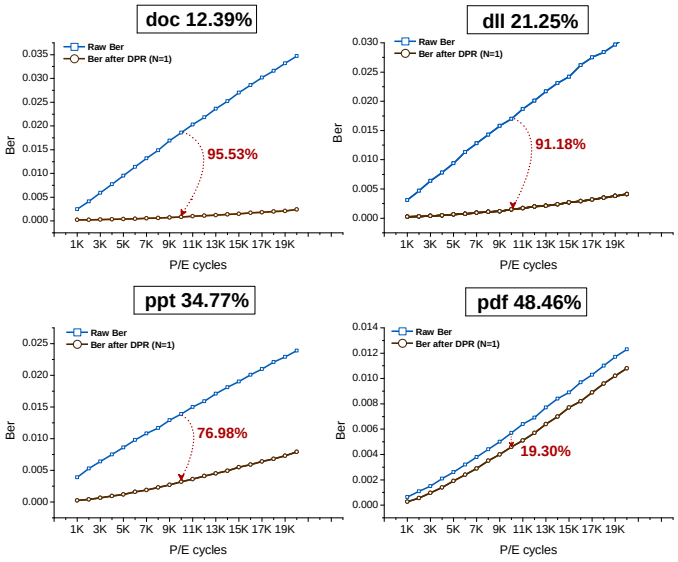


Fig. 10. BER of different files before and after CeSR

different files of which 1's ratios vary greatly when P/E cycles are from 1K to 20K. Fig. 10 compares the RBER before and after the CeSR strategy, the horizontal axis indicates the P/E cycles and the vertical axis indicates the RBER. N=1 means the longest length of CeSR process unit. It can be observed that the RBER is significantly reduced after the CeSR strategy through quantitative analysis. Specifically, when N=1 and P/E cycles are 10K, CeSR decreases the RBER by 95.53% for the selected .doc file (1's ratio is 12.39%), 91.18% for the selected .dll file (1's ratio is 21.25%), 76.98% for the selected .ppt file (1's ratio is 34.77%) and 19.30% for the selected .pdf file (1's ratio is 48.46%).

3) Different lengths of data segment: To evaluate the effectiveness of different processing granularities, we test the variation of RBERs with different lengths of data segment. Fig. 11 shows the BER of hot data in the five cases, and the larger N indicates finer granularity of CeSR process unit. We can draw two conclusions from Fig. 11. First, the BER

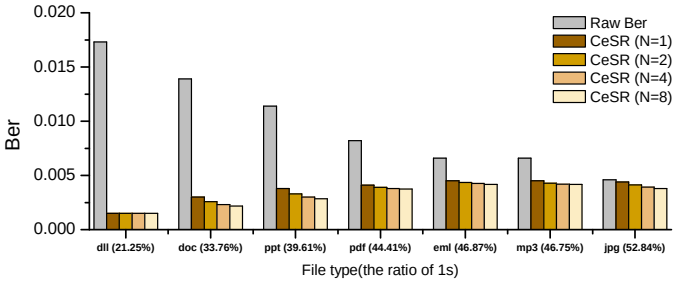


Fig. 11. BER using different process granularities of CeSR for hot data

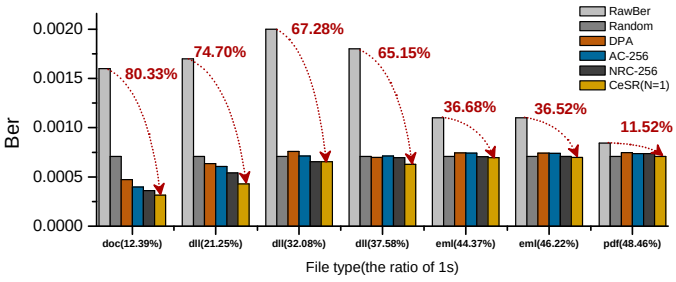


Fig. 12. BER using different strategies for hot data

after CeSR is significantly reduced compared with the raw BER. Second, while the process unit becoming smaller, the BER decreases slowly. Specifically, N=1 means the maximum process unit of CeSR, i.e., one page, and program interference can be greatly restrained in this case.

4) Comparison of different strategies: In order to compare the effectiveness of CeSR with other strategies that are also dedicated to data preprocessing, we test the RBER before and after Random, DPA, AC, NRC and CeSR strategies for different types and sizes of files when P/E cycles are 5K and retention time is 1 day. Here, Random means the data is randomized by the on-chip randomizer. AC-256 denotes the code length is 256 bits in the AC strategy and NRC-256 indicates the code length is 256 bytes in the NRC strategy. It is worth noting that N=1 is the most unfavorable situation for our CeSR strategy. Fig. 12 shows that our proposed CeSR strategy reduces the RBER of hot data more efficiently than the other strategies in the most unfavorable situation. Specifically, CeSR decreases the RBER by up to 80.33% in the selected files. The more 1's ratio in the raw data deviates from 50%, the better CeSR performs. In addition, it can be seen that when 1's ratio is very close to 50%, the CeSR strategy can also reduce the RBER effectively. Compared with the state-of-the-art NRC strategy, CeSR reduces the RBER by up to 20.30% for hot data.

B. Test of Cold Data

1) Data pattern comparison: For cold data, the goal of CeSR is to increase the ratio of "10" while reducing the ratio of "01". Fig. 13 shows an example of how data pattern changes after CeSR. Specifically, in Fig. 13, the ratio of 1s in the raw cold data (P/E cycles=5K, retention time is 5 years) is 20% and the process unit is one page (i.e., N=1). Through Monte

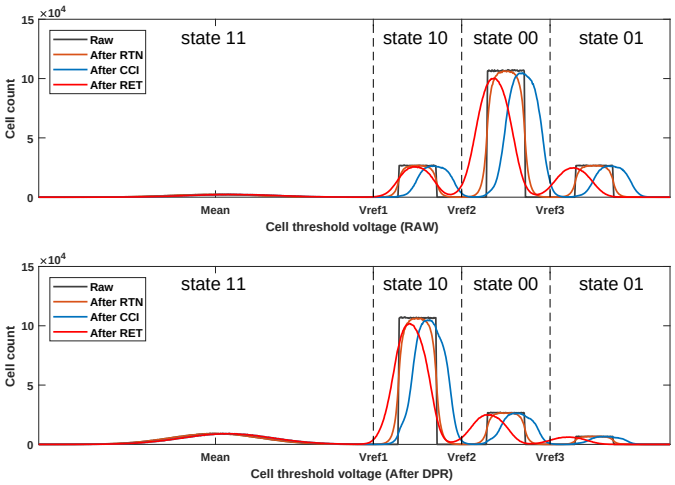


Fig. 13. Data patterns of cold data (1's ratio:20%) before and after CeSR

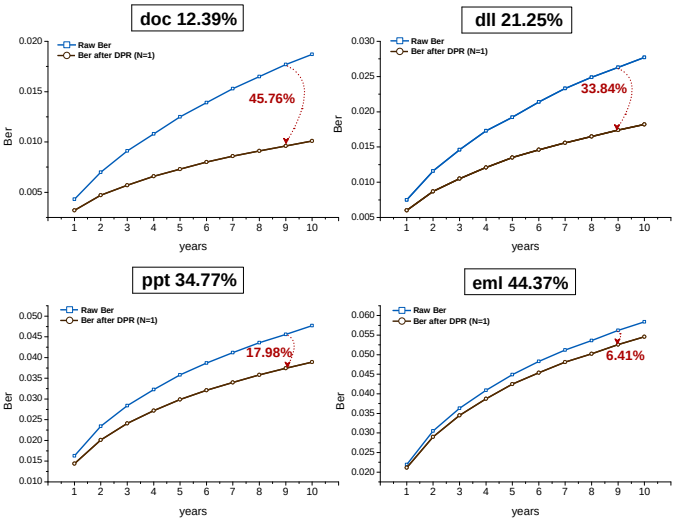


Fig. 14. BER of different files before and after CeSR

Carlo simulation, we can know that "11", "10", "00" and "01" occupy 4%, 16%, 64% and 16% of the total data, respectively, and the ratio is 1:4:16:4. After CeSR, ratios of state "11", "10", "00" and "01" are 16%, 64%, 16% and 4%, respectively, and the ratio is 4:16:4:1. Similar to hot data, the remapping of the cell states is consistent with our design goal and the RBER seems to decrease because of the reduction of V_{th} distortion.

2) Test under different retention time: To evaluate the effectiveness of CeSR on cold data, we test the variation of RBERs of four different files when P/E cycles are 5K and retention time is from 1 to 10 years. Fig. 14 exhibits that the RBER significantly reduces after CeSR. To be specific, when N=1 and retention time is 9 years, the CeSR strategy decreases the RBER by 45.76% for the selected .doc file (1's ratio is 12.39%), 33.84% for the selected .dll file (1's ratio is 21.25%) and 17.98% for the selected .ppt file (1's ratio is 34.77%). For the selected .eml file of which 1's ratio is 44.37% (close to 50%), the RBER reduction is relatively small, being 6.41%.

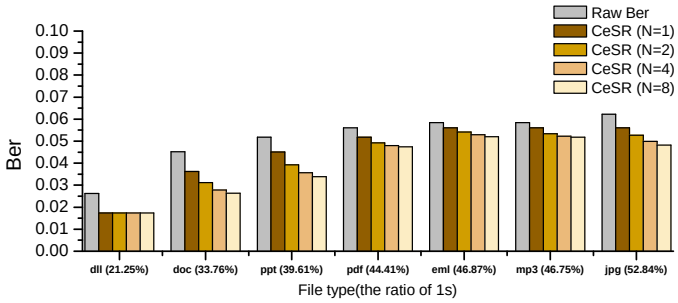


Fig. 15. BER using different process granularities of CeSR for cold data

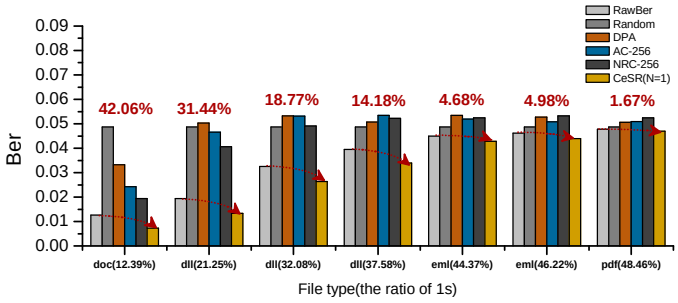


Fig. 16. BER using different strategies for cold data

3) Different lengths of data segment: Fig. 15 shows the BER of cold data in different processing granularities. Compared with hot data, the CeSR strategy on cold data is less effective. In addition, we can see that the increase of N has more returns on decreasing the RBER for cold data than hot data. The reason is that the BER of cold data is reduced less than hot data when N=1.

4) Comparison of different strategies: To compare the effectiveness of CeSR with other strategies, we test the RBER before and after Random, DPA, AC, NRC and CeSR strategies for different files when P/E cycles are 5K and retention time is 5 years. We compare these strategies in the most unfavorable situation for the CeSR strategy (i.e., N=1). Fig. 16 shows the RBER of cold data with program interference. It can be seen that the RBER increases as 1's ratio in the raw data increases since program interference partly counteracts retention errors. In this case, DPA, AC and NRC which increase 1's ratio in turn increase the RBER. It can also be seen from Fig. 4(b) that increasing 1's ratio does not necessarily reduce the RBER when considering both two errors. Whereas our proposed CeSR strategy minimizes the ratio of state "01" that is most prone to retention errors and leverages program interference to mitigate retention errors, ultimately reducing the RBER. Fig. 16 shows that CeSR performs well when 1's ratio is low. For example, CeSR decreases the RBER by 42.06% for the selected .doc file (1's ratio is 12.39%). The RBER slightly decreases when 1's ratio is close to 50%, but we achieve better performance with less cost compared with other strategies. Compared with the state-of-the-art NRC strategy, CeSR reduces the RBER by up to 67.24% for cold data.

In conclusion, the more 1's ratio deviates from 50%, the better the CeSR strategy performs. For the raw data of which 1's ratio is closer to 50%, the increase of N can obtain more returns on decreasing the RBER. Furthermore, CeSR can be implemented with acceptable overheads.

C. Flag Bits Overhead Analysis

When the page size is 16KB, the flag bits consume 64 bits of user space in the AC strategy when the code length is 256 bits. For the NRC strategy, the number of flag bits in the OOB is 512 bits when the code length is 256 bytes. For our CeSR strategy, the number of flag bits in the OOB is 2 bits when N=1, and 9 bits when N=8. In comparison, our storage overhead is much lower.

D. Analysis of Disabling the Randomizer

The on-chip randomizer is applied to ensure the reliability of NAND flash. From Fig. 12 and Fig. 16 we can see that DPA, AC and NRC strategies perform worse than randomizing data in many cases, especially for cold data. When these three strategies decrease the RBER less than randomizing data, disabling the on-chip randomizer is harmful to flash reliability. Different from them, our proposed CeSR strategy performs better than randomizing data, that is, flash reliability can be improved by CeSR in spite of disabling the randomizer.

V. Conclusion

In this paper, we proposed a novel CeSR strategy that remaps the cell states to adjust the ratio of each state in programming data, thus reducing the probability that data errors occur. As a result, program interference and data retention errors of 3D MLC NAND flash are restrained and the RBER is decreased. Compared with DPA, AC and NRC, our proposed CeSR strategy leverages the counteraction between program interference and retention errors, and it can achieve more reliability improvement. Experimental results show that the proposed CeSR strategy can reduce the raw bit error rates of hot and cold data by up to 20.30% and 67.24%, respectively, compared with the state-of-the-art NRC strategy. In the future, we plan to verify the strategy on a real hardware platform and cooperate with ECC to further ensure flash reliability.

Acknowledgment

This work was supported by the National High Technology Research and Development Program (863 Program) No.2015AA016701; the National Natural Science Foundation of China under Grant 61832007, Grant 61821003, Grant 61772222, and Grant U1705261; the Shenzhen Research Funding of Science and Technology under Grant JCYJ20170307172447622. This work was also supported by Engineering Research Center of Data Storage Systems and Technology, Ministry of Education, China.

References

[1] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang, and N. Zheng, "LDPC-in-SSD: making advanced error correction codes work effectively in solid state drives." in *11th USENIX Conference on File and Storage Technologies (FAST)*, vol. 13, 2013, pp. 244–256.

[2] M. Zhang, F. Wu, X. He, P. Huang, S. Wang, and C. Xie, "REAL: A retention error aware LDPC decoding scheme to improve NAND flash read performance," in *32nd Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2016, pp. 1–13.

[3] 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)*. IEEE, 2017, pp. 49–60.

[4] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in MLC NAND flash memory: measurement, characterization, and analysis," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 521–526.

[5] C. Kim, J. Ryu, T. Lee, H. Kim, J. Lim, J. Jeong, S. Seo, H. Jeon, B. Kim, I. Lee *et al.*, "A 21 nm high performance 64 Gb MLC NAND flash memory with 400 MB/s asynchronous toggle DDR interface," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 4, pp. 981–989, 2012.

[6] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data retention in MLC NAND flash memory: characterization, optimization, and recovery," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 551–563.

[7] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "HeatWatch: improving 3D NAND flash memory device reliability by exploiting self-recovery and temperature awareness," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 504–517.

[8] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: characterization, analysis, and modeling," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1285–1290.

[9] Y. Cai, O. Mutlu, E. F. Haratsch, and K. Mai, "Program interference in MLC NAND flash memory: characterization, modeling, and mitigation," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*. IEEE, 2013, pp. 123–130.

[10] Y. Luo, Y. Cai, S. Ghose, J. Choi, and O. Mutlu, "WARM: improving NAND flash memory lifetime with write-hotness aware retention management," in *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2015, pp. 1–14.

[11] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. S. Unsal, and K. Mai, "Flash correct-and-refresh: retention-aware error management for increased flash memory lifetime," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 2012, pp. 94–101.

[12] Y. Pan, G. Dong, Q. Wu, and T. Zhang, "Quasi-nonvolatile SSD: trading flash memory nonvolatility to improve storage system performance for enterprise applications," in *2012 IEEE 18th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2012, pp. 1–10.

[13] J. Guo, D. Wang, Z. Shao, and Y. Chen, "Data-pattern-aware error prevention technique to improve system reliability," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 4, pp. 1433–1443, 2017.

[14] M. Doi, S. Tanakamaru, and K. Takeuchi, "A scaling scenario of asymmetric coding to reduce both data retention and program disturbance of NAND flash memories," *Solid-State Electronics*, vol. 92, pp. 63–69, 2014.

[15] D. Wei, L. Deng, P. Zhang, L. Qiao, and X. Peng, "NRC: a nibble remapping coding strategy for NAND flash reliability extension," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 11, pp. 1942–1946, 2016.

[16] Y. Lu, J. Shu, W. Zheng *et al.*, "Extending the lifetime of flash-based storage through reducing write amplification from file systems," in *11th USENIX Conference on File and Storage Technologies (FAST)*, vol. 13, 2013.

[17] 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)*. IEEE, 2017, pp. 161–164.

[18] Y. Pan, G. Dong, and T. Zhang, "Exploiting memory device wear-out dynamics to improve NAND flash memory system performance," in *9th USENIX Conference on File and Storage Technologies (FAST)*, vol. 11, no. 2011, 2011, pp. 18–18.

[19] K. Takeuchi, T. Tanaka, and H. Nakamura, "A double-leve1-Vth select gate array architecture for multilevel NAND flash memories," *IEICE Transactions on Electronics*, vol. 79, no. 7, pp. 1013–1020, 1996.

[20] K. D. Suh, B. H. Suh, Y. H. Lim, J. K. Kim, Y. J. Choi, Y. N. Koh, S. S. Lee, S. C. Kwon, B. S. Choi, J. S. Yum *et al.*, "A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.

[21] Y. Feng, D. Feng, W. Tong, Y. Jiang, and C. Liu, "Using disturbance compensation and data clustering (DC)2 to improve reliability and performance of 3D MLC flash memory," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 565–572.

[22] Micron Technology, "MLC 256Gb to 1Tb Async/Sync NAND features."