Reducing MLC Flash Memory Retention Errors through Programming Initial Step Only

Wei Wang
Computational Science Research Center
San Diego State University
San Diego, CA 92115
Email: wang@rohan.sdsu.edu

Tao Xie
Computer Science Department
San Diego State University
San Diego, CA 92115
Email: txie@mail.sdsu.edu

Antoine Khoueir, Youngpil Kim
Seagate Technology
Shakopee, MN 55379
Email: antoine.khoueir@seagate.com
youngpil.kim@seagate.com

***Abstract*—Retention error has been recognized as the most dominant error in MLC (multi-level cell) flash. In this paper, we propose a new approach called PISO (Programming Initial Step Only) to reduce its number. Unlike a normal programming operation, a PISO operation only carries out the first programming-and-verifying step on a programmed cell. As a result, a number of electrons are injected into the cell to compensate its charge loss over time without disturbing its existing data. Further, we build a model to understand the relationship between the number of PISOs and the number of reduced errors. Experimental results from 1y-nm MLC chips show that PISO can efficiently reduce the number of retention errors with a minimal overhead. On average, applying 10 PISO operations each month on a one-year-old MLC chip that has experienced 4K P/E cycles can reduce its retention errors by 21.5% after 3 months.**

I. Introduction

NAND flash memory (hereafter, flash memory) has been extensively used in various computer systems because of its attractive features such as high I/O performance and low energy consumption [1], [2]. To largely reduce NAND flash memory cost in dollars per GB, manufacturers are aggressively scaling down memory cell size and pushing each cell to store more bits [3]. However, these technologies substantially lower the reliability and endurance of flash memory. For example, a typical 70-nm technology SLC (single-level cell) flash can survive ∼100k P/E (programming/erase) cycles with a minimum 2-bit per 1 KB ECC capacity (i.e., 2 bit errors can be corrected per 1 KB data), whereas a sub 50-nm technology 2-bit MLC flash can only tolerate no more than 10k P/E cycles with a minimum 8-bit per 1 KB ECC capacity [3]. The decreasing reliability demands stronger ECC schemes whose capacity increasing rate may not be able to keep the same pace with the shrinking lithography [4]. Reducing bit errors, therefore, becomes a critical way to improve the reliability of flash memory under a certain ECC capacity.

A flash memory cell is a floating-gate MOS transistor [1] (see Fig. 1a). The floating-gate of a cell stores a number of electrons, which affects the cell's threshold voltage V_{th}. The value of V_{th} is measured to determine the state of a cell [5]. A 2-bit MLC cell can store 2-bit data with four different states (i.e., $S_0, S_1, S_2,$ and S_3, see Fig. 1b). In this paper, we only investigate 2-bit MLC flash. A programming operation is to charge a cell to a particular threshold voltage level, whereas an erase operation is to remove charges stored in each cell's

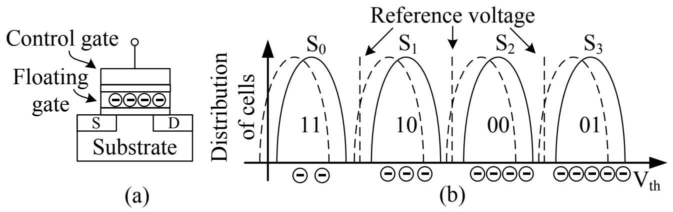


Fig. 1: (a) A cell; (b) MLC threshold voltage distribution.

floating-gate. If a cell's V_{th} shifts across the boundary of its reference voltage(s), its data will be misinterpreted, and thus, an error happens [4]. A flash memory error could occur due to various noise sources like cell-to-cell interference, random telegraph noise, and data retention [6]. A retention error is caused by electron leakage and de-trapping phenomenon [7] over time, which shift the V_{th} of a programmed cell to a lower level across its left reference voltage (see the four dotted curves in Fig. 1b). Retention error has been identified as the dominant flash memory error [8], [6]. Obviously, the number of retention errors enlarges as the retention time increases. To compensate a cell's charge loss over time, and thus, correct its retention error, an intuitive solution is to inject an appropriate amount of electrons into a cell so that its V_{th} can be pushed back to its original level.

Compared with a read operation, a programming operation provides a more effective means of charging a cell because it applies a higher voltage on the control gate [6]. A programming operation uses the ISPP (incremental step pulse programming) method to lift a cell's threshold voltage to an expected level [4]. Fig. 2 illustrates how ISPP changes the threshold voltage of a cell. The expected V_{th} is the threshold voltage, to which a cell is going to be programmed. ISPP consists of a series of programming-and-verifying steps. In each step, a programming voltage, V_{program}, is first applied to raise a cell's threshold voltage by injecting a number of electrons. Next, a verifying process is carried out to compare the cell's current V_{th} with the expected V_{th}. If the current V_{th} does not reach the expected V_{th}, a subsequent programming-and-verifying step is executed to further increase the cell's V_{th}. Otherwise, the programming process is terminated. ΔV_{pp}, the V_{program} increase between two adjacent steps, affects programming speed for programming time can be shortened if it is increased [4].

MLC cell programming is complicated because the two bits

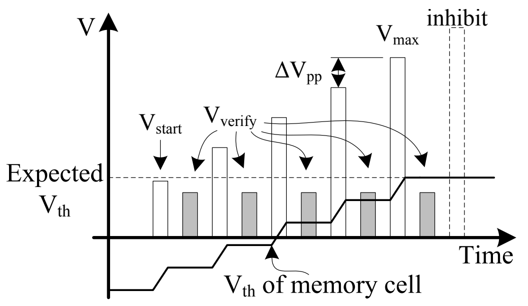


Fig. 2: Incremental step pulse programming (ISPP).

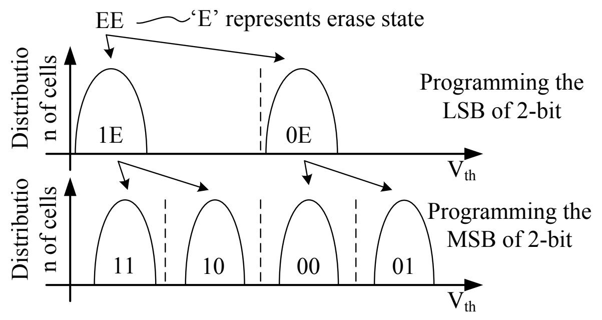

EE
'E' represents erase state
Distribution of cells
1E
0E
Programming the LSB of 2-bit
V_th
Distribution of cells
11
10
00
01
Programming the MSB of 2-bit
V_th

Fig. 3: Programming an MLC cell.

of an MLC cell belong to two different pages, and thus, have to be programmed separately. The most significant bits (MSBs) of all cells in a block form MSB pages, whereas the least significant bits (LSBs) are grouped to LSB pages. Programming an LSB page is faster than programming an MSB page because the latter requires to first read its corresponding LSB page, and then, a smaller ΔV_{pp} is applied to accurately control the precision of the final threshold voltage level [4]. Fig. 3 shows an MLC cell programming procedure. For example, after programming '0' to the LSB of an erased cell, its LSB (i.e., the left bit) becomes '0' while its MSB (i.e., the right bit) remains in the erase state 'E'. If we continue to program '1' to the cell's MSB, the cell's current state (i.e., '0E') will be first read out. And then a further threshold voltage increase will be added on top of it so that the threshold voltage of the cell reaches the final '01' state. This is how the data '01' is programmed into an MLC cell.

After a cell is newly programmed, its V_{th} must be within one of the four threshold voltage ranges (i.e., S_0, S_1, S_2, and S_3 in Fig. 1b). If we deliberately program the data corresponding to a *safe threshold voltage* (i.e., a voltage lower than the cell's current V_{th}, see Section III-A) into the cell, based on the programming-and-verifying mechanism the programming operation will be immediately terminated after the first verifying process for the cell's V_{th} is already higher than the expected V_{th}. As a result, the data stored in the cell remains unchanged. This one step programming process, however, can correct retention errors as it injects some electrons into the cell. Inspired by this observation, we develop PISO (Programming Initial Step Only), a new approach to reducing retention errors. Its basic idea is to periodically program the data corresponding to a safe V_{th} into a programmed cell. The injected electrons can partially compensate charge loss over time so that retention error can be mitigated. PISO incurs a minimum overhead as it can program a cell *in-place* without requiring prior knowledge of the data stored. Besides, it can be readily implemented in an SSD device driver or the firmware of an SSD controller. Major contributions of this paper include: (1) We propose a new approach called PISO that can correct retention errors with a low cost; (2) We build an analytical model to understand the relationship between the number of PISOs and the number of reduced retention errors; (3) We conduct experiments on modern 1y-nm MLC chips from two vendors; experimental results show that PISO lowers retention errors by 21.9% for a 4K-P/E-cycled flash memory chip after 3 month retention baking; (4) We qualitatively compare PISO with existing schemes to illustrate its advantages and limitations.

The remainder of the paper is organized as follows. Section II briefly summarizes the related work. Section III introduces the PISO approach. An analytical model derived from the threshold voltage distribution of flash memory is introduced in Section IV to calculate an appropriate number of PISO operations. Section V evaluates the effectiveness of the PISO approach and compares it with the read disturb. Section VI discusses how to apply PISO on real applications. The paper is concluded in Section VII.

II. Related Work

A few retention error reduction schemes [8], [9], [10], [11] have been recently reported in the literature. A dynamic threshold scheme for flash memory was developed by Sala *et al.* to reduce retention errors [9]. After a V_{th} shifts to its left, the scheme pushes reference voltages to the same direction so that the shifted V_{th} can still be correctly interpreted. However, finding a suitable reference voltage change typically requires a series of read retry operations with different reference voltage settings, which incurs a high overhead [9]. Moreover, as the corrected data has to be written to a new place, the block with the original data will be reclaimed later, which consumes one P/E cycle, and thus, reduces its lifetime [9]. Ma *et al.* proposed a retention error correcting scheme called RE-WPD (Reliability Enhancing through Word-line Program Disturbance) [11]. Each RE-WPD operation writes 0xFF sequentially on all least significant bit (LSB) pages in a selected block so that cells in the selected word line are all biased in the program inhibit status due to the 0xFF data pattern. As a result, a small amount of electrons are injected into the cells floating gates because the high voltage between the control gate and the channel induces slightly FN (Fowler-Nordheim) tunneling [11]. The RE-WPD operation can be repeatedly applied multiple times in order to inject more elections for the data retention error recovery purpose. Unlike the approach presented in [11], PISO directly utilizes the programming operation and can be applied on both MSB and LSB pages.

Cai *et al.* proposed a flash correct-and-refresh (FCR) technique, which periodically corrects retention errors by reading, correcting, and re-programming (in-place) [8]. To avoid the over-programmed problem, after a number of in-place programming operations it moves the data to a new page [8]. However, they found that the remapping-based FCR technique could negatively affect the performance of a read-intensive

TABLE I: The layout of an MLC block (256 pages x 16 KB).

Row Index	LSB of the 2^{17} cells	MSB of the 2^{17} cells
0	page 0	page 2
1	page 1	page 4
2	page 3	page 6
...	...	...
126	page 251	page 254
127	page 253	page 255

application due to the fact that it incurs considerable data movement and unnecessary refreshes although the average flash memory lifetime can be significantly improved. To reduce the overhead of data movement caused by reprogramming, they further proposed hybrid FCR. The key idea is that a page can be refreshed by reprogramming it in-place instead of by re-mapping it to another location. This is based on the insight that retention errors are caused by charge loss and cells with retention errors are re-programmable without first erasing them. To further reduce unnecessary refreshes, they also proposed adaptive-rate FCR, which has a low refresh rate for a flash block when its retention error rate is low (i.e., early in its lifetime) [8]. Nevertheless, dynamically monitoring the number of over-programmed errors and determining a good timing of a data migration degrade performance and reduce P/E cycles [8]. Tanakamaru *et al.* introduced an error reduction scheme by using read disturb effect in flash [10]. Their scheme is named DRRP (data-retention error-recovery pulse), which is applied to the memory cells. With DRRP, electrons are injected into the floating gate, and thus, DRRP mitigates the data-retention errors. Unfortunately, it has to perform hundreds of read operations (see Section V for details), which results in a high overhead. A low-cost yet effective retention error reduction approach is greatly needed.

III. The PISO Approach

A. The Safe Threshold Voltage

Table I shows the page layout of an MLC block, which consists of 128 rows of 2^{17} cells. These cells are mapped to 256 pages with each of them being 16 KB. Page 0 and page 2 share the first row of 2^{17} cells. To program page 0, an LSB page, the threshold voltage that represents data '1' is the safe threshold voltage as it is the lowest V_{th} among the two possible V_{th} ranges (see 'LSB programming' in Fig. 3). Because only one programming-and-verifying step is executed, programming data '1' to an LSB page will not change its current data, and thus, it is safe. Programming page 2, an MSB page, demands a read operation of the cells' current states. If a cell's V_{th} is within '1E' state (i.e., the cell's LSB is '1'), it can only be further programmed to either '11' state (i.e., the cell's MSB is '1') or '10' state (i.e., the cell's MSB is '0'). Since the V_{th} of state '11' is lower than that of state '10', writing data '1' to the MSB of the cell is equivalent to programming it to the safe threshold voltage. Similarly, if a cell's LSB stores '0', programming data '0' to the cell's MSB equals to programming it to the safe threshold voltage (see 'MSB programming' in Fig. 3). In other words, for an LSB page, data '1' always represents the safe V_{th}. For an MSB page, the data stored in its associated LSB page represent each individual cell's safe V_{th}.

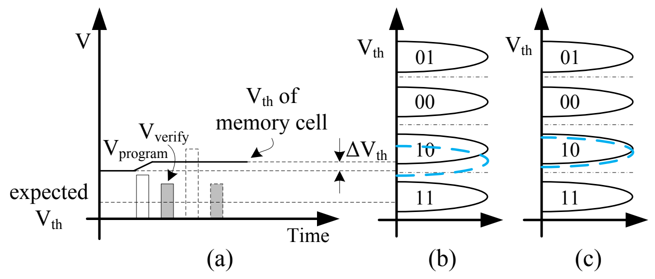


Fig. 4: (a) A PISO operation; (b) before; (c) after PISO.

B. A PISO Operation

PISO exploits the first programming-and-verifying step in a programming operation by programming the data corresponding to the safe V_{th}. Fig. 4 illustrates a PISO operation and the threshold voltage change before and after applying it. Assuming that a cell is programmed to the '10' state, after a certain period of retention time a retention error occurs as its threshold voltage shifts left into the range of state '11' (see the dotted curve in Fig. 4b). Now, a PISO operation programs data '1' (i.e., the safe V_{th}) to the cell's LSB. The expected V_{th} (i.e., the safe V_{th}) is shown in Fig. 4a. The PISO operation will be immediately terminated after the first programming-and-verifying step because the cell's V_{th} exceeds the expected V_{th}. Consequently, the V_{th} of the cell is pushed back to its original V_{th} range (i.e., '10' state), and thus, the retention error is corrected (see the dotted curve in Fig. 4c).

C. PISO on LSB/MSB Pages

MLC uses different programming procedures for the LSB and MSB of a cell. Hence, applying the PISO operation on an LSB page is different to that of on an MSB page. To correct a cell's retention error, a PISO operation can be applied on either an LSB page or its associated MSB page because they share the same memory cells. Applying a PISO operation on an LSB page can be simply carried out by programming data '1' to the page. In contrast, applying a PISO operation on an MSB page is complicated as the data on its associated LSB page need to be first read out. And then, the data are programmed to the MSB page. Compared with an LSB page PISO operation, the overhead of performing an MSB page PISO operation is much higher because: (1) an MSB page PISO operation requires an extra page read operation; (2) programming an MSB page demands more clock cycles due to the internal programming mechanism (typically, programming an MSB page is $3 \sim 4$ times slower than programming an LSB page [4]). Besides, the retention error correction capabilities of the two types of PISO operations are similar as both of them execute one programming-and-verifying step, which injects a close number of electrons. Based on the analysis above, an LSB page PISO operation is superior to an MSB page PISO operation. Thus, we only carry out LSB page PISO operations in all our experiments shown in Section V.

D. Preventing the Excessive Use of PISO

Although a PISO operation can reduce retention errors, an excessive use of it may generate new errors. First of all, a

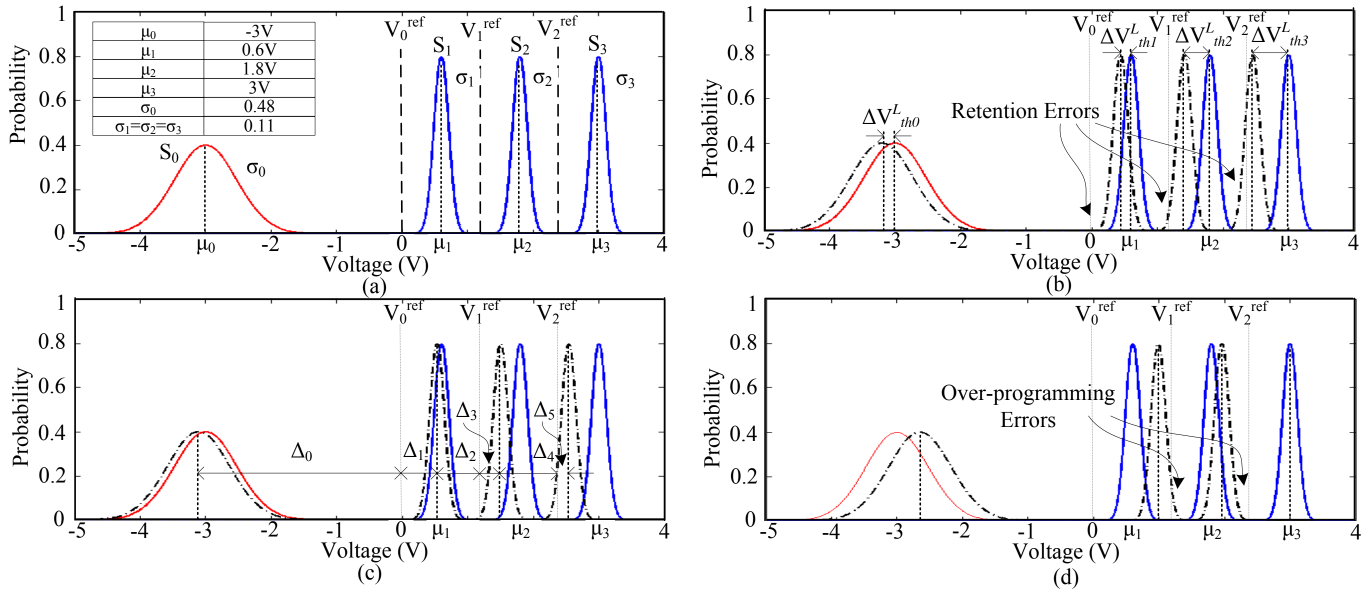
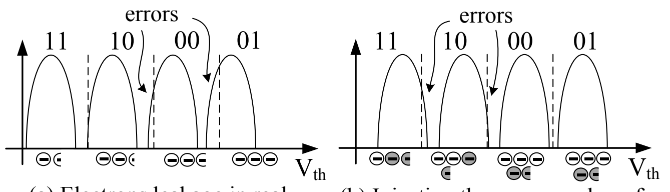


Fig. 6: V_{th} distributions; (a) normal; (b) left shifted; (c) with inadequate number of PISOs; (d) over-programmed.



(a) Electrons leakage in real applications.

(b) Injecting the same number of electrons.

Fig. 5: Threshold voltage distribution shifts.

programming operation may also introduc programming errors. Secondly, memory cells that are programmed to a higher threshold voltage (i.e., state '01' and state '00') incur more retention errors [6] (see Fig. 5). Cai *et al.* [6] show that 46% and 44% of total retention errors are state '01' $\rightarrow$ state '00' and state '00' $\rightarrow$ state '10' errors, respectively. This observation can be explained by the mechanisms of retention errors. The electron loss during retention can be attributed to two reasons: the first one is the stress induced leakage current (SILC) and the second one is the relaxation of interface traps [6]. As memory cells in higher threshold voltage states (e.g., state '00' and state '01') have a larger number of electrons, SILC is higher and therefore more electrons leak away from these cells. The threshold voltage drift, therefore, is more large in these cells. As shown in Fig. 5a, the decrease of threshold voltage level in memory cells with state '01' is the largest, whereas the threshold voltage levels in memory cells with state '11' do not have obvious shift. Thus, injecting the same amount of electrons into all memory cells cannot push the threshold voltage distributions back to their original positions. Even worse, the threshold voltage of cells in a lower state (i.e., '11' or '10') may be over-pushed, and thus, new errors are generated (see Fig. 5b). Thus, an appropriate number of PISO operations needs to be carefully chosen.

IV. An Analytical Model

In this section, we derive an analytical model to understand the relationship between the number PISOs and the number of reduced errors.

The threshold voltage distribution of flash memory follows a sum of Gaussian distributions model [4]:

$$\begin{aligned} f(x) &= \sum_{s=0}^{3} P(S) f_s(x) \\ &= \sum_{s=0}^{3} \frac{1}{4\sqrt{2\pi}\delta_s} exp\{\frac{-(x-\mu_s)^2}{2\delta_s^2}\}, \end{aligned} \quad (1)$$

where $P(S)$ is the probability of state S. Ideally, $P(S)$ for each state is approximately equal to $\frac{1}{4}$ in an MLC flash because there are a large amount of memory cells in one flash chip. μ_s is the mean V_{th} value of state S. The distribution model of a standard MLC flash is illustrated in Fig. 6a and all the parameters used are set according to [4]. It is obvious that the state S_0 has a flattened bell shape while the distributions of other three states are much taller and tighter. V_0^{ref}, V_1^{ref}, and V_2^{ref} are reference voltages used to differentiate the four different states. When a V_{th} leaves its original range and crosses a reference voltage, an error happens (see Fig. 6b). It is clear that the cells that are at the tail of distribution incur errors when a threshold voltage changes.

A higher threshold voltage results in a higher SILC, which leads to a larger loss of electrons. As a result, ΔV_{th}^L (i.e., the left shift amount of a cell's V_{th}) is in direct proportion to its original V_{th}. Also, it is well understood that a longer retention time results in more retention errors. Thus, ΔV_{th}^L for state S can be estimated as below:

$$\Delta V_{th,S}^L = \alpha(t) \cdot V_{th,S} \ , \quad (2)$$

where $V_{th,S}$ is the original V_{th} of a cell in state S and $\alpha(t)$ is the retention time.

The original V_{th} distance between state S_0 and state S_3 is $(V_{th,3}-V_{th,0})$. After retention time $\alpha(t)$, the threshold voltages of the four states all shift to left. Their shift amount can be calculated by equation (2). Thus, the new distance between the state S_0 and state S_3 can be derived as follows:

$$\begin{aligned}&(V_{th,3}-\Delta V^{L}_{th,3})-(V_{th,0}-\Delta V^{L}_{th,0})\\ =&(1-\alpha(t))\cdot V_{th3}-(1-\alpha(t))\cdot V_{th0}\\ =&(1-\alpha(t))(V_{th,3}-V_{th,0}).\end{aligned} \tag{3}$$

Clearly, the charge loss shrinks the distance between $V_{th,3}$ and $V_{th,0}$. Since the three reference voltages keep unchanged, applying PISO operations to push the four states back to their original positions has three possible consequences: (1) The number of PISO operations is insufficient. As shown in Fig. 6c, in this case the V_{th} of state S_1 could be pushed back to its correct range, whereas state S_2 or S_3 could not because no enough electrons are injected; (2) The V_{th} of a state is over-pushed across its right boundary for too many PISO operations are carried out. Fig. 6d illustrates this situation. In this scenario, state S_3 is fully pushed to its initial V_{th} range. However, cells whose V_{th} are originally in state S_1 and S_2 are over-charged. Their V_{th} pass the upper boundaries, which incurs over-programmed errors. (3) An appropriate number of PISO operations are carried out so that a maximum number of errors are reduced.

We assume that each PISO operation can shift V_{th} by $\Delta V^{R}_{th,S}$ (i.e., the right shift amount of a cell's V_{th} in state S). After m PISO operations, the total threshold voltage right shift amount for a state S is $m\Delta V^{R}_{th,S}$. Thus, the voltage distribution model shown in Equation (1) can be modified as below:

$$f(x)=\sum_{s=0}^{3}\frac{1}{4\sqrt{2\pi}\delta_s}exp\{\frac{-[x+m\Delta V^{R}_{th,S}-(1-\alpha(t))\mu_s]^2}{2\delta_s^2}\}. \tag{4}$$

The tail probability function of each state is used to compute errors existing in this distribution model [4]. Hence, the error minimization problem can be expressed as follows:

$$\begin{aligned}&min[\frac{1}{4}Q_{S_0}(\frac{|\Delta_0|}{\delta_0})+\frac{1}{4}Q_{S_1}(\frac{|\Delta_1|}{\delta_1})+\frac{1}{4}Q_{S_1}(\frac{|\Delta_2|}{\delta_1})\\ &+\frac{1}{4}Q_{S_2}(\frac{|\Delta_3|}{\delta_2})+\frac{1}{4}Q_{S_2}(\frac{|\Delta_4|}{\delta_2})+\frac{1}{4}Q_{S_3}(\frac{|\Delta_5|}{\delta_3})],\end{aligned} \tag{5}$$

where Δ_s is the distance between the mean V_{th} value of state S and a reference voltage (see Fig. 6). $Q_s(x)$ is the tail probability function of state S:

$$Q_s(x)=\frac{1}{\sqrt{2\pi}}\int_x^{\infty}\exp(-\frac{t^2}{2})dt\ . \tag{6}$$

For example, in state S_1, $\Delta_1=V_1^{read}-m\Delta V^{R}_{th,S}-(1-\alpha(t))\mu_1$. By solving this problem, we can obtain an optimal number of PISO operations m.

In practice, the $\Delta V^{R}_{th,S}$ and $\alpha(t)$ of a flash memory chip vary from vender to vender. Hence, an optimal m for a particular flash memory can be empirically measured in a lab by a manufacturer, and then, can be provided to users.

TABLE II: Parameters of the two flash chips.

	Flash A	Flash B
Page size	16 KB	16 KB
Pages per block	512	256
Blocks per plane	2,048	2,048
Plane per die	2	1
Dies per package	4	2
Read latency (μs)	47	47
LSB page write latency (μs)	471	566
MSB page write latency (μs)	1,353	1,870

Group	P/Es
A	1 K
B	2 K
C	4 K
D	6 K
E	12 K
F	20 K

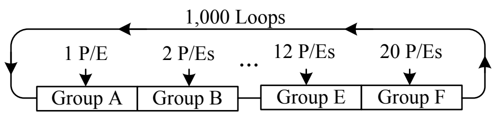


Fig. 7: Variable relaxation cycling procedure.

V. Evaluation

A. Experimental Setting

We use a TRIAD TCII-1600IC NAND flash memory tester [12] to perform P/E cycling and PISO operations as well as to collect error numbers. The tester has 16 DUTs, which can execute I/O commands to 16 flash chips simultaneously. We study two types of flash from two venders called Flash A and Flash B, respectively. We use 16 chips of each type. They are all cMLC flash chips manufactured in 1Y-nm technology. Table II shows their detailed parameters. The read and programming latency are directly measured on the tester. The latency values shown in the table are the mean value of all measured pages. The chips are specified to survive 3K P/E cycles with 10-bit ECC capacity per 512 bytes.

B. Testing Methodology

All chips are experienced *Variable Relaxation Aging* and *Retention Acceleration* before testing. The goal of applying the variable relaxation aging is to examine the effectiveness of PISO on chips with different P/E cycles. In our experiments, all chips are supposed to be used in a 3 years @ 45°C environment, which means a flash memory will be cycled to a certain number of P/E cycles in a 3-year long period and the operating temperature is 45°C. Before applying the aging method, a read bad block operation is performed on all flash memories to identify their bad blocks. All bad blocks are recorded and will not be used in our experiments. For each plane in a chip, 24 blocks are randomly chosen. In order to minimize the programming interference between blocks, choosing the 24 blocks follows two restrictions: (1) the 24 blocks should be spread across an entire chip as widely as possible; (2) the number of blocks between two adjacent testing blocks should be larger than 10. These 24 blocks are separated into 6 groups with each being cycled to a particular P/Es (see the table in Fig. 7). The cycling procedure used in the variable relaxation aging method is illustrated in Fig. 7. The entire procedure consists of 1,000 loops. In each loop, different P/E cycles are performed on different groups according to their

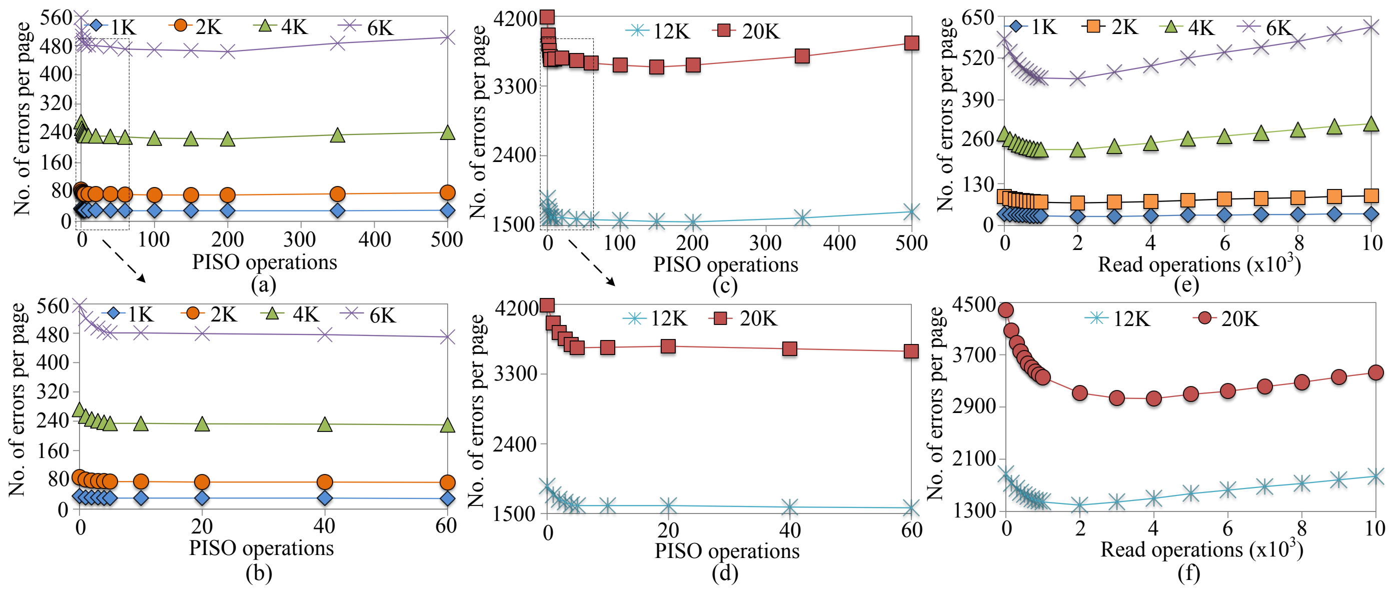


Fig. 8: Number of errors under (a), (b) PISO on small cycles; (c), (d) PISO on large cycles; (e), (f) read operations.

expected P/E cycles. As a result, the total number of P/E cycles is equal to the number designed. Clearly, the relaxation time of each group varies. Group A receives the longest relaxation time, whereas group F has the shortest one. Pseudo-random data are programmed to all chips. After cycling, we apply an endurance bake for all the chips to simulate a 3 years @ 45°C operating environment. According to the Arrhenius equation ($1.1eV$ activation energy is used) [6], all chips are baked under 100°C for 70.6 hours.

The retention acceleration is used to simulate a long period of retention time in a laboratory environment. We assume that all chips are stored under 40°C. Thus, in order to simulate a 3-month retention time, chips are baked for 63 hours under 70°C ($1.1eV$ activation energy is used in the Arrhenius equation). Two types of retention baking schemes are used in our experiments: (1) long-term baking scheme (LB) applies 63-hour baking time in a single baking process, which directly accelerates flash chips to a 3-month retention; PISO operations are carried out after the baking process. (2) step-baking scheme (SB) divides the entire 63 hours into 3 equal sections so that each 21-hour section simulates a 1-month retention. The time interval between two adjacent sections is set to 1 hour, within which a group of PISO experiments are performed.

It is understood that the number of retention errors varies under different temperatures. In this research, we assume that all flash memory devices are used in a temperature-controlled environment like a data center. Thus, the impact of temperature on the PISO approach is not considered.

C. Evaluating the Effectiveness of PISO

Fig. 8a to 8d show the average number of bit errors under PISO on Flash A under the LB scheme, whereas Fig. 8e and 8f show that under the read disturb scheme. From the figures we can clearly see that the number of bit errors on all blocks rapidly decreases within the first 10 PISO operations. On average, the number of bit errors is reduced by 13.7% after the first 5 PISO operations. As the number of PISO operations increases, the number of bit errors on all blocks continuously decreases but in a lower rate. Among all cases shown in Fig. 8, PISO delivers its best performance when it reduces bit errors by 21.9% on 4K-cycled blocks. The rapid decrease in retention errors within the first few PISO operations is due to the fact that when the width of a cell transistor is below 100-nm the generation of interface traps increases rapidly by a positive stress (i.e., PISO operation) on a memory cell [7]. The newly generated interface traps have the same effect as injected electrons, which push drifted V_{th} back to its original voltage range. When more PISO operations are applied, the total number of interface traps in the tunnel oxide of a memory cell reaches its saturation point. After that, injecting electrons becomes the major factor to correct retention errors. Therefore, the V_{th} change becomes moderate. After a particular point (e.g., 200 PISOs on 20K-cycled blocks in Fig. 8c), further increasing the number of PISO operations enlarges the number of bit errors. The error increasing rate in different blocks varies. The blocks cycled to 20K experience the most significant error increase, while blocks cycled to 1K have the lowest error increasing rate. The number of errors in 20K-cycled blocks and in 1K-cycled blocks increase by 8% and 3.3%, respectively. The reason behind this is that electrons are more prone to escape from and to be injected into blocks with a larger P/E cycles due to a weakened tunnel oxide.

D. Comparing Cost with Read Disturb

The PISO approach and the read disturb scheme can be categorized into one camp because of the following reasons: (1) they reduce retention errors by injecting electrons; (2) they do not require a prior knowledge of the original data; (3) they do not consume extra P/E cycles. Therefore, we compare the cost of the two schemes in this section.

Fig. 8e and 8f show the retention error reduction capability of the read disturb scheme. Read disturb demands a

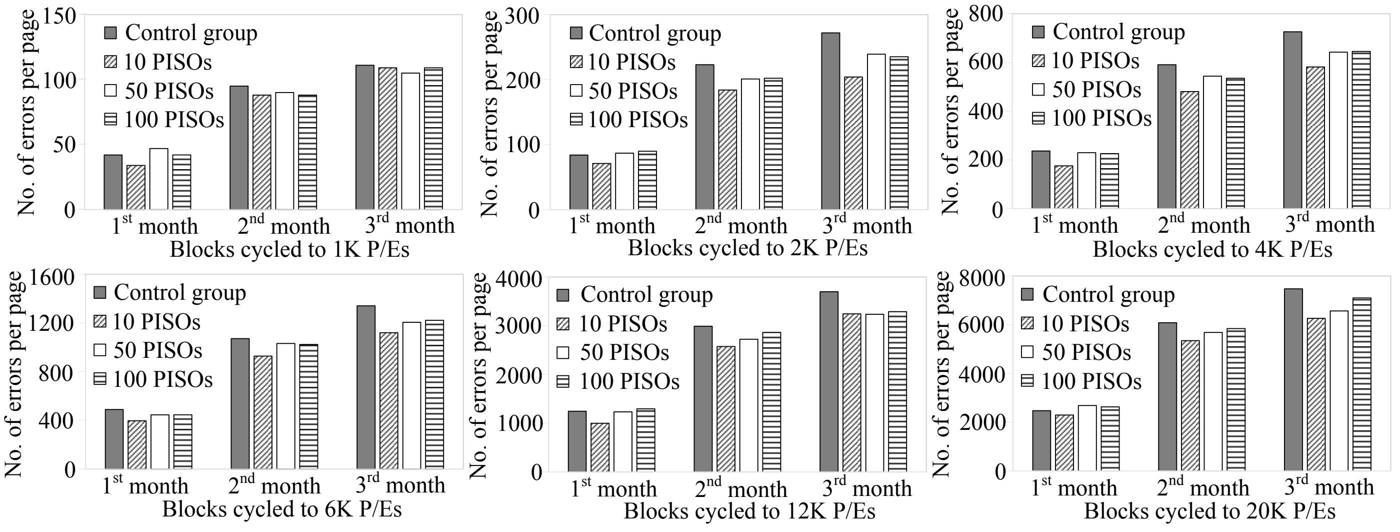


Fig. 9: The impact of the number of PISO operations under different P/E cycles.

much larger number of read operations in order to reduce a similar number of retention errors that PISO can achieve. For example[1], 5 PISO operations can reduce 17% of errors on 6K-cycled blocks (see Figure 8c), whereas roughly 700 read operations are needed to reduce the same amount of retention errors by the read disturb scheme (see Figure 8e). According to the read and write latencies shown in Table II, performing 5 PISO operations costs $5\times1,353\mu s = 6.8ms$, whereas applying 700 read operations costs $700 \times 47\mu s = 32.9ms$. Therefore, the PISO approach is much faster than the read disturb scheme in order to reduce the same amount of bit errors.

Typically, a read operation consumes $1 \sim 2\mu J$ energy, whereas a programming operation costs $15 \sim 30\mu J$ energy [13]. Therefore, reducing 17% of errors on 6K-cycled blocks consumes at least $700\mu J$ energy (700 operations $\times$ $1\mu J$) by the read disturb scheme. PISO, however, at most costs $150\mu J$ energy (5 operations $\times$ $30\mu J$). Furthermore, as a PISO operation only performs the first programming-and-verifying step, the total energy consumption is much lower than $150\mu J$. Thus, PISO is superior to the read disturb scheme in both operation time and energy consumption.

E. Impact of the Number of PISO Operations

In this section, we use an experiment to illustrate the impact of the number of PISO operations on error reduction. In our experiments, the PISO operations are directly performed by a TRIAD tester on raw flash chips. Unlike real application environments, we use retention bake to accelerate the retention time in a laboratory environment.

Fig. 9 exhibits the retention errors on Flash B under an SB scheme. Sixteen chips are evenly divided into four groups with each performing a particular number of PISO operations (see Fig. 9). One group is served as a control group on which no operations is performed. The other three groups receive different numbers of PISO operations after every retention bake period. It is obvious that applying PISO operations 10 times each month can reduce the largest number of retention errors among the four groups. On average, the retention errors can be reduced by 21.5% if 10 PISO operations are applied each month. Further, we can see that if applying more than 10 PISO operations (i.e., 50 and 100) each month, the number of errors begins to increase. The reason is that the over-programmed errors introduced by PISO (see Figure 6d) outweighs the number of its reduced retention errors. From Fig. 9 we can see that there is no noticeable difference in terms of retention errors between 50 PISOs and 100 PISOs in most cases. This is because the number of retention errors further reduced by the extra 50 PISOs is almost equal to their introduced over-programmed errors. When the number of P/E cycles is small (i.e., flash is at the beginning for its life), the impact of PISO on retention reduction is limited (see the blocks cycled to 1K P/Es). Furthermore, the differences of bit errors among the four groups are also small. This is because electrons are difficult to leak from or inject into the floating gate of a less damaged memory cell. Therefore, performing PISO operations on a new flash is neither necessary nor helpful.

Fig. 8 and 9 demonstrate that the PISO approach noticeably reduces retention errors on both Flash A and B.

VI. Applying PISO on Real Applications

The PISO approach can be simply implemented in either the FTL of an SSD or in a flash file system. Still, there are multiple issues on how to efficiently apply PISO operations on real applications. In this section, we provide discussions on these issues.

Discussion 1: When is the best time to launch PISO operations? In this group of experiments, the PISO operations are simply launched every month. For real applications, however, the timing to invoke a PISO operation has to be carefully chosen as various flash memory devices may incur different numbers of retention errors over the same amount of retention time. In addition, it is clear that retention error does not exist in a piece of newly written data. Therefore, performing

[1]Both PISO and the read disturb scheme are carried out on all the pages of a certain block. We choose one page as an example to compare their costs.

PISO operations on the blocks with newly written data is unnecessary.

Discussion 2: What is the best time interval between two adjacent PISO operations? As stated in Section IV, the distance between V_{th0} and V_{th3} shrinks when retention time increases. If the interval time is too long, the change of V_{th} distribution will be too large to be pushed back to its initial state. In this case, PISO has little impact on correcting retention errors. For distinct types of flash chips, the effects of a PISO operation vary due to manufacturing processes and materials. Hence, an optimal interval time should be carefully examined for a particular flash type.

Discussion 3: How many PISO operations are needed in one error correcting process? An appropriate number of PISO operations to be performed demands a comprehensive consideration. A practical solution to the above two issues is to establish a dynamic retention error detection mechanism, which periodically samples retention errors. When the number of retention errors exceeds a pre-defined threshold, it launches a PSIO operation. After a few number of PISO operations (e.g., 5), the mechanism measures the current number of retention errors to decide whether to stop or to carry out more PISO operations. To reduce the performance degradation caused by the mechanism, the priority of PISO operations is configured to a level lower than normal operations. Obviously, an SSD can launch PISO operations whenever it is idle.

A PISO operation can introduce over-programmed errors, which may be accumulated over time. The over-programmed errors neutralize the effect of a PISO operation. As a result, the maximum number of reduced errors by PISO decreases. When the minimum number of errors after PISO operations is close to the ECC capacity of a flash memory, any further PISO operations may damage the stored data as the ECC can no longer correct all the errors. In this case, data have to be remapped to another location so that all accumulated errors are eliminated. The remapping process is combined with the dynamic retention error detection mechanism.

VII. Conclusions

We propose a novel retention error reduction approach called PISO (programming initial step only). It injects electrons to compensate charge loss over time by programming data that represents a memory cell's lowest threshold voltage to a programmed cell. Experimental results from two types of 1y-nm NAND flash memory prove the effectiveness of PISO and its low-cost compared with read disturb [10]. Moreover, an analytical model is built to guide us to discover an appropriate number of PISO operations on a particular flash memory device. The PISO approach has several advantages over the existing retention error reduction schemes discussed in Section II. First of all, a PISO operation is more efficient than a read operation in terms of compensating a cell's charge loss over time for it uses a higher programming voltage to inject electrons into it. Experimental results demonstrate that 5 PISO operations can reduce the same amount of retention errors on a 6K-cycled block as 700 read operations [10]. Secondly, the over-programmed issue can be well controlled as the number of PISO operations can be easily modulated. On the contrary, the FCR technique [8] only has one programming process, and thus, has no opportunity to manage the over-programmed issue. Finally, the cost of a PISO operation is low because it performs neither a prior read operation nor an ECC algorithm.

In current study, we only examine the effectiveness of PISO on raw flash chips. In future work, we will integrate it into a flash storage product like an SSD to further evaluate its efficiency in real-world applications. Furthermore, the impact of the dynamic retention error detection mechanism will be comprehensively studied.

Acknowledgments

We would like to thank the SSD media group in Seagate for their support on this work. We also would like to thank the anonymous reviewers for their constructive comments that improve this paper. This work is sponsored in part by the U.S. National Science Foundation under grant CNS-1320738.

References

[1] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489–502, 2003.

[2] J. He, A. Jagatheesan, S. Gupta, J. Bennett, and A. Snavely, "Dash: a recipe for a flash-based data intensive supercomputer," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2010, pp. 1–11.

[3] E. Deal, "Trends of nand flash memory error correction," *Cyclic Design*, June 2009.

[4] W. Wang, T. Xie, and D. Zhou, "Understanding the impact of threshold voltage on mlc flash memory performance and reliability," in *Proceedings of the 28th ACM international conference on Supercomputing (ICS'14)*, 2014.

[5] N. Mielke, H. Belgal, I. Kalastirsky, P. Kalavade, A. Kurtz, Q. Meng, N. Righos, and J. Wu, "Flash eeprom threshold instabilities due to charge trapping during program/erase cycling," *Device and Materials Reliability, IEEE Transactions on*, vol. 4, no. 3, pp. 335–344, 2004.

[6] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in mlc nand flash memory: Measurement, characterization, and analysis," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, March 2012, pp. 521–526.

[7] J.-D. Lee, J.-H. Choi, D. Park, and K. Kim, "Degradation of tunnel oxide by fn current stress and its effects on data retention characteristics of 90 nm nand flash memory cells," in *Reliability Physics Symposium Proceedings, 2003. 41st Annual. 2003 IEEE International*. IEEE, 2003, pp. 497–501.

[8] 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 *Computer Design (ICCD), 2012 IEEE 30th International Conference on*. IEEE, Sep 2012, pp. 94–101.

[9] F. Sala, R. Gabrys, and L. Dolecek, "Dynamic threshold schemes for multi-level non-volatile memories," *Communications, IEEE Transactions on*, vol. 61, no. 7, pp. 2624–2634, 2013.

[10] S. Tanakamaru, Y. Yanagihara, and K. Takeuchi, "Highly reliable solid-state drives (ssds) with error-prediction ldpc (ep-ldpc) architecture and error-recovery scheme," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*. IEEE, 2013, pp. 83–84.

[11] H. Ma, H. Zou, L. Pan, and J. Xu, "Mlc nand flash retention error recovery scheme through word line program disturbance," in *Next-Generation Electronics (ISNE), 2014 International Symposium on*. IEEE, 2014, pp. 1–2.

[12] TRIAD. TCII Testing Systems. [Online]. Available: http://www.triadspectrum.com/tcii1600ic.html

[13] V. Mohan, T. Bunker, L. Grupp, S. Gurumurthi, M. R. Stan, and S. Swanson, "Modeling power consumption of nand flash memories using flashpower," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 7, pp. 1031–1044, 2013.