Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation

Yu Cai[1], Onur Mutlu[1], Erich F. Haratsch[2] and Ken Mai[1]
1. Data Storage Systems Center, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA
2. LSI Corporation, San Jose, CA
yucaicai@gmail.com, {omutlu, kenmai}@andrew.cmu.edu

***Abstract*— As NAND flash memory continues to scale down to smaller process technology nodes, its reliability and endurance are degrading. One important source of reduced reliability is the phenomenon of *program interference*: when a flash cell is programmed to a value, the programming operation affects the threshold voltage of not only that cell, but also the other cells surrounding it. This interference potentially causes a surrounding cell to *move to* a logical state (i.e., a threshold voltage range) that is different from its original state, leading to an error when the cell is read. Understanding, characterizing, and modeling of program interference, i.e., how much the threshold voltage of a cell shifts when another cell is programmed, can enable the design of mechanisms that can effectively and efficiently predict and/or tolerate such errors.**

In this paper, we provide the first experimental characterization of and a realistic model for program interference in modern MLC NAND flash memory. To this end, we utilize the *read-retry* mechanism present in some state-of-the-art 2Y-nm (i.e., 20-24nm) flash chips to measure the changes in threshold voltage distributions of cells when a particular cell is programmed. Our results show that the amount of program interference received by a cell depends on 1) the location of the programmed cells, 2) the order in which cells are programmed, and 3) the data values of the cell that is being programmed as well as the cells surrounding it. Based on our experimental characterization, we develop a new model that predicts the amount of program interference as a function of threshold voltage values and changes in neighboring cells. We devise and evaluate one application of this model that adjusts the read reference voltage to the predicted threshold voltage distribution with the goal of minimizing erroneous reads. Our analysis shows that this new technique can reduce the raw flash bit error rate by 64% and thereby improve flash lifetime by 30%. We hope that the understanding and models developed in this paper lead to other error tolerance mechanisms for future flash memories.

Keywords—NAND flash, program interference, reliability, read retry, error correction, error model

I. Introduction

Continued decrease in per-bit cost of NAND flash memory has enabled it to be widely adopted as a low-latency storage medium in a wide variety of systems ranging from mobile devices to enterprise servers. This scaling has been enabled due to continued reductions in the feature size of flash memory cells as well as the use of the multi-level cell (MLC) technology. Unfortunately, as feature size reduces (beyond 20nm today) and MLC technology is applied more aggressively, NAND flash memory cells become increasingly more subject to circuit level noise, leading to reduced reliability and endurance [15][16]. As a result, more sophisticated mechanisms to tolerate (i.e., prevent and/or correct) errors become increasingly necessary. To design effective and efficient error tolerance mechanisms, a strong understanding of the causes of errors and the mechanisms that lead to the manifestation of different error causes is necessary. This paper builds such an understanding by providing the first characterization and modeling of an important class of NAND flash memory errors, called *program interference errors.*

Past works characterized flash memory errors into four types: erase, program interference, retention, and read [15][16]. Retention errors, which occur due to flash cells gradually losing charge over time, were shown to be the dominant cause of errors in state-of-the-art MLC NAND flash memories, whereas *program interference errors,* which occur when the data stored in a cell/page changes unintentionally while a neighboring page is programmed, were shown to be the second dominant cause [15][16]. Building upon this understanding, we developed techniques to reduce retention errors by adaptively refreshing and correcting flash cells [12], showing that retention errors can be effectively mitigated. Unfortunately, a detailed characterization and analysis of the second most dominant cause of errors, *program interference*, along with mitigation mechanisms for it, has not been provided in previous works. Our goal in this paper is to build a detailed understanding of the phenomenon of program interference in state-of-the-art MLC NAND flash memory via experimental characterization of existing commercial 2Y-nm (i.e., 20-24nm) flash memory chips.

In MLC NAND flash memory, the logical value stored in a memory cell is determined by the threshold voltage range into which the cell's actual threshold voltage falls [8]. Program interference is the phenomenon in which the threshold voltage of a flash cell, called the *victim cell*, *unintentionally* changes (i.e., gets *disturbed*) while another cell's value is being programmed [3][4][15][16]. This is due to parasitic capacitance coupling between neighboring cells, which gets worse as cells become closer to each other in distance with the reduction in feature size. If the change in threshold voltage due to program interference causes the victim cell's voltage to shift to a different threshold voltage range, then the victim cell's value becomes incorrect, leading to an error when the cell is read.

In this work, we first experimentally characterize how program interference changes the threshold voltage distribution of neighboring cells under a wide variety of programming conditions. In particular, we analyze the effects of the locations of programmed and victim cells, programming order of cells, the number of P/E (program/erase) cycles endured so far by the victim cell, and the values stored in the victim cell and its neighbors. Using the results of this characterization, we next build a model that can predict the change in the threshold voltage of a victim cell due to program interference caused by the programming of another cell. Unlike previous models developed in the past for this purpose [3][4], our model does not require the knowledge of cell-to-cell parasitic coupling capacitances, which are unknown to the designers of the flash controllers and which could vary from cell to cell. As a result, our model is more realistic to be implemented in a flash controller. Finally, we use our predictive model to develop a new technique to adjust the *read reference voltage* (i.e., the voltage level used to distinguish different threshold voltage ranges) to the predicted threshold voltage distribution such that the logical value read from a cell that is affected by program interference is more likely to be the correct original value stored in the cell.

This paper is the first to make the following contributions and observations:

1) We provide detailed rigorous experimental characterization and analyses of the program interference phenomenon in modern 2Y-nm flash memory chips. Our characterization shows that how much a cell is affected by the program interference it receives depends on three factors: i) the location of the programmed cells (wordline to wordline interference is much greater than bitline to bitline), ii) the order in

which cells are programmed (out-of-order programming of flash pages leads to much more interference than in-order programming), and iii) the data values of the cell that is being programmed as well as the cells surrounding it. We find that the amount of P/E cycles a cell has endured is not a significant factor that determines the program interference it receives.

2) We show that program interference can be accurately modeled as additive noise following Gaussian-mixture distributions. We build a new model for predicting the threshold voltage change in a cell due to program interference from surrounding cells. This linear regression based model is realistically implementable in the flash controller, unlike past works, because it does *not* require knowledge of coupling capacitance values between cells. We find that this model can predict the threshold voltage change of a victim cell with 96.8% accuracy.

3) We introduce a new error tolerance mechanism that can adjust the read reference voltage dynamically based on the predictive threshold voltage model. The key observation is that the threshold voltage distribution of a cell shifts to the right and widens with program interference *in a manner that is predictable via online learning of the distribution.* As a result, the optimum read reference voltage between different threshold voltage ranges (i.e., logical values) can be predicted. Our evaluations show that doing so with our online mechanism reduces the raw bit error rate (BER) by 64% and improves the P/E cycle lifetime of NAND flash memory by 30% compared to a baseline that uses state-of-the-art error-correcting codes (ECC).

II. Background and Related Work

A. All Bit Line (ABL) Flash

A NAND flash memory chip is divided into thousands of two-dimensional arrays of flash cells, called blocks. Within a block, all the cells in the same row share a wordline. A wordline typically spans 32k to 64k cells. Our experimental flash designs use an all-bit-line (ABL) architecture wherein all the cells on the same wordline are read and programmed as a group [11] to achieve high performance as more cells can be programmed or read simultaneously. In a 2-bit MLC ABL design, all the least significant bits (LSB) on a wordline form what is called *the LSB page* and all the most significant bits (MSB) form the *MSB page*. Each page is assigned a unique page number. The LSB and MSB page numbers on wordline n are (2n-1) and (2n+2) respectively, as shown in the example in Fig. 1(a) (Wordline 0, where LSB and MSB page numbers are 0 and 2, is the only exception to this rule). Flash memory is programmed page by page: the MSB page of a wordline is programmed at a different time from the LSB page of the same wordline. In general, flash memory manufacturers recommend that the pages inside a flash block be programmed sequentially in *page number order* (0, 1, 2, ...). We will explain the order of programming in Section III as it affects program interference.

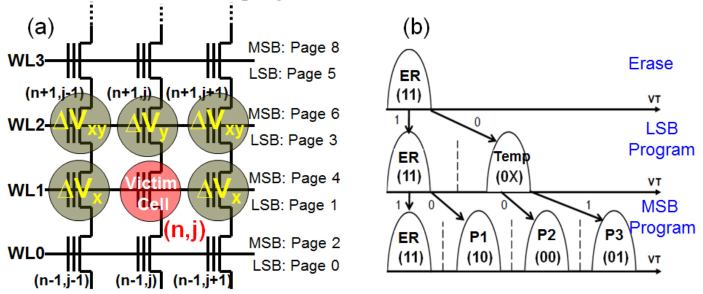

Fig. 1. (a) All-bit-line NAND flash block architecture with victim cell circled in red and aggressor cells circled in grey; (b) Two-bit MLC flash programming scheme. Cell states are encoded in format (LSB, MSB)

A contemporary two-bit-per-cell MLC flash cell is programmed to a desired value in two stages, as shown in Fig. 1(b). After an erase operation (applied to the page the cell resides in), the cell starts out at the erased state (ER). If the LSB of the cell is programmed as 0 (during the programming of the corresponding LSB page), the flash cell moves into a temporary program state (Temp). Otherwise it remains in the ER state. During the programming of the corresponding MSB page, if bit value 1 is programmed into the cell's MSB, the flash cell either remains in the ER state or moves from the Temp state into the P2 state. If 0 is programmed into the cell's MSB, the flash cell moves either from the ER state to the P1 state or from the Temp state to the P3 state.

B. Program Interference and Related Work on Modeling It

In MLC NAND flash memory, the logical value stored in a memory cell is determined by the threshold voltage range into which the cell's actual threshold voltage falls. MLC NAND flash chips use incremental step pulse programming (ISPP) [2] to set the state of the cell's threshold voltage to a value within the voltage range corresponding to the logical value to be stored. However, due to coupling capacitance between neighboring floating gates, the programmed threshold voltage of a cell may change when neighbor cells are programmed later. Program interference is the phenomenon in which the threshold voltage of a flash cell, called the *victim cell*, *unintentionally* changes (i.e., gets *disturbed*) while another cell, called the *aggressor cell,* is being programmed [3][4]. If the change in threshold voltage due to program interference causes the victim cell's voltage to shift to a different threshold voltage range, then the victim cell's value becomes incorrect, leading to an error when the cell is read. An example victim cell and its aggressor cells are shown in Fig 1(a).

C. Previous Work on Modeling Program Interference

In previous work [3][4], the threshold voltage change of the victim cell is modeled as a function of the coupling capacitance and threshold voltage changes of the aggressor cells:

$$\Delta V_{victim} = (2C_x \Delta V_x + C_y \Delta V_y + 2C_{xy} \Delta V_{xy}) / C_{total} \qquad (1)$$

where ΔV_{victim} is the estimated threshold voltage change of the victim cell. C_{total} is the total capacitance of the victim floating gate, C_x, C_y and C_{xy} are the coupling capacitances between the victim cell and its neighbors in the horizontal, vertical and diagonal directions, respectively. ΔV_x, ΔV_y and ΔV_{xy} are the threshold voltage changes of the aggressor cells in the horizontal, vertical and diagonal directions, respectively.

This model assumes linear correlation between the program interference induced threshold voltage change of the victim cell and the threshold voltage changes of the aggressor cells. Unfortunately, the model has a fundamental shortcoming: the coupling capacitance and total capacitance of each flash cell are process and design dependent, and can be affected by the systematic and random variations of the manufacturing process. Thus, the exact values for coupling capacitance are extremely difficult to determine, even by the flash manufacturers. Due to this, the previously proposed model cannot realistically be used by a flash controller. We will overcome this fundamental shortcoming in the model we develop, as described in Section IV.

Past works on modeling program interference had several other shortcomings, which we overcome in this paper. First, past models were verified only in 120-nm [4] and 60-nm [3] flash memory; we expect program interference to be a much bigger problem at 2Y-nm flash memory we test as the distance between neighboring cells is smaller. Second, these works only considered aggressors that are nearest neighbors to the victim cell. Third, these previous works [3][4] did not show the threshold voltage distribution shifts due to program interference, which we rigorously analyze in this paper. Finally, there are several other works that provided models for program interference [5][6][7], yet their models were based on simulations which in turn were based on intuitive assumptions that were not verified via experimental characterization of *real* flash memory chips. In this work, our goal is to provide a comprehensive and rigorous characterization of program interference based on experimental data from real 2Y-nm flash memory chips.

D. Experimental Testing Platform and Methodology

To characterize threshold voltage changes due to program interference, we use an FPGA based flash testing platform that allows us to issue commands to raw flash chips without ECC [9]. We test 2-bit MLC NAND flash memory devices manufactured in 2Y-nm technology. We use the *read-retry* feature present in these devices to accurately read threshold voltages of cells, as described in [8]. Our previous work describes our characterization platform in detail [8][9].

III. PROGRAM INTERFERENCE CHARACTERIZATION

There are two types of program interference: (1) *bitline to bitline interference*, where the interference is due to the aggressor cells on the neighboring bitlines of the victim cell that are on the *same wordline*; (2) *wordline to wordline interference*, where the interference is due to the aggressor cells in the neighboring wordlines of the victim cell. Section III.A characterizes bitline to bitline interference and shows that it is negligible for ABL flash memory because all the cells in the same wordline are programmed at the same time. The effect of wordline to wordline interference depends on the page programming order. Generally, flash memory manufacturers recommend that the pages inside a flash block be programmed sequentially in page number order. With this programming policy, pages in Fig. 1(a) would be programmed in the following order: page 0 (LSB of WL 0), page 1 (LSB of WL 1), page 2 (MSB of WL 0), page 3 (LSB of WL 2), page 4 (MSB of WL 1), and so on. This is called *in-page-order programming*. On the other hand, a flash block can be programmed without following this recommendation, a method called *out-of-page-order programming*. This increases the flexibility in programming flash. This method can have many variants. For example, pages in a block can be programmed in a completely random order, or the pages in a block can be programmed in their wordline number order. Section III.B and III.C respectively characterize potential program interference caused by both in-order and out-of-order programming.

A. Bitline to Bitline Program Interference

In ABL flash, the cells on the same wordline are programmed *simultaneously*. Thus, it is very difficult to isolate the bitline to bitline interference by comparing the threshold voltage of the victim cell before and after *only* its neighbor aggressor cells are programmed. To evaluate bitline to bitline program interference, we investigate what happens to the threshold voltage distribution of a victim cell when its value X is kept constant over multiple programming cycles, but the values of its left and right neighbors (L and R, respectively) are varied. Fig. 2(a) shows L, X, R pictorially. All other cells in the same wordline are programmed to random values, and neighboring wordlines are not programmed. For each programmed state X (i.e., P1, P2 and P3 states), we record the threshold voltage distribution for different L-R data patterns programmed in the neighboring bitlines (after thousands of programming cycles). Since L and R each can take 4 values (in 2-bit MLC flash), this results in 16 different threshold voltage distributions which are plotted in Fig. 3.

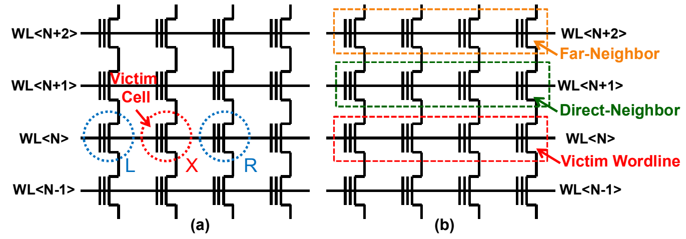

Fig. 2. Terminology used for our program interference characterization. (a) Bitline to bitline interference; (b) Wordline to wordline interference.

Each distribution in Fig. 3 represents the voltage of the victim cell for a different data pattern in its immediate left and right neighbors. The figure shows that the 16 distributions *mostly* overlap. For example, the normalized mean V_{th} values of all 16 distributions for P2 State (00) are between 278.9 and 282.3 and the difference between the maximum and minimum is less than 1.1%. This means that the value of the victim cell does not significantly depend on the values programmed in left and right neighbor cells. Hence, we conclude that the bitline to bitline program interference is small.

We hypothesize that the reason for the lack of bitline to bitline interference is fundamental to all-bit-line flash memory. This is because ISPP (Incremental Step Pulse Programming) [2] used for programming the cells in the entire wordline can be (and is likely) optimized such that bitline to bitline interference that might occur during programming is minimized [3]. For example, flash cells are programmed such that the programming of all cells within the same wordline finishes *at the same time* [3], and the amount of voltage that is applied to each cell in the wordline can be customized such that voltage shifts due to bitline interference are taken into account [3].

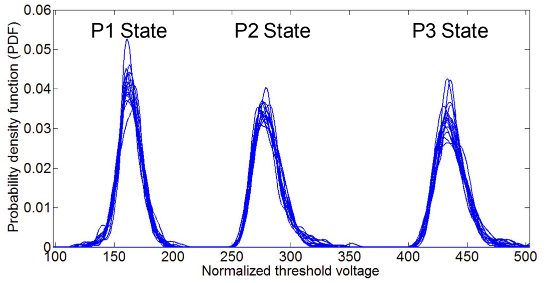

Fig. 3. Victim cell threshold voltage distributions with different values programmed on left and right aggressor cells (16 distributions are plotted).

B. WL to WL Interference with In-Page-Order Programming

When pages are programmed in order (see the beginning of Section III), there are two types of wordline to wordline interference suffered by a victim cell in wordline N: (1) *direct-neighbor*, (2) *far neighbor* interference. *Direct neighbor interference* is the interference caused by programming of data on the MSB page of WL N+1, which is directly above the wordline of the victim cell. Note that the programming of data on the LSB page of N+1 happens *before* the programming of data on the MSB page of WL N, so it cannot cause interference to WL N. *Far neighbor interference* is caused by the programming of data on either the LSB or the MSB page of WL N+2 (and other wordlines with WL number greater than N+2). For example, the threshold voltage of a cell in WL1 can change when the LSB page (page 5) and MSB page (page 8) of WL3 in Fig. 1(a) are programmed. We find that the far neighbor interference caused by wordlines with numbers greater than N+2 (i.e., wordlines that are farther than two wordlines above the victim cell) is negligible, so we do not discuss this further.

Using our testing platform described in Section II-D, we first record the threshold voltage of each cell on the victim wordline (WL N), which gives us the voltage distribution with "*no program interference*". Then, we do three different experiments to measure direct and far neighbor interference: 1) after programming the victim wordline, we program the MSB page of the wordline directly above the victim wordline (WL N+1), after which we read and record the threshold voltage of the victim wordline (this gives us the voltage distribution after "*direct neighbor interference*"), 2) we repeat the first experiment except we program the LSB page of WL N+2 (to measure the distribution after "*far neighbor (LSB) interference*", 3) we repeat the first experiment except we program both the LSB and MSB pages of WL N+2 (to measure the distribution for "*far neighbor (LSB+MSB) interference*". Fig. 4 shows the respective threshold voltage distributions of the victim cells with and without interference. We average the mean shifts of the distributions with interference and the standard deviations of the distributions for all programmed states and plot them in Fig. 5(a) and Fig. 5(b) respectively.

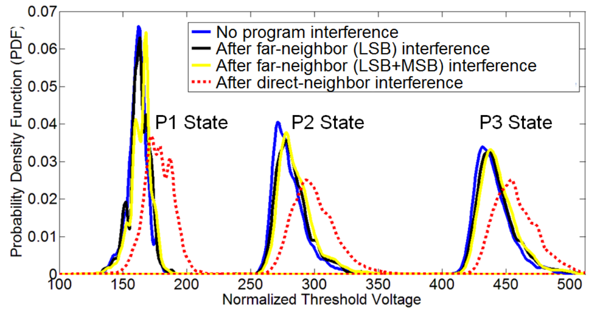

Fig. 4. Victim cell threshold voltage distributions before and after interference (with in-page-order programming).

We make two major observations. First, program interference caused by both near and far neighbor cells can disturb the data stored in victim cells by increasing the threshold voltage of the victim cells. This effect is seen clearly in Fig. 4 and Fig. 5, which show that the threshold voltage distributions of programmed states of the victim cells systematically shift to the right and widen due to program interference from neighboring wordlines. Since flash cells can only be programmed from low threshold voltage to high using ISPP [2] to inject electrons to the floating gates, the threshold voltage changes of the aggressor cells, i.e. ΔVx, ΔVy, ΔVxy values in equation (1), are always non-negative. As the coupling capacitance values between victim and aggressor cells (also in equation (1)) are also positive, program interference can only increase the threshold voltage of the victim cell, which causes its threshold voltage distribution to systematically shift to the right. The distribution also widens with program interference due to the effects of process variation across cells. Due to process variations, both the threshold voltage changes of the aggressor cells as well as the coupling capacitance values across different victim-aggressor cell pairs vary, even if the same data values are programmed across all cells. As a result of this, the effect of program interference on different victim cells is different, leading to the widening of the distributions.

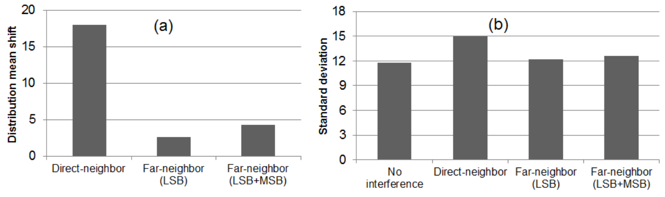

Fig. 5. Victim cell threshold voltage distribution (a) mean value shifts due to program interference (b) standard deviations (with in-order programming).

Second, program interference due to the direct-neighbor wordline is dominant. The voltage distribution mean shift caused by direct-neighbor interference is approximately 4.2 times larger than that caused by far-neighbor (LSB+MSB) interference, as Fig. 5(a) shows. However, program interference due to far-neighbor wordline also exists and is observable in the 2Y-nm flash memory we test. Hence, the amount of program interference correlates with the distance between the victim wordline and the aggressor wordline. This is because the coupling capacitance between the victim and aggressor cells, which causes the program interference, is determined by the distance between the wordlines.

C. *WL to WL Interference with Out-of-Page-Order Programming*

When pages are programmed out of order (see the beginning of Section III), victim cells in a wordline can potentially receive interference due to the programming of *both the LSB and the MSB* of the neighboring wordline above and the neighboring wordline below. Note that with out-of-page-order programming, a victim wordline *can* receive interference from *both* the LSB and MSB page of any of its direct neighbors, whereas this is not possible with in-order programming. As such, the victim wordline can receive interference *once* (after only LSB of the direct-neighbor above is programmed), *twice* (from both LSB and MSB of the direct-neighbor above), *three times* (from all pages of the above direct-neighbor and LSB of the below direct-neighbor), or *four times* (from all pages of both direct-neighbors), depending on how many of its four neighboring pages are programmed. Fig. 6 shows the effect of all four types of interference on the threshold voltage distributions of the victim cells. Fig. 7(a) and Fig. 7(b) summarize the mean shifts and standard deviations of distributions. We make several observations.

First, out-of-page-order programming also leads to significant program interference on the victim cells. The interference caused by the first page programmed in the above direct neighbor (indicated by *once* in the figures) is similar in amount to the interference caused by the above direct neighbor when using in-order programming (as can be seen by comparing the distribution mean shift of "*once"* in Fig. 7(a) to that of "direct-neighbor" in Fig. 5(a)). Second, the mean shift in voltage distribution of victim cells shift correlates linearly with the number of times the victim wordline receives interference. For example, when the victim cells receive interference four times (i.e., after both the LSB and MSB pages on two neighboring wordlines are programmed), the average mean shift is 85, which is about 4.4 times that of when the victim cells receive interference only once (i.e., after only the LSB of the direct-neighbor is programmed). Third, the standard deviation also increases linearly with the number of times of interference. This is because the program interference received by different victim cells varies due to process variations, as we explained in Section III-B.

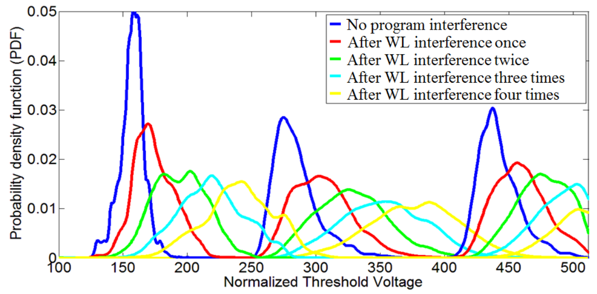

Fig. 6. Victim cell threshold voltage distributions before and after interference (with out-of-page-order programming).

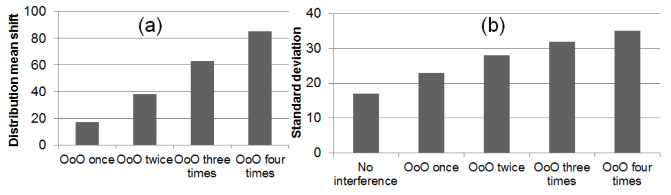

Fig. 7. Victim cell Vth distribution (a) mean value shifts due to program interference (b) standard deviations (with out-of-order programming).

Comparing interference with in-order and out-of-order programming: It is clear from these results that out-of-page-order programming can potentially cause significantly more interference to a victim wordline: with in-order programming, interference is only due to the programming of the MSB page of the direct neighbor wordline, but with out-of-order programming, interference *can be* due to the programming of both the LSB and the MSB pages of the direct neighbor wordlines above and below, leading to approximately 4.4 times as much interference. However, this worst-case may not always be exercised as how much interference a victim wordline receives from its direct neighbors is dictated by the exact programming policy. This interference can vary from *once* to *four times*, as evaluated above.

Note that one example out-of-order programming policy is one in which pages in the same wordline are programmed consecutively and wordlines are programmed in order. In other words, with this programming policy, pages in Fig. 1(a) would be programmed in the following order: page 0 (LSB of WL 0), page 2 (MSB of WL 0), page 1 (LSB of WL 1), page 4 (MSB of WL 1), page 3 (LSB of WL 2), and so on. This policy is called in-wordline-order programing policy, and it causes a victim wordline to receive interference exactly *twice* from its direct-neighbor wordlines.

If we think of the flash cell as storing a signal (i.e., the originally programmed value) and the program interference as adding noise to this signal, we can evaluate the signal to noise ratio (SNR) of different programming policies. The higher the SNR, the better the storage quality of the flash cell and the more likely it will be read correctly (i.e., the less susceptible to error it is). Fig. 8 shows this SNR value for 1) no program interference, 2) in-page-order programming (page order), 3) in-wordline-order programming (wordline order), and other potential versions of out-of-order programming (OoO) which respectively cause interference 1, 2, 3, 4 times from the direct neighbor to the victim (indicated by once, twice, three times, and four times). We make two conclusions. First, in-page-order programming leads to significantly higher SNR than in-wordline order programming. Second, the SNR of the flash cell degrades significantly with more interference it receives from neighboring wordlines. Hence, out-of-page-order programming can greatly degrade the signal quality of the stored value in the flash cell due to program interference. Both of these provide a potential explanation of why flash memory vendors recommend the use of in-order programming, i.e. to maintain signal integrity of data stored in flash cells.

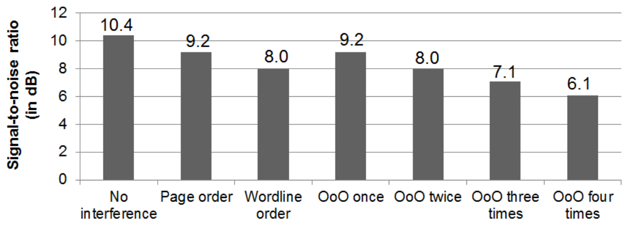

Fig. 8. Signal-to-noise ratio of flash cells under various program orders.

D. *Data Value Dependence of Program Interference*

We investigate the effect of the values stored in victim and aggressor cells on the amount of threshold voltage change of the victim cell caused by program interference. Fig. 9 shows the measured average threshold voltage shift in the victim cell as a function of the direct-above-neighbor aggressor cell and victim cell data values when respectively the LSB (Fig. 9 (a)), the MSB (Fig. 9 (b)), and both the LSB and the MSB (Fig. 9 (c)) programming of the aggressor cell affect the victim. We draw three major conclusions.

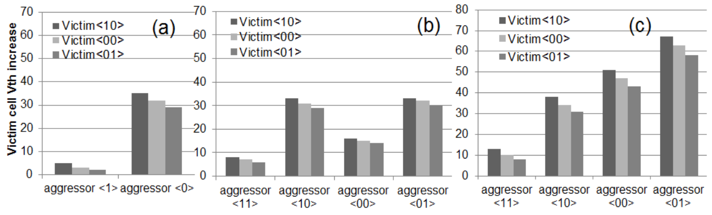

Fig. 9. Average threshold voltage increase in victim cell as a function of victim cell and direct-above-neighbor aggressor cell values when respectively (a) the LSB, (b) the MSB, and (c) both the LSB and the MSB are programmed in the aggressor cell.

First, the amount of program interference received by the victim cell depends greatly on the value programmed into the aggressor cell. This is because different values programmed into the aggressor cell cause different amounts of threshold voltage increase/shift in the aggressor cell, which in turn determines the amount of threshold voltage increase in the victim cell. As shown in Fig. 9 (a), if the aggressor cell's LSB is programmed to 1, then the interference is very low, since the aggressor cell remains in the Erased (ER) state (see Fig. 1(b)), without incurring a large change in its own threshold voltage. In contrast, if the aggressor cell's LSB is programmed to 0 (cell programmed to the intermediate Temp state as in Fig. 1(b)), then the aggressor cell's threshold voltage shifts significantly, causing the victim cell's threshold voltage to also shift due to capacitive coupling between the two cells' floating gates. When the aggressor cell's MSB is programmed, the largest interference is exerted on the victim when the target aggressor value is 10 or 01, as shown in Fig. 9 (b). These two target values again cause the largest threshold voltage increase in the aggressor cell: either from erased state to P1 state or from Temp state to P3 state, as shown in Fig. 1(b). A target value of 00 leads to less interference, as it only causes the state to change from Temp state to P2 (see Fig. 1(b)) state with a smaller threshold voltage change. A target value of 11 leads to the smallest interference as the aggressor cell still remains in erased state (Fig. 1(b)) with very modest threshold voltage change. Similar observations can be made for when both the LSB and the MSB programming of the aggressor cell cause interference to the victim cell (Fig. 9(c)): the target values of the aggressor cell that cause larger threshold shifts in the aggressor cell lead to higher amounts of threshold voltage shifts in the victim cell.

Second, the target values of the aggressor cells that lead to most interference depends on whether or not LSB, MSB, or both LSB and MSB programming of the aggressor cell affect the victim cell. For example the highest interference is caused by target aggressor values of 10 and 01 when only MSB programming affects the victim cell (Fig. 9(b)), but the highest interference is caused by target aggressor values of 00 and 01 when both the LSB and MSB programming of the aggressor affects the victim cell (Fig. 9(c)). A direct result of this observation is that different page programming orders would have different aggressor cell data patterns that would lead to the most interference: with in-page-order programming only the MSB page programming of the direct neighbor affects the victim, leading to the data pattern dependence shown in Fig. 9(b).

Third, the amount of program interference received by the victim cell also depends on the data value in the victim cell. For example, for the same aggressor target data value, the interference in the victim cell is the highest for a victim cell value of 10 and the lowest for a victim cell value of 01. This is because some states of the victim cell are more likely to receive additional electrons when neighbor cells are programmed. If the victim cell is in a state with fewer electrons (e.g. state 10), more electrons are likely to be injected into the victim cell due to program interference (because the effective electric field on the tunnel oxide between the floating gate and the substrate is relatively high in a state with few electrons). If the victim cell is already in a state with more electrons (e.g., state 01), the electrons in the floating gate partly counteract the electric field due to the high programming voltage applied to the aggressor cell, making it more difficult for electrons to be injected into the victim cell.

Fourth, the value to be programmed on the aggressor cell has much more impact on the program interference received by the victim than the value of the victim cell. This is because for modern flash designs (such as ABL 2Y-nm MLC flash), the effects of coupling capacitance between neighboring cells is more dominant than the susceptibility of a programmed state to electron injection.

E. *Summary of Program Interference Characterization*

We make the following major conclusions from the above characterization: (1) Program interference increases the threshold voltage of victim cells and thus causes threshold voltage distributions to shift to the right and become wider; (2) Direct neighbor wordline program interference is the dominant source of interference, while neighbor bitline interference and far-neighbor wordline interference are orders of magnitude lower; (3) The amount of program interference depends on the programming order of pages in a block: in-page-order programming likely causes the least amount of interference and out-of-page-order programming causes much more interference; (4) The amount of program interference depends on the values of both the aggressor cells and the victim cells, with aggressor cell value having a greater affect as it determines the amount of threshold voltage shift on both the aggressor cell and the victim.

IV. MODELING AND PREDICTING PROGRAM INTERFERENCE

A. *Linear Regression Model to Predict Program Interference*

Feature extraction: As we discussed in section III, the amount of program interference on the victim cell correlates with 1) the threshold voltage changes of its aggressor cells programmed after the victim cell, 2) the threshold voltage of the victim cell before interference. Thus, we can extract the *relative weights/coefficients* of the threshold voltage changes of aggressor cells and threshold voltage of the victim cell as features to predict the program interference on the victim cell.

Extended Interference Model: As noted in Sec. II-C, threshold voltage change of the victim cell can be modeled as a linear combination of the threshold voltage changes of aggressor cells. Thus, we can use linear regression with least-square-error estimation to build a model that represents how the program interference vector ΔV depends on the feature matrix X, which have correlations with ΔV on the victim cells. We enhanced the previous linear model [3][4], shown in Equation (1), by considering more features. Our new model includes the threshold voltage changes of the *indirect* adjacent neighbors and the threshold voltage of *the victim cell itself* before program interference (Sec. III.D showed these values affect the amount of program interference). Suppose the victim cell is on the n-th wordline and j-th bitline as in Fig. 1(a). We incorporate all the cells in the region up to K cells to the left and right of the victim cell in the horizontal direction and up to M wordlines adjacent above the victim cell as part of our new linear model, rewritten as:

$$\Delta V_{victim}(n,j)=\sum_{y=j-K}^{j+K}\sum_{x=n+1}^{n+M}\alpha(x,y)\Delta V_{neighbor}(x,y)+\alpha_0 V_{victim}^{before}(n,j) \tag{2}$$

where $\alpha(x,y)$ and α_0 are fitting coefficients, while V_{victim}^{before} is the threshold voltage of the victim cell before interference.

We can combine all these coefficients as a p×1 regression coefficient vector, called α, and p equals to (2K+1)×M to include all the coefficients. The recorded threshold voltage changes in all the victim cells of one wordline forms an n×1 vector Y, where n is the number of cells on one wordline. The features for each victim cell forms a 1×p vector and thus the features of all the victim cells form an n×p feature matrix X. Each row of matrix X corresponds to the features of one victim cell, which is a 1×p vector. Thus the complete learning model considering all the victim cells can be expressed as:

$$Y = X\alpha + \varepsilon \qquad (3)$$

where ε is the noise error to be minimized to improve the prediction accuracy of the model.

We measure Y by recording the threshold voltage changes of all victim cells. Each row of X is formed by recording the threshold voltage changes of up to (2K+1)×M-1 neighbor aggressor cells and the threshold voltage of each victim cell before program interference. We use the linear regression model to learn the coefficient vector α. Note that this vector α can be different for different manufacturers, generations, and even locations in a flash chip.

Model Coefficient Learning and Analysis: The task of learning of coefficients can be formulated as a linear regression problem as program interference approximately follows a linear model. We select LASSO with L1 regularization [14] to learn the coefficients because LASSO can shrink the fitting coefficients, set non-important factors to be zero and retain only the highly relevant features to simplify the model without losing accuracy. In the coefficient learning stage, the goal is to learn the estimate of fitting coefficients α such that the learning errors are as small as possible, expressed as:

$$\underset{\alpha}{\arg\min}\,(\|X \times \alpha - Y\|_2^2 + \lambda\|\alpha\|_1) \qquad (4)$$

where λ is the parameter to control the amount of shrinkage that is applied to the estimates. Using our flash testing data and LASSO learning, the non-zero fitting coefficients are characterized. Fig. 10 provides the learned values of the model coefficients α, based on our experimental data. Several conclusions are in order.

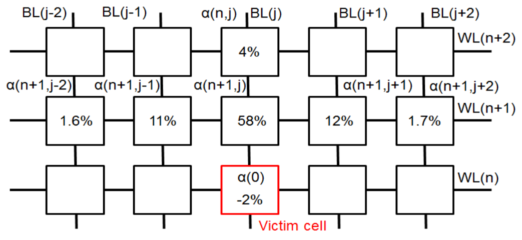

Fig. 10. Model coefficient distribution for equation (2).

First, different features (i.e., neighbor cells) contribute very differently to program interference as there is a wide variance in their coefficients. The physical meaning of α(x,y) in Fig. 10 is the ratio of coupling capacitance between the aggressor cell (x, y) and the victim cell over the total capacitance of the victim cell. We can see that the interference of the top direct neighbor contributes most (58%) to the total program interference. The interference from the cells on the left-diagonal and right-diagonal ranks the second and contributes about 11%, which is 1/5th of the contribution of the top adjacent cell. The cells at the 2-adjacent diagonal contribute almost 1/5th of the contribution of the cells at the 1-adjacent diagonal, which shows that the coupling capacitance decreases exponentially with the distance between the aggressor cell and the victim cell. Thus, the program interference introduced by neighbors, which are two cells away, can likely be neglected for current 2Y-nm NAND flash. Note that our learning techniques can capture those coefficients if their effects may no longer be negligible when flash memory further scales down and the distance between neighbor cells further decreases.

Second, the threshold voltage already programmed on the victim cell has negative impact on the program interference (the coefficient α_0 is negative). The larger the original threshold voltage programmed on the victim cell, the less the interference on the victim cell would be. This is expected as a cell with a large threshold voltage is less likely affected by program interference, as shown in detail in Sec. III-D.

Model Accuracy Evaluation: After learning the parameters of the model in the training stage using measured training data from a set of flash memory cells, we apply this model to the flash cells in other locations in the same flash chip to evaluate the accuracy of our proposed model. Our model would be considered accurate if it correctly predicts the threshold voltage changes in cells due to program interference. Fig. 11(top) shows the threshold voltage of cells before interference (x axis) and after interference (y axis), showing that threshold voltages shift up (as we observed earlier). Fig. 11 (bottom) shows the predicted threshold voltage with our model (y axis) versus the threshold voltage before interference. The predicted threshold voltage is obtained by subtracting the predicted threshold voltage change (estimated by our model) from the measured threshold voltage after program interference.

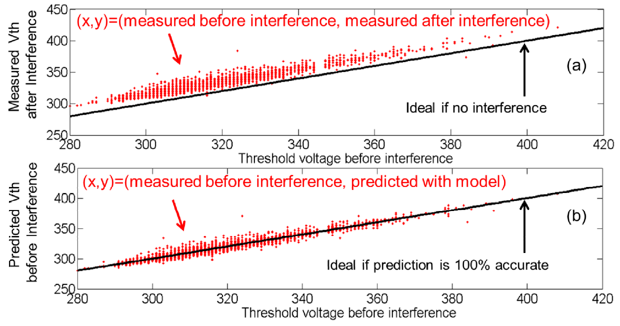

Fig. 11. (a) Measured Vth vs. programmed Vth (b) Vth using our model in Equation (2) vs. programmed Vth. Black curve show the ideal case where measured and predicted Vth should be the same as the programmed Vth.

We make two major observations from Fig. 11. First, if we simply read the threshold voltage of cells measured after interference as an estimate of the programmed threshold voltage of cells, there is an obvious systematic deviation and the accuracy of the measured threshold voltage is only about 83%, as shown in Fig. 11(top). Note that this is the state-of-art in existing literature, where flash cells are directly read without any program interference cancellation techniques. Second, our model in Equation (2) is effective at accurately predicting the threshold voltage of cells by cancelling the effects of program interference, as shown in Fig. 11(bottom). The predicted value fluctuates around the programmed value, and the accuracy of the predicted voltage is 96.8%. We conclude that our model can accurately eliminate the error in threshold voltage measurements caused by program interference.

B. Program Interference Noise Distribution Model

In Sec. III, we observed that the program interference coupling causes the threshold voltage of the victim cells to increase. Thus, we can model the program interference as *additive noise to previously programmed* data as:

$$V_{th}^{(after)} = V_{th}^{(before)} + \Delta V_{pgm}^{noise} + \varepsilon \qquad (5)$$

where $V_{th}^{(before)}$ and $V_{th}^{(after)}$ are the threshold voltages of the victim cells before and after neighbor cell program interference respectively. ΔV_{pgm}^{noise} is the program interference noise, which is systematic and predictable depending on the threshold voltage changes of the neighboring aggressor cells and the original threshold voltage of the victim cell, as described in Section III. ε is random residual noise.

Program interference on the victim cell can be caused by up to m nearest neighbors and each neighbor cell may experience K possible state changes (e.g., from erased state to some programmed state). Without loss of generality, we denote the state change of the i-th neighbor as N_i, which is likely to be a random event that is determined by the data pattern programmed in the neighboring cells. Thus, the m-dimension vector $(N_1, N_2, \ldots, N_m)$ represents the state changes of all the neighbors of the victim cell. There are up to K^m possible random state change event combinations caused by neighboring cells. Thus, the *probability density function (PDF) of the random variable* ΔV_{pgm}^{noise} *of the victim cell* can be expressed as:

$$p(\Delta V_{pgm}^{noise}) = \sum_{i=1}^{K^m} p(\Delta V_{pgm}^{noise}, N_1, N_2, ..., N_m) \tag{6}$$

$$= \sum_{i=1}^{K^m} p(N_1, N_2, ..., N_m) p(\Delta V_{pgm}^{noise} \mid N_1, N_2, ..., N_m)$$

where $p(N_1, N_2, ..., N_m)$ is the expected probability of the particular combination of state changes in neighboring cells and $p(\Delta V_{pgm}^{noise}|N_1, N_2, ..., N_m)$ is the conditional probability distribution of the program interference given a certain combination of neighboring cell state changes. The probability distribution of program interference ΔV can be modeled as a mixture of these conditional probability distributions.

The complexity of the mixture of distributions increases exponentially with the number of neighboring interference cells. However, as we have shown, the most dominant source of program interference is the direct-above-neighbor cell, which is on the same bitline as the victim cell and the adjacent direct-neighbor wordline, i.e., cell (n+1, j) to the victim cell (n, j) as in Fig. 1(a). Thus, the mixture distribution model can be further simplified by considering the interference from *only* this neighbor. This way, the probability density function (PDF) of program interference noise can be simplified from the sum of K^m conditional probability density functions (as in Equation 6) to the sum of only K^1 conditional PDFs. We make this simplification to ease the analysis below.

For two-bit-per-cell MLC, if the program operations are executed strictly following the sequential page order, the direct-above-neighbor aggressor cell may have four possible state changes (i.e., Erased → Erased, Erased → P1, Temp → P2, and Temp → P3, as in Fig. 1(b)). When random data are programmed into the aggressor wordline, each state change is expected to occur with 25% probability. Fig. 12 shows the conditional probability distribution of the program interference noise (i.e., threshold voltage change of the victim cells) for each possible state change. Fig. 12 also shows the combined probability distribution, which amounts to the sum of the four conditional probability distributions. This combined distribution is the distribution one would see if the state change of the neighboring cell is not known. We conclude that the noise caused by program interference is a mixed multi-modal Gaussian distribution, consisting of individual discernible truncated Gaussian distributions, each corresponding to a particular state change in aggressor cells.

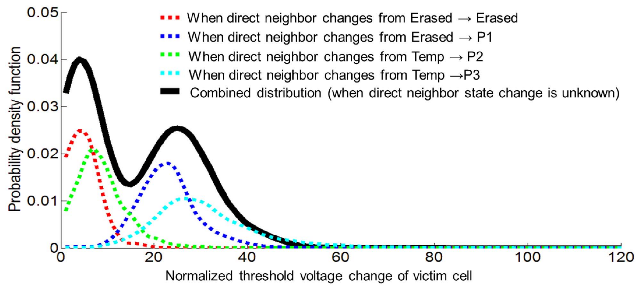

Fig. 12. Program interference noise distribution of one victim wordline when pages are programmed in page order with random data.

C. *Effect of P/E Cycles on Program Interference Noise*

Fig. 13 shows the distribution of program interference noise under different program/erase (P/E) cycles endured by the victim cells. Note that the nominal lifetime of 2Y-nm flash memory is about 3000 P/E cycles and we explore beyond 15 times over this nominal lifetime in our testing. The main observation is that program interference noise experienced by victim cells does not change significantly with P/E cycles (i.e., as flash cells age). This is because program interference is mainly caused by the coupling capacitance between neighboring flash cells. Coupling capacitance is mainly determined by two factors: 1) material properties and 2) distance between neighboring floating gates. Neither of these two factors is affected by P/E cycles. The material used between neighboring floating gates is simply air gap [1] since its relative static permittivity is almost 1 and it can achieve small coupling capacitance to minimize program interference. This material's properties do not significantly change over P/E cycles. The distance between floating gates also do not change over P/E cycles as distance is a function of manufacturing and not use of the circuit. Therefore, we conclude that the amount of program interference does not significantly depend on P/E cycles.

We observe that the program interference noise distribution widens slightly as the number of P/E cycles endured by the victim cells increases. This is because program and erase operations may accumulate defects in the tunnel oxide which sits between floating gate and the substrate, when electrons are tunneling through due to Fowler-Nordheim mechanisms during program and erase operations. Such defects may make it easier for additional electrons to be inadvertently injected into the floating gate of the victim cell when neighboring cells are programmed. However, the data shown in Fig. 13 indicates that the effect of such defects on program interference is small compared to the large effect of coupling capacitance.

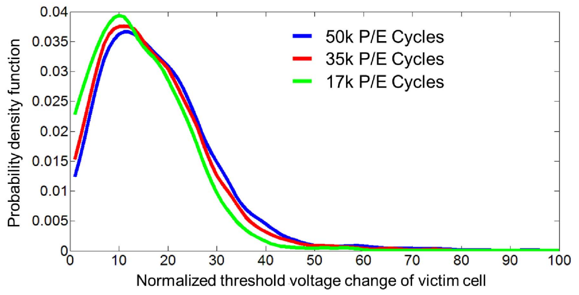

Fig. 13. The probability distribution of program interference noise for different number of P/E cycles endured by the victim cell. (Aggregated over one million victim cells).

D. *Summary of Program Interference Noise Modeling*

We make the following major conclusions from the above modeling studies: 1) Program interference can be modeled as positive and additive noise, which can be predicted by a model that linearly combines threshold voltage changes of aggressor cells and the threshold voltage of the victim cell before program interference. 2) The model coefficients can be learned by the flash controller using LASSO with L1 regularization [14] via online testing. Unlike past models [3][4] this does not require any knowledge of coupling capacitance, which is unknown to the flash controller. 3) Program interference noise follows a multi-modal Gaussian-mixture distribution, which is composed of a few truncated Gaussian distributions, each due to one type of neighbor-cell state change. 4) Program interference noise distribution does not change significantly with P/E cycles. We believe these findings and models can enable new methods to alleviate the effects of program interference noise to increase flash memory reliability. We propose and evaluate one such method next.

V. Mitigation: Read Reference Voltage Prediction

Since we can accurately estimate the threshold voltage shift due program interference (Sec. IV), we can also estimate the threshold voltage distribution of cells after program interference. In this section, we show one example of how to take advantage of this estimation to increase the reliability of reading data from flash memory.

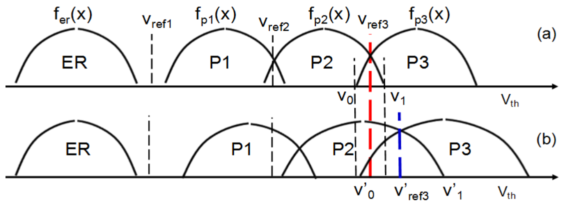

Fig. 14. Illustration of optimum read reference voltage before and after program interference.

Basic Idea: We show an illustration of threshold voltage distributions before and after program interference in Fig. 14(a) and Fig. 14(b) respectively. Here, $f_{p1}(x)$, $f_{p2}(x)$ and $f_{p3}(x)$ represent the threshold voltage distribution probability density function (PDF) of P1, P2 and P3 states, respectively, for 2-bit MLC. V_0 and V'_0 represent the lower limit of the P3 state distribution before and after program interference. V_1 and V'_1 represent the upper limit of P2 state distribution before and after program interference. If we set the read reference voltage between the P2 and P3 states as V_{ref3}, the raw bit error rate for reading cells in the P2 and P3 states are:

$$BER = \int_{V_{ref3}}^{+\infty} f_{p2}(x)dx + \int_{-\infty}^{V_{ref3}} f_{p3}(x)dx \qquad (7)$$

The left summation in equation (7) is the error rate when cells are programmed to P2 but read as P3. The right summation is the error rate when cells are programmed to P3 but read as P2. The minimum raw BER can only be achieved when V_{ref3} is chosen to be at the cross-point of the V_{th} distribution function of P2 and P3 states. However, if the victim cells' neighbor wordline is programmed, the V_{th} distribution of the victim cells will systematically shift to the right and widen, as we have already seen in Section III. Thus, the crossing point of the neighbor threshold voltage distribution also shifts to the right. As in Fig. 14(b), the optimum read reference voltage between P2 and P3 states changes from V_{ref3} to V'_{ref3} after program interference. If we still use V_{ref3} as the read reference voltage, the raw BER will be larger than reading with reference voltage V'_{ref3}. If we choose a read reference voltage between P2 and P3 states in the region of $[V_{ref3}, V'_{ref3}]$, the raw BER will decrease as V_{ref} increases until the optimum point V'_{ref3} is reached. Even if V_{ref} goes beyond V'_{ref3}, there is still a margin δ such that as long as V_{ref} is less than $V'_{ref3} + \delta$, the raw BER is still less than that if we read using the original reference voltage V_{ref3}.

Methodology: To decrease the raw BER, we propose to *predict the optimum read reference voltage between neighboring threshold voltage distribution states*. Our mechanism has two major parts.

The first part of our mechanism learns the program interference noise distribution. We trigger this learning task in the background, i.e. every 1000 full disk P/E cycle operations. Periodically, the learning task selects a victim wordline and programs it with a known data pattern and characterizes the threshold voltage. Then, it programs the neighboring wordlines with random data and characterizes the effect of this programming on the victim wordline. By comparing the threshold voltage before and after interference for the victim wordline, the learning task can learn the distribution of the program interference noise. For simplicity, we record only the mean value of the program interference noise. Recall from Section IV.C that the program interference does not change significantly with P/E cycles. We therefore use the statistics of program interference discovered in the learning stage to predict the program interference during the next period, i.e. the next 1000 P/E cycles. Note that 1000 P/E cycles equals approximately 50 days even if the disk has 20 full disk writes per day under write intensive applications [12]. This learning task can run as low priority and can be interrupted by regular I/O operations to reduce the response time penalty.

The second part of our technique modifies the read reference voltage to be used for a read operation based on the learned model. Before reading from a wordline, the flash controller can check whether a closeby neighboring wordline has been programmed. If not, it uses the default read reference voltage provided by the flash manufacturer to classify the read voltage into a logical value. Otherwise, the flash controller uses our predicted read reference voltage, which is the sum of the default read reference voltage and the predicted amount of shift in the threshold voltage cross-point of two neighboring logical states before and after program interference.

Evaluation Results: Using an FPGA platform [8][9], we tested the raw bit error rate (BER) with and without read reference voltage prediction. Results are shown in Fig. 15 for different P/E cycles. Read reference voltage prediction achieves an average BER reduction of 64% at 3K P/E cycles and the reduction is consistent across all P/E cycle values. One can take advantage of this in two ways: 1) keeping the strength of error correction (ECC) the same, we can have flash memory that has higher P/E cycle lifetime, 2) keeping the P/E cycle lifetime the same, we can use simpler error correction codes, which can be used to achieve the same lifetime. For example, if we apply a 32k-bit BCH code in the flash controller, the acceptable raw BER is 2×10^{-3} [12] and the P/E cycle lifetimes achieved with and without read reference voltage prediction are 26k P/E cycle and 34k P/E cycles, respectively (as shown in Fig. 15). Thus, our read reference voltage prediction techniques can achieve a 30% lifetime improvement. We conclude that read reference voltage prediction, and potentially other mitigation techniques for program interference, can greatly improve flash memory lifetime (by using the model proposed in this paper).

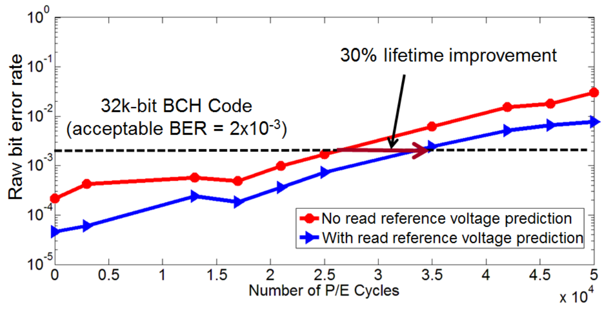

Fig. 15. Raw BER with and without read reference voltage prediction.

VI. CONCLUSIONS

We characterized and analyzed the effects of program interference on cell threshold voltage distributions in state-of-art 2Y-nm NAND flash memory chips. We devised an accurate model that can predict the amount of voltage change due to program interference. Based on this understanding, we developed a mechanism that dynamically determines the read reference voltage that distinguishes the logical states of a multi-level cell. Our evaluations show that this mechanism can significantly reduce raw bit error rate and improve flash memory lifetime because it increases the likelihood that the value of a cell subject to program interference is read correctly. We hope that the characterization, understanding, and models developed in this paper would enable the design of other new and even more effective error tolerance mechanisms, which are critical to develop as flash memory scales down to smaller technology nodes.

ACKNOWLEDGEMENT

This work was partially supported by the Intel Science and Technology Center on Cloud Computing.

REFERENCES

[1] J. Seo et al., "A middle-1X nm NAND flash memory cell (M1X-NAND) with highly manufacturable integration technologies", IEDM 2011.

[2] K.-D. Suh et al., "A 3.3V 32 Mb NAND flash memory with incremental step pulse programming scheme", JSSC 1995.

[3] K. Park et al. "A Zeroing Cell-to-Cell Interference Page Architecture with Temporary LSB Storing and Parallel MSB Program Scheme for MLC NAND Flash Memories", JSSC 2008.

[4] J. Lee et al. "Effects of Floating-Gate Interference on NAND Flash Memory Cell Operation", IEEE EDL 2002.

[5] G. Dong et al. "Using Data Postcompensation and Prediction to Tolerate Cell-to-Cell Interference in MLC NAND Flash Memory", IEEE TCAS- I, 2010.

[6] D. Park et al. "Floating-gate coupling canceller for multi-level cell NAND flash", IEEE Transactions on Magnetics, 2011.

[7] D. Lee et al. "Least squares based cell-to-cell interference cancelation technique for multi-level cell NAND flash memory", ICASSP 2012.

[8] Y. Cai et al. "Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling", DATE 2013.

[9] Y. Cai et al. "FPGA-Based Solid-State Drive Prototyping Platform", FCCM 2011.

[10] T. Hara et al., "A 146 mm2 8 Gb NAND flash memory with 70 nm technology," ISSCC 2005.

[11] R. Cernea et al., "A 32 MB/s MLC Write Throughput 16 Gb NAND with All Bit Line Architecture on 56 nm Technology", JSSC 2009.

[12] Y. Cai et al. "Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime", ICCD 2012.

[13] C. Lee et al. "A 32-Gb MLC NAND flash memory with Vth endurance enhancing in 32nm CMOS", JSSC 2011.

[14] R. Tibshirani, "Regression shrinkage and selection via the LASSO", Technical report, University of Toronto, 1994.

[15] Y. Cai et al., "Error patterns in MLC NAND flash memory: measurement, characterization and analysis", DATE 2012.

[16] N. Mielke et al., "Bit error rate in NAND flash memories", IEEE RPS, 2008.

[17] Y. Cai et al., "Error analysis and retention-aware error management for NAND flash memory", Intel Technology Journal (ITJ), 2013.