Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories

Yu Cai[1], Gulay Yalcin[2], Onur Mutlu[1], Erich F. Haratsch[4],
Osman Unsal[2], Adrian Cristal[2,3], and Ken Mai[1]
[1]Electrical and Computer Engineering Department, Carnegie Mellon University
[2]Barcelona Supercomputing Center, Spain [3]IIIA – CSIC – Spain National Research Council [4]LSI Corporation
yucaicai@gmail.com, {omutlu, kenmai}@ece.cmu.edu, {gulay.yalcin, adrian.cristal, osman.unsal}@bsc.es

ABSTRACT
Continued scaling of NAND flash memory to smaller process technology nodes decreases its reliability, necessitating more sophisticated mechanisms to correctly read stored data values. To distinguish between different potential stored values, conventional techniques to read data from flash memory employ a single set of reference voltage values, which are determined based on the overall threshold voltage distribution of flash cells. Unfortunately, the phenomenon of program interference, in which a cell's threshold voltage unintentionally changes when a neighboring cell is programmed, makes this conventional approach increasingly inaccurate in determining the values of cells.

This paper makes the new empirical observation that identifying the value stored in the immediate-neighbor cell makes it easier to determine the data value stored in the cell that is being read. We provide a detailed statistical and experimental characterization of threshold voltage distribution of flash memory cells *conditional upon* the immediate-neighbor cell values, and show that such conditional distributions can be used to determine a set of read reference voltages that lead to error rates much lower than when a single set of reference voltage values based on the overall distribution are used. Based on our analyses, we propose a new method for correcting errors in a flash memory page, neighbor-cell assisted correction (NAC). The key idea is to re-read a flash memory page that fails error correction codes (ECC) with the set of read reference voltage values corresponding to the conditional threshold voltage distribution assuming a neighbor cell value and use the re-read values to correct the cells that have neighbors with that value. Our simulations show that NAC effectively improves flash memory lifetime by 33% while having no (at nominal lifetime) or very modest (less than 5% at extended lifetime) performance overhead.

Categories and Subject Descriptors
B.3.4 [**Memory Structure**]: Reliability, Testing, and Fault-Tolerance, C.4 [**Performance of Systems**]: Modeling techniques; Reliability

General Terms
Algorithms, Measurement, Performance, Design, Reliability.

Keywords
NAND flash memory, Program Interference, Threshold Voltage Distribution, Error Correction, ECC, Fault Tolerance.

SIGMETRICS'14, June 16–20, 2014, Austin, TX, USA.

1. INTRODUCTION
NAND flash memory is widely used in diverse applications, ranging from mobile electronics to enterprise servers. Such a wide application range is mainly driven by the ever-increasing, low-cost, non-volatile storage capacity provided by NAND flash memory due to aggressive transistor scaling. Unfortunately, as flash cells scale down to smaller technology nodes, they become increasingly vulnerable to circuit level noise, reducing the probability that stored data will be read correctly (even if it were stored correctly at the time it was written) even with the use of aggressive error-correcting codes (ECC) [1][2][3]. As a result, more accurate and sophisticated mechanisms to accurately read and correct the data values stored in flash cells become increasingly necessary. This paper introduces such a new mechanism based on a rigorous experimental analysis of real Multi-Level Cell (MLC) NAND flash memory chips and new findings on how the threshold voltage distribution of flash memory cells can be classified for more accurate identification of the logical data values stored in cells.

In MLC NAND flash memory, the logical value stored in a memory cell is determined by the threshold voltage range (or, window) into which the cell's actual threshold voltage falls [3][4]. As cell size is scaled down and more bits per cell are stored, the threshold voltage range used to represent each logical value becomes smaller, leading to increased error rates in determining a cell's logical value. This is because process variations become more prevalent when the amount of charge stored in a flash cell reduces with feature size, causing the threshold voltages of different cells storing the same value to become significantly different. Hence, deciding what logical value a cell's threshold voltage actually corresponds to is increasingly difficult.

The distribution of threshold voltages across different cells in flash memory is called the "threshold voltage (probability) distribution" [4]. A flash controller uses this distribution across all cells (in a flash memory chip) to determine the "reference threshold voltage values" to distinguish between cells that store different logical values upon reading an address in flash memory. For example, distinguishing between 11, 10, 00, and 01 in a 2-bit flash cell requires three read reference voltages, each one used to distinguish between the adjacent two logical values in terms of threshold voltage ranges. Traditionally, the read reference voltage used to distinguish between two adjacent logical values is a single reference voltage, and this voltage is determined by the manufacturer to minimize the probability that a cell's programmed logical value is misread as another, incorrect value.

Unfortunately, using a single read reference voltage to determine a cell's logical value becomes increasingly difficult due to the phenomenon of program interference [3]. Program interference is the phenomenon where the threshold voltage of a flash cell, called the victim cell, unintentionally changes (that is, gets disturbed) while another, neighboring, cell's value is being programmed

[1][2][3][5][6]. If the change in threshold voltage due to program interference causes the victim cell's voltage to shift to a different threshold voltage range (based on a single read reference voltage), then the victim cell's logical value becomes incorrect, leading to an error when the cell is read. As previous works have shown [3], program interference is responsible for a large fraction of difficult-to-correct errors in flash memory and therefore is a key challenge against scaling flash memory to higher densities.

In this paper, we make the new observation that knowing (or, more broadly, distinguishing) the value programmed in the immediate-neighbor cell makes it easier to determine the value stored in the flash cell that is being read. In other words, one can classify the threshold voltage distribution of flash memory cells into *multiple* threshold voltage distributions based on the logical value of the immediate-neighbor flash cell and use the appropriate *"classified" (or, conditional) voltage distribution* (and the corresponding read reference voltage values associated with it, called the "local optimum read reference voltages") to more accurately determine the value of the cell that is being read. Doing so leads to a more accurate identification of the logical value of the flash cell being read because the classified voltage distributions of the flash cells based on neighbor cell values have smaller overlap between threshold voltage ranges of different logical values than a single overall threshold voltage distribution for all cells. We show both experimentally and analytically this is the case.

Based upon this new observation, we introduce a new error correction method, called neighbor-cell assisted error correction (NAC). With this method, a flash memory page is first read using the regular read reference voltages (and the read logical values are buffered). If there are no errors uncorrectable by ECC, nothing else is done. However, if there are errors uncorrectable by ECC, the flash page is re-read with the read reference voltages corresponding to the voltage distribution assuming a particular immediate-neighbor value. The buffered values of the cells with that particular immediate-neighbor cell value are replaced using this reading, and ECC is applied again. If ECC is successful this time, the buffered page is deemed to be corrected and is supplied to the system. If ECC fails again, the process is repeated with another set of read reference voltages corresponding to the voltage distribution assuming another particular immediate-neighbor value until either ECC passes or all different potential immediate-neighbor values are exhausted. We evaluate NAC and show that it can significantly improve flash memory lifetime with very modest performance loss during the extended lifetime and no performance loss during the nominal lifetime.

To our knowledge, this is the first paper that classifies threshold voltage probability distributions of flash memory cells based on neighboring cell values, analyzes these distributions both statistically and experimentally, and makes use of these classified threshold voltage distributions to improve error correction capability for MLC NAND flash memory. Our major contributions in this paper are as follows:

1. We provide a detailed statistical and experimental characterization of threshold voltage distributions of flash memory cells conditional upon the immediate-neighbor cell values, and show that such conditional distributions can be used to determine read reference voltages that can minimize raw bit error rate (RBER) when the cells are read. We show that RBER can be reduced by using information from neighbor cell values. (Sections 4 and 5)
2. Based on our analyses, we propose a new method for correcting errors in flash memory, *neighbor-cell assisted correction (NAC)*, which complements ECC (Section 6). The key idea is to re-read a flash memory page that initially failed ECC with a set of read reference voltages corresponding to the conditional threshold voltage distribution assuming a neighbor cell value and use the re-read values to correct the cells that have neighbors with that value. We show that the error correction capability of particular neighbor cell values is higher than that of others because the threshold voltage changes interference caused by some neighbor cell values are more likely to lead to errors in the cells they interfere with. Based on this, we propose the idea of *prioritized NAC,* which prioritizes reading assuming such neighbor cell values (Section 6.2). We also show that even if the reading of the neighbor cells fails ECC, the read raw data from the neighbors is accurate enough to provide better correction capability than ECC alone (Section 6.3).
3. We evaluate all our techniques using real I/O workload traces and a high-fidelity simulation infrastructure that is driven by measured data from real flash chips. Our evaluations show that NAC increases flash memory lifetime by 33% without any performance loss within the nominal lifetime and with less than 5% performance loss during the extended lifetime (Section 7).

2. BACKGROUND

We briefly provide background on relevant aspects of NAND flash memory necessary to understand the rest of the paper. For more detailed background on flash memory operation and characteristics, please refer to Cai et al. [1][3][4][7][8].

2.1 Basics of NAND Flash Memory

As mentioned, the logical value stored in a flash memory cell is determined by the threshold voltage range into which the cell's actual threshold voltage falls. The threshold voltage of a flash cell can be modulated by the amount of electrons programmed on the floating gates. For n-bit MLC NAND flash memory, the threshold voltage of a cell is logically divided into 2^n separate regions and each region represents a unique n-bit value. For the specific case of 2-bit MLC NAND flash memory, the bits stored in a cell can be classified into the most significant bit (MSB) and the least significant bit (LSB), depending on the location of the bit inside the flash bit-string [1][3].

NAND flash memory generally contains thousands of blocks. A block consists of a 2-D array of flash cells. Each row of the array forms one wordline and each column forms one bitline. A block has *N* wordlines and the address of the wordline increases one by one from the bottom to the top inside a block. Thus, a cell location can be uniquely determined by its wordline and bitline address inside a block. In this paper, we define *C(x,y)* to be the flash cell that is on the *x-th* wordline and *y-th* bitline. For *All-Bit-Line* NAND flash memory, the MSBs of *all the cells on the same wordline* are programmed and read simultaneously: these set of MSBs on the same wordline are referred to as an *MSB page*. Similarly, the LSBs of all the cells on a wordline form one *LSB page*. Each page has its unique physical address inside a block. Fig. 1(a) shows an example of the page address mapping inside a flash block. We can see that the LSB page number on wordline *n* is *2n-1,* and the MSB page number on wordline *n* is *2n+2* for all-bit-line flash memory. The exceptions are the bottom wordline (i.e., wordline 0) and top wordline (i.e., wordline *N*) of a block.

2.2 NAND Flash Operations

Program Operation: Programming the threshold voltage of flash cells to the target threshold voltage region requires two steps for 2-bit MLC flash memory. These two steps are shown in Fig. 1(b): 1) LSB programming, which programs the threshold voltage of the cell into either the erased state region (ER) or a temporary state region (Temp) based on the value of LSB; and 2) MSB programming, which programs the threshold voltage of the cell

into one of the four regions (ER, P1, P2, P3) determined by both the LSB and MSB bit values.

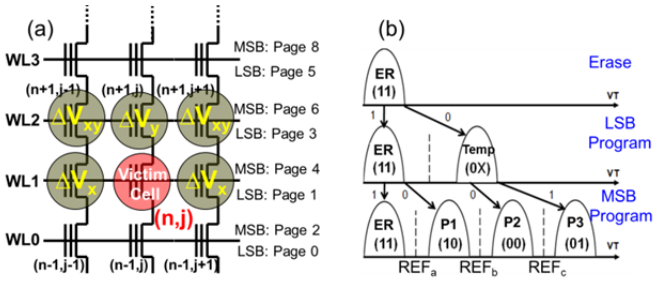

Fig. 1 (a) All-bit-line flash block architecture with the victim cell circled in red and aggressor cells circled in grey; (b) Two-bit MLC flash programming scheme. Cell states are encoded in format (LSB, MSB)

Read Operation: The read operation determines the logical value of a flash cell by determining the region its threshold voltage falls in. To read the LSB of 2-bit MLC flash memory, a reference voltage (REF_b in Fig. 1(b)) is selected to compare with the threshold voltage of the flash cell. If the threshold voltage of the flash cell is larger than REF_b, the LSB is read as 0, otherwise it is read as 1. To read the MSB, the threshold voltage of a cell is compared to two reference voltages (REF_a and REF_c in Fig. 1(b)). If the cell's threshold voltage is within the range of [REF_a, REF_c], the MSB is read as 0, otherwise it is read as 1. Each of the reference voltages is called a *read reference voltage*.

If the set of read reference voltages is set to a set of values that are not optimal, a cell's logical value may be incorrectly read (i.e., the cell may be attributed to belong to a threshold voltage window that is different from the window corresponding to the logical value it was programmed with). This leads to errors in read operation. There are at least two issues that complicate the optimal determination of read reference voltage values: 1) the threshold voltage distribution of flash cells gets distorted as the cells wear out (i.e., as the number of P/E cycles increases) [4], 2) the threshold voltages of cells get disturbed as neighboring cells are programmed [3].

Recent flash memory devices [9] allow the read reference voltages to be configurable online, which allows the flash controller to try different reference voltages in order to find one that leads to a lower error rate. This functionality is called ***read-retry***. Previous works [3][4] leverage the *read-retry* feature to identify the exact threshold voltage of each flash cell and experimentally characterize the threshold voltage distributions.

2.3 Cell-to-cell Program Interference

Due to coupling capacitance between flash cells, the threshold voltage of a flash cell can change when its neighbor cells are being programmed [3][5]. The former cell is called the *victim cell*, while the latter are called *aggressor cells*, as shown in Fig. 1(a). Generally, the pages inside a block are programmed in page number order to reduce program interference [3]. The flash cells on lower wordlines will finish programming before the MSB page programming of the upper wordlines. Thus, the cells on a lower wordline will not be the aggressor cells for those on an upper wordline. For all-bit-line flash memory, the cells on the same wordline are programmed simultaneously and finish programming at the same time [3][5]. Therefore, program interference from neighbor cells on the same wordline is negligible, as demonstrated in [3][5]. The program interference on a victim flash cell *C(n,j)* due to the aggressor cells on the wordline that is immediately above the victim wordline and can be modeled as [3]:

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

Here V_{victim}^{before} is the threshold voltage of the victim cell before aggressor cells are programmed and $\Delta V_{neighbor}(x, y)$ is the threshold voltage change of the aggressor cell *C(x,y)*. $\alpha(x,y)$ is a positive coefficient, which represents the ratio of the coupling capacitance between the aggressor cell *C(x,y)* and the victim cell *C(n,j)* over the total capacitance of the victim cell *C(n,j)*. α_0 is a negative coefficient, which models the fact that the victim cell with more electrons programmed tends to suffer less from program interference. The coupling coefficients decrease exponentially with the distance between neighbor cells [3] and the dominant interference to a victim cell in all-bit-line NAND flash comes from the aggressor cell that is directly above the victim [3], i.e. the *direct or immediate neighbor* (e.g., aggressor *C(n+1, j)* to victim *C(n, j)*). Thus, equation (1) can be simplified to:

$$\Delta V_{victim}(n,j) \approx k \times \Delta V_{neighbor}(n+1,j) \quad (2)$$

3. OUR GOALS AND RELATED WORK

Our goal in this work is threefold. First, we would like to empirically and statistically analyze the threshold voltage distributions of flash cells *conditionally upon the values of immediately neighboring cells*. Second, we would like to find better sets of read reference voltages that can minimize read error rates *by using information from neighboring cell values*. Third, we aim to devise new error correction mechanisms that can *take advantage of the values of neighboring cells* to reduce error rates after the application of conventional ECC.

To our knowledge, no previous work explored the effect of neighboring cell values on flash memory threshold voltage distributions and provided ways of taking advantage of neighboring cell *values* to minimize flash memory error rates. We briefly discuss the most relevant works below.

Cai et al. [4] was the first to characterize and propose an empirical model for threshold voltage distributions in MLC NAND flash memory. Cai et al. [3] also characterized the effect of programming of neighboring cells (program interference) on threshold voltage distributions of cells. Building upon the empirical measurements in [3], they proposed a mechanism that can predict and use a set of read reference voltages to minimize bit error rate, taking into account the changes in the *overall threshold voltage distribution* of cells due to program interference. However, [3] did not characterize or take advantage of the fact that the changes in threshold voltage distribution are dependent on the *values* programmed in the neighboring aggressor cells. This paper builds upon [3] and shows that knowing the values stored in neighboring cells can lead to the determination of a better set of read reference voltages that can further reduce bit error rate and improve lifetime compared to the neighbor-cell-value-unaware read reference voltage prediction mechanism of [3] (Section 7.1).

Two other works [10][11] proposed signal processing techniques to overcome program interference effects in flash memory. However, these works face multiple implementation challenges and are based on hypothesized (as opposed to experimentally measured and validated) models for program interference, which we briefly summarize. First, these works rely on accurate measurement of 1) the threshold voltage changes of neighbor aggressor cells and 2) threshold voltages of the victim cells. This means that an n-bit representation (e.g. 4~6 bits to represent the threshold voltage for 2-bit MLC flash) needs $2x2^n$ read operations to sense the threshold voltage of neighbor aggressor cells (LSB

and MSB pages) and 2^n read operations for the victim cell to be read. This will cause severe performance and energy overheads during a read operation. Second, these works need to record the threshold voltage *changes* of all the aggressor cells, which cannot be accurately obtained as at the time of reading: only the *current* threshold voltage of neighboring aggressor cells can be measured while their *past* threshold voltages before they were fully programmed are difficult to obtain. To circumvent this problem, [11] uses the mean threshold voltage value of the erased (ER) state to estimate the threshold voltages of *all* aggressor cells before they are programmed, which is inaccurate (see Cai et al. [4], which shows that the cells in the same state can have greatly different threshold voltages) and difficult to implement (it is difficult to apply negative read voltage on the control gate to measure threshold voltages of cells in the erased state). In contrast to these works, the mechanisms proposed in this paper to improve bit error rates do not need to measure, record or estimate the voltage of aggressor cells *before* they were fully programmed.

To our knowledge, this is the first work that uses information stored in neighbor cells to correct errors in a victim cell with low-overhead mechanisms that can be implemented practically. We propose to still use conventional ECC, such as BCH codes [12], to correct most of the errors (including those due to program interference) upon reading a flash cell, but leverage information of neighbor cells *only when* conventional ECC fails and minimize the number of read operations on the neighbor cells by taking advantage of our experimental observations from real flash chips.

4. Flash Voltage Distribution Measurement

We first describe our methodology to measure cell threshold voltage distributions using real flash memory chips. As shown in previous work [4], the threshold voltages of different flash cells are different even when the cells are programmed with the same value. This is due to manufacturing process variations within a flash chip. The threshold voltage of flash cells can be represented by a random variable x. The probability density function (PDF) of x, $p(x)$, is generally called the (*threshold) voltage distribution*. The value programmed on a cell's direct-neighbor aggressor cell is a discrete bit value, which can be denoted as z. For n-bit MLC NAND flash memory, z can take up to 2^n values and can be treated as a discrete random variable. As such, the voltage distribution of flash cells can be expressed as:

$$p(x) = \sum_{m=1}^{2^n} p(z=m)\,p(x \mid z=m) \tag{3}$$

where $p(z=m)$ is the probability that the direct-neighbor aggressor cell is programmed with the logical data value m. $p(x|z=m)$ is the conditional probability that the threshold voltage of a (victim) cell equals x given that value m is programmed to its direct-neighbor aggressor cell z. Equation (3) can be reformulated as:

$$p(x) = \sum_{m=1}^{2^n} p(x, z=m) \tag{4}$$

where $p(x, z=m)$ is the joint probability density function of random variable x (threshold voltage of the victim cell) and z (value programmed in the direct-neighbor aggressor cell). For the rest of this paper, we refer to $p(x, z=m)$ as the *conditional distribution* and $p(x)$ as the *overall distribution* of threshold voltages for all flash memory cells.

Experimental Measurement Methodology: To measure the threshold voltage distributions, we use an FPGA-based testing platform [4][13] that can issue commands to raw flash chips. We tested 2-bit MLC NAND flash memory devices manufactured in 2Y-nm technology. There are four conditional distributions $p(x, z=11)$, $p(x, z=10)$, $p(x, z=00)$ and $p(x, z=01)$. Recall that $p(x, z=m)$ is equal to $p(z=m)p(x|z=m)$. $p(z=m)$ is measured by dividing the number of direct-neighbor aggressor cells with value $z=m$ by the total number of direct-neighbor aggressor cells. Since random (or pseudo-random) data is generally programmed into raw flash chips due to data encryption and randomization techniques employed before programming [14][15], we measure the threshold voltage of cells with random data values programmed in all cells. As a result, $p(z=m)$ is approximately 25% for each possible value of m. Each *conditional distribution* $p(x|z=m)$ is measured via the following steps: 1) program flash memory with random data; 2) select all the cells whose direct neighbor aggressor cell is programmed with value $z=m$ and measure the threshold voltage of these selected cells; 3) count the number of cells with threshold voltage equal to x; 4) divide the count found in step 3 by the total number of cells selected in step 2. The *overall distribution* $p(x)$ can be measured by dividing the number of cells whose threshold voltage equals x over the total number of flash cells.

Example Distribution and Initial Observations: Fig. 2 shows the overall distribution (dotted line) and all four conditional distributions (solid line) for a flash memory chip tested after 35k P/E cycles (More details on threshold voltage distribution measurement methodology can be found in our past works [3][4].). We make several observations, which we will formally back up in Section 5 with statistical analyses. We will also use some of these observations to develop new error correction methods in Section 6. First, and most importantly, *two neighboring states in each conditional distribution are farther apart from each other than they are in the overall distribution*, as shown by the "large margin" and "small margin" arrows in the figure between states P1 and P2. This indicates that accurately distinguishing between the neighboring states (i.e., identifying the logical values of flash cells) can be easier if the conditional threshold voltage distribution is used instead of the overall distribution, which in turn suggests that knowing the value of the immediate-neighbor cell can enable more accurate identification of the logical value of a cell when it is read. Second, for a given state, each of the conditional distributions has a smaller variance (informally, spread of the threshold voltage values belonging to that state) than the overall distribution. Third, the overall distribution is the sum of all the conditional distributions (as expected). Fourth, the conditional distributions for direct-neighbor cell values 10 and 01 are similar to each other and have the highest average threshold voltage values for each state, whereas the conditional distribution for the direct-neighbor cell value 11 (corresponding to the Erased state in the neighbor cell) has the lowest average threshold voltage values for each state. This latter observation is expected because programming a direct-neighbor cell to 11 (the erased state) leads to the smallest amount of injected charge into that cell (causing the smallest amount of program interference to the victim cell) and programming a direct-neighbor cell to 10 or 01 leads to the largest amount of injected charge as can be seen in Fig. 1(b) (causing the largest amount of program interference to the victim cell).

5. Flash Voltage Distribution Analysis

In this section, we first statistically analyze the threshold voltage distribution characteristics of flash memory from a formal standpoint and develop a model for both overall and conditional threshold voltage distributions (Sections 5.1-5.3). We then provide empirical measurement results that validate the statistical

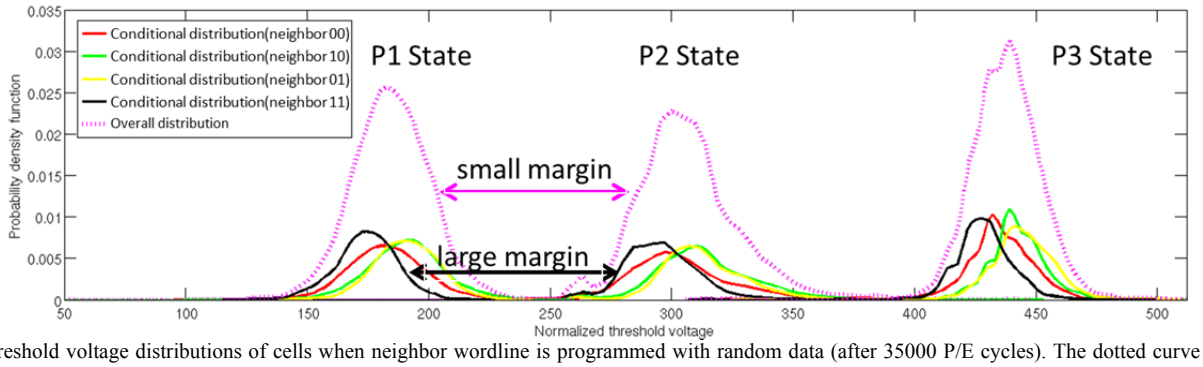

Fig. 2 Threshold voltage distributions of cells when neighbor wordline is programmed with random data (after 35000 P/E cycles). The dotted curve shows the overall distribution of all cells. Solid curves show the four conditional distributions corresponding to four possible direct-neighbor cell values.

model (Sections 5.4-5.5). In particular, our statistical analyses are to show that: 1) there exists an optimum read reference voltage between two neighboring logical states that minimizes the raw bit error rate (Section 5.1), 2) raw bit error rate can be minimized by either increasing the distance between the average threshold voltage values of neighboring states or reducing the variance of threshold voltage distributions of each logical state (Section 5.2), and 3) the conditional distributions for each logical state have smaller variances and larger signal-to-noise ratios than the overall distribution and therefore using the conditional distribution to determine the value of a cell that is being read leads to a smaller bit error rate than using the overall distribution (Section 5.3).

5.1 Optimizing the Read Reference Voltage

We first show that there exists an optimum read reference voltage to distinguish between two logical states in a threshold voltage distribution. Fig 3(a) shows an illustration of the threshold voltage distributions of two neighboring states of NAND flash memory. Assume that the probability density functions (PDFs) of cells programmed into state P_i and state P_{i+1} are *f(x)* and *g(x)* respectively. When voltage V_{ref} is selected as the read reference voltage to differentiate between these two neighboring states during a read operation, the fraction of cells that are actually programmed into the P_i state but misread as belonging to the P_{i+1} state is shown in the blue area in Fig. 3(a). We call this fraction as the *error rate due to P_i misread as P_{i+1}* and formulate it as:

$$ErrRate^{P_i->P_{i+1}} = \int_v^{+\infty} f(x)dx \tag{5}$$

Similarly, the fraction of cells that are actually programmed into the P_{i+1} state but misread as belonging to the P_i state is shown in the red area in Fig. 3(a) and can be formulated as:

$$ErrRate^{P_{i+1}->P_i} = \int_{-\infty}^{v} g(x)dx \tag{6}$$

Assume that the probability of a cell to be programmed into the P_i state is P_0 and the probability of a cell to be programmed into the P_{i+1} state is P_1. Then, the total error rate is the sum of equations (5) and (6) weighted by these probabilities:

$$ErrRate^{total} = P_0 \times \int_v^{+\infty} f(x)dx + P_1 \times \int_{-\infty}^{v} g(x)dx \tag{7}$$

The optimum read reference voltage, V_{opt}, which minimizes the total error rate between these two states, is obtained by setting $\partial ErrRate^{total} / \partial v$ to zero. V_{opt} satisfies the following equation:

$$P_0 \times f(V_{opt}) = P_1 \times g(V_{opt}) \tag{8}$$

If the probabilities of programming a cell into the P_i state and P_{i+1} state are equal (i.e., P_0 equals to P_1), the optimum read reference voltage V_{opt} satisfies the criteria $f(V_{opt})=g(V_{opt})$ and V_{opt} is at the cross-point of neighboring distributions, as shown in [3].

Previous works [4][16] have shown that the threshold voltage distributions of each state approximately follow the Gaussian distribution. Due to the characteristics of the Gaussian distribution, the distribution PDF can be completely defined by the statistics of the mean and variance pair (e.g., (μ_1, σ_1) for the P_i state and (μ_2, σ_2) for the P_{i+1} state, as illustrated in Fig. 3(a)). We replace the *f(.)* and *g(.)* functions in Equation (8) with Gaussian PDFs. Thus, the optimum read reference voltage V_{opt} between two neighboring states should satisfy:

$$P_0 \times \frac{1}{\sqrt{2\pi}\sigma_1} \exp(-\frac{(v_{opt}-\mu_1)^2}{2\sigma_1^2}) = P_1 \times \frac{1}{\sqrt{2\pi}\sigma_2} \exp(-\frac{(v_{opt}-\mu_2)^2}{2\sigma_2^2}) \tag{9}$$

By applying logarithmic operation to both sides of Equation (9) and solving the quadratic equation, we can get the optimum read reference voltage as $V_{opt} = (-B \pm \sqrt{B^2 - 4C})/2$. Here, B is $(\mu_2\sigma_1^2 - \mu_1\sigma_2^2)/\sigma_1^2\sigma_2^2(\sigma_2^2-\sigma_1^2)$ and C is $\frac{\mu_1^2\sigma_2^2 - \mu_2^2\sigma_1^2}{2\sigma_1^2\sigma_2^2(\sigma_2^2-\sigma_1^2)} - (\log\frac{P0}{P1} + \log\frac{\sigma_2}{\sigma 1})$. When the variances of the two neighboring states are approximately equal (σ_1 and σ_2 equal to value σ), the optimum read reference voltage that achieves the minimum raw BER can be simplified as:

$$V_{opt} = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_2 - \mu_1} \log\frac{P0}{P1} \tag{10}$$

If the probabilities of programming a cell into the P_i state and P_{i+1} state are equal (i.e., P_0 equals to P_1), which is the case for data values that are random or pseudo-random (which was shown for real flash memory chips due to the heavy use of data encryption and randomization techniques employed before programming [14][15]), the optimum read reference voltage is the average of the mean threshold voltages of the neighboring states:

$$v_{opt} = (\mu_1 + \mu_2)/2 \tag{11}$$

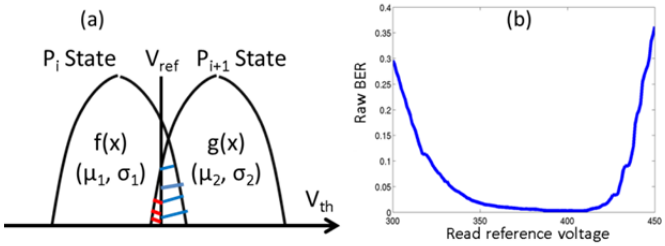

Fig. 3 (a) Read reference voltage V_{ref} between neighboring states. The area marked by blue lines correspond to Equation 5 and red lines correspond to Equation 6; (b) Measured raw BER vs. read reference voltage between P2 and P3 states at 35k P/E cycle endurance.

Measurement: To experimentally demonstrate how raw BER changes with read reference voltage, we sweep the read reference voltage between neighboring states and calculate the

corresponding raw BER on our testing platform. Fig. 3(b) shows raw BER vs. read reference voltage when sweeping the latter from the center of the P2 state to the center of the P3 state for flash memory at 35k P/E cycles. We can see that raw BER first decreases as read reference voltage increases but increases after a certain point. There exists a point that can achieve minimum raw BER, as these empirical results demonstrate.

5.2 Minimizing Raw Bit Error Rate

The minimum raw bit error rate can be obtained by setting the variable v in Equation (7) to be the optimum read reference voltage. To get a simple closed-form expression, we assume that the threshold voltage distributions follow Gaussian distributions with equal variance, as shown as an approximation in [3][16]. The mean values of the neighboring states are μ_1 and μ_2 respectively. We also assume that the probabilities for each state are equal since random data are programmed to cells [14][15] and thus the optimum read reference voltage is $(\mu_1+\mu_2)/2$ according to Equation (11). Thus, the minimum raw BER can be expressed as:

$$\frac{1}{2}\times\frac{1}{\sqrt{2\pi}\sigma}\left(\int_{(\mu1+\mu2)/2}^{+\infty}\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)dx+\int_{-\infty}^{(\mu1+\mu2)/2}\exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right)\right)dx \quad (12)$$

After a series of algebraic manipulation, this can be simplified to:

$$\frac{1}{\sqrt{2\pi}}\int_{(\mu2-\mu1)/2\sigma}^{+\infty}\exp(-x^2/2)dx \quad (13)$$

To simplify the notation, we define the function $Q(x)$ as:

$$Q(x)=\frac{1}{\sqrt{2\pi}}\int_{x}^{+\infty}\exp(-x^2/2)dx \quad (14)$$

By setting x to be $(\mu_2-\mu_1)/2\sigma$ in $Q(x)$, we can obtain the minimum raw BER. Fig. 4 plots the function $Q(x)$. We can see that $Q(x)$ monotonically decreases as x increases. Thus, in order to decrease the minimum raw BER, it is desirable to have distributions that have a higher value of $(\mu_2-\mu_1)/2\sigma$. This is possible in two ways:

1. Have a larger threshold voltage distance $(\mu_2-\mu_1)$ between neighboring distributions. The farther apart from each other the neighbor distributions, the larger the value of $(\mu_2-\mu_1)/2\sigma$.
2. Have a smaller variance σ^2 for the threshold voltage distributions. The narrower the distributions (i.e., their standard deviation σ), the larger the value of $(\mu_2-\mu_1)/2\sigma$.

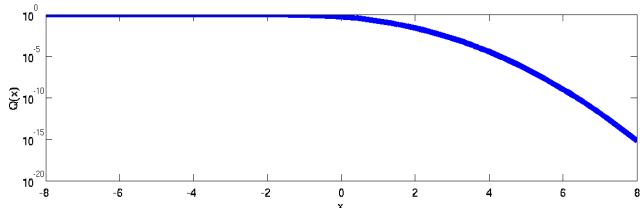

Fig. 4 Monotonic property of Q(x): Q(x) decreases as x increases

Signal to Noise Ratio (SNR): The half distance $(\mu_2-\mu_1)/2$ between neighboring states can be thought of as the signal and the standard deviation σ of the threshold voltage distribution can be thought of as noise. The ratio $(\mu_2-\mu_1)/2\sigma$ can therefore be defined as signal-to-noise ratio (SNR) when reading flash memory. The larger the SNR ($(\mu_2-\mu_1)/2\sigma$), the smaller the minimum raw BER, which is equal to Q(SNR).

The next section shows that conditional distributions of flash memory cells that distinguish between the values of the direct-neighbor cell (as opposed to the overall distribution that does not) lead to a larger SNR: two neighboring states in each conditional distribution have a similar threshold voltage distance $(\mu_2-\mu_1)$ *as* the same two neighboring states in the overall distribution but they have a significantly smaller variance σ^2. As such, using conditional distributions to determine the value of cells improves SNR, thereby reducing the minimum raw BER.

5.3 SNR and Minimum Raw BER Analysis of Conditional and Overall Distributions

Since a cell's direct-neighbor aggressor cell can take up to $N=2^n$ possible values for n-bit MLC flash memory, the threshold voltage X of a cell can be one of the N conditional random variables x_m as $X\in\{x_0,x_1,x_2,x_3,...,x_{N-1}\}$. Here, x_m represents the threshold voltage of a cell with direct neighbor aggressor cell programmed to value m. *The PDF of X is the overall threshold voltage distribution. The PDF of x_m is the conditional distribution*. We now statistically analyze the distance of neighboring distributions, the variance of distributions, and their resulting SNR and BER comparatively between the overall distribution and conditional distributions. Our key finding is that the threshold voltage distance between neighboring states is similar for both the overall distribution and the conditional distributions, while the variance of the overall distribution is much larger than those of the conditional distributions, and thus using the conditional distributions to determine the optimum set of read reference voltages leads to larger SNR and smaller BER.

Distance of Neighboring Distributions: The mean of the overall distribution is the average of the means of all the conditional distributions (i.e., $E(X)=\sum E(x_m)/N$). Therefore, the distance of neighboring states in the overall distribution is equal to:

$$\begin{aligned}&E(X^{P(i+1)})-E(X^{P(i)})\\&=\frac{1}{N}\sum E(x_m^{P(i+1)})-\frac{1}{N}\sum E(x_m^{P(i)})=\frac{1}{N}\sum(E(x_m^{P(i+1)})-E(x_m^{P(i)}))\end{aligned} \quad (15)$$

where $E(X^{P(i+1)})$ and $E(X^{P(i)})$ are the mean values of the threshold voltage distributions of all the cells in state P_{i+1} and state P_i respectively. $E(x_m^{P(i+1)})$ and $E(x_m^{P(i)})$ are the means of the *conditional* distributions for cells in state P_{i+1} and P_i respectively with direct aggressor cell programmed to value m. $E(x_m^{P(i+1)})-E(x_m^{P(i)})$ is the distance between neighboring states in the same conditional distribution. We can see from Equation (15) that the distance of two neighboring states in the overall distribution is the average of the distances of the same two neighboring states in all of the conditional distributions.

Variance of Distributions: The variance of the threshold voltage of all the cells in a state is equal to $Var(X)=E(X^2)-(E(X))^2$. By applying $E(X^2)=\Sigma E(x_m^2)/N$ and $E(X)=\Sigma E(x_m)/N$, we can get the variance of the threshold voltage distribution after algebraic manipulation:

$$Var(X)=\frac{1}{N}\sum Var(x_m)+\frac{1}{N^2}\sum\sum(E(x_m)-E(x_n))^2 \quad (16)$$

From Equation (16), we can see that the variance of the overall distribution consists of two parts: 1) the average of the variances of all of its conditional distributions, $\sum Var(x_m)/N$ and 2) the average of the square of the differences between the mean values of every pair of conditional distributions, $\sum\sum(E(x_m)-E(x_n))^2/N^2$. Since the latter is non-negative, the variance of the overall distribution is no less than the average of the variance of all of its conditional distributions. We can consider two hypothetical extreme cases:

1. The conditional distributions completely overlap with each other. As a result, the variance of the overall distribution would simply be the average of the variances of all of its conditional distributions since $E(x_m)-E(x_n)$ is zero for (m,n) pairs (i.e., the second sum part of Equation (16) is zero).

2. The conditional distributions are completely separate (i.e., far away) from each other. As a result, the variance of the overall distribution would be much larger than the average variance of all of its conditional distributions and would be dominated by the second sum part of equation (16).

We next analyze the mean values of conditional distributions to show that the latter hypothetical case is closer to empirical reality and, as a result, the variance of the overall distribution is much larger than the average variance of the conditional distributions.

Analysis of the Mean Values of Conditional Distributions: Assume the threshold voltage of a cell before program interference is a random variable Y, and the voltage change caused by the program interference is ΔV. Then the threshold voltage after program interference will be X=Y+ΔV. As in equation (2), ΔV is mainly determined by the threshold voltage change of the direct-neighbor aggressor cell [3]. Assuming the threshold voltages of the direct-neighbor aggressor cell before and after programming value m are $V_{before}^{Nb(m)}$ and $V_{after}^{Nb(m)}$ respectively, then ΔV approximately equals $k \times (V_{after}^{Nb(m)} - V_{before}^{Nb(m)})$ according to Equation (2), where k is the coefficient denoting the coupling capacitance over the total capacitance of the victim cell. Then, the mean value of the conditional distribution x_m is

$$E(x_m) \approx E(Y) + E(k) \times (E(V_{after}^{Nb(m)}) - E(V_{before}^{Nb(m)})) \quad (17)$$

As the LSB page number of an upper wordline is always smaller than the MSB page number of a lower wordline in the same block (see Fig. 1(a)) and the pages inside a block are programmed in sequential page number order to minimize program interference [3], only MSB page programming of the aggressor cell affects the threshold voltage of the victim cell. For 2-bit MLC flash memory (as in Fig. 1(b)), the starting state before MSB page programming is either the ER state or the Temp state, and the ending state could be the ER, P1, P2 or P3 state. When either P3 (value 01) or P1 (value 10) is programmed to the aggressor cell, the threshold voltage of the aggressor cell changes either from TEMP to P3 or from ER state to P2, respectively. Either of these cases leads to a larger threshold voltage change $E(V_{after}^{Nb(m)}) - E(V_{before}^{Nb(m)})$, as illustrated in Fig. 1(b) [3], compared to when ER (value 11) or P2 (value 00) is programmed to the aggressor cell. Thus, the mean values of conditional distributions of x_{01} and x_{10} are larger than the mean values of the conditional distributions of x_{11} and x_{00}. When value 11 is programmed on aggressor cell, the starting and ending states are the same, i.e., the ER state, which corresponds to the smallest threshold voltage change. Therefore, the mean of the conditional distribution of x_{11} is the smallest [3]. Through similar reasoning, the mean of the conditional distribution of x_{00} sits in-between the mean of the conditional distribution of x_{11} and the mean of the conditional distributions of x_{01} and x_{10}. In summary, the means of conditional distributions for 2-bit MLC satisfy the following property:

$$E(x_{11}) < E(x_{00}) < E(x_{01}) \approx E(x_{10}) \quad (18)$$

Fig. 2 shows that this property empirically holds for different conditional distributions; in Fig. 2, it is clear that the mean threshold voltage value of the conditional distributions with direct neighbor values 01 and 10 are similar and the highest for all states, whereas the mean value of the conditional distribution with direct neighbor value 11 is the smallest.

The result in Equation (18), empirically supported by Fig. 2, shows that the second sum part of Equation (16) is clearly greater than zero as the mean values of conditional distributions do not completely overlap. As a result, the variance of the overall distribution is larger than the average variance of the conditional distributions.

Analysis of Variance of Conditional Distributions: The variance of the conditional distribution x_i can be expressed as:

$$\mathrm{var}(x_m) \approx \mathrm{var}(Y) + k^2 \times (\mathrm{var}(V_{after}^{Nb(m)}) + \mathrm{var}(V_{before}^{Nb(m)})) \quad (19)$$

Due to the accurate control of the distribution width of each programmed state through incremental step pulse programming (ISPP), the variances of threshold voltage distributions of aggressor cells before and after programming are approximately close for different programming values [4][16]. For example, $var(V_{after}^{Nb(m)})$ is close to $var(V_{after}^{Nb(n)})$, even when the programmed value m and n are not the same. Also, the model coefficient k is around 0.06 [3] and the small value of k^2 can further reduce the value-dependent differences of variances of different conditional distributions in Equation (19). Therefore, the variances of conditional distributions of victim cells are close when different values are programmed to their direct-neighbor aggressor cells (e.g., $var(x_m)$ is close to $var(x_n)$ even if m is not equal to n) and thus are approximately equal to the average variance of all conditional distributions. Since the variance of the overall distribution is larger than the average of the variances of conditional distributions (as we have shown based on Equation (18)), the variance of the overall distribution is larger than the variance of each of the conditional distributions.

Analysis of the Distance of Distributions between Neighboring States: Assume the threshold voltages of cells that are interfered with by a direct-neighbor with value m in the P_i and P_{i+1} states are respectively x_m^{Pi} and x_m^{Pi+1}. The threshold voltages before neighbor interference are respectively Y^{Pi} and Y^{Pi+1}, which are independent of the value m programmed into the direct-neighbor cells. The amounts of interference the victim cells receive are respectively Δ_m^{Pi} and Δ_m^{Pi+1}. Thus, the distance of the conditional distributions between the two neighboring states is $E(x_m^{Pi+1})$-$E(x_m^{Pi})$, which is equal to $E(Y^{Pi+1}+\Delta_m^{Pi+1})$-$E(Y^{Pi}+\Delta_m^{Pi})$. Since program interference is mainly determined by the threshold voltage changes on the direct-neighbor aggressor cells, $E(\Delta_m^{Pi+1})$ and $E(\Delta_m^{Pi})$ are close as the neighbors are programmed to the same value m. Thus, $E(x_m^{Pi+1})$-$E(x_m^{Pi})$ is similar to $E(Y^{Pi+1})$-$E(Y^{Pi})$, which is equal to the distance of the neighboring distributions before neighbor cells are programmed and is independent of the exact values programmed to the direct-neighbor aggressor cells. Thus, the distance of the different conditional distributions in neighboring states are equal (or close). Therefore, the distance of the overall distributions between neighboring states is also equal to or close to the distance of any of the conditional distributions between neighboring states.

Final SNR and BER Analysis: So far, we have shown that: 1) the variance of each of the conditional distributions is smaller than the variance of the overall distribution, 2) the distance of the conditional distributions of two neighboring states is similar to the distance of the overall distributions of the same two neighboring states. As a result, the signal-to-noise ratio (SNR), $(\mu_2-\mu_1)/2\sigma$, of the overall distribution is smaller than that of the conditional distribution, as σ of the overall distribution is larger. Therefore, the minimum raw BER, i.e., $Q(SNR)$ (see Equations (13) and (14)), obtained with the set of read reference voltages,determined based on the overall distribution would be larger than the minimum raw BER obtained with the set of read reference voltages determined based on the conditional distributions. We will make use of this statistically-demonstrated conclusion to devise our error correction mechanisms in Section 6.

5.4 Experimental Measurement Results

We now show experimental results that empirically validate the major conclusions of the statistical model we have developed so far in Sections 5.1-5.3. We program random data into 2Y-nm 2-bit MLC NAND flash devices after 10K P/E cycles and leverage the methodology discussed in Section 4 to measure the overall and the conditional distributions. Table 1 shows the results: the average distance of different distributions between neighboring states (signal), along with the average variance (noise), the signal-to-noise ratio (SNR) and the resulting minimum raw BER of the overall distribution and all four conditional distributions. The empirical data supports the main conclusions we have drawn from statistical analyses: 1) average distance between neighboring states is similar regardless of whether we use the overall distribution or any of the conditional distributions, 2) the variance, the SNR, and the resulting BER of each of the conditional distributions is smaller than those of the overall distribution. As such, experimental data also favors using conditional distributions to maximize SNR and therefore minimize BER. Section 5.5 explains the BER evaluations in more detail.

Table 1. Measurement Results of Different Distributions

	Overall	x_{11} (ER)	x_{10} (P1)	x_{00} (P2)	X_{01} (P3)
Distance	65.4	65.4	64.7	66.4	65.8
Variance	385.9	286.2	256.7	242.8	252.1
SNR	3.4	3.8	3.9	4.2	4.1
BER	$3x10^{-4}$	$7x10^{-5}$	$5x10^{-5}$	$2x10^{-5}$	$3x10^{-5}$

5.5 Minimizing BER using Conditional Distributions – Mechanism and Measurements

To measure the minimum BER of reading with the overall distribution, we set the read reference voltage between two neighboring states to $REFx=(\mu^{P(i)}+\mu^{P(i+1)})/2$, which is the average of the mean values of the overall distribution in neighboring states P_i and P_{i+1}. We define this set of read reference voltages as the *global optimum read reference voltage*. The read data is compared with the originally programmed data to count the number of errors. The BER is measured by dividing the error count with the total number of bits read.

To measure the minimum BER of reading with each conditional distribution (e.g., x_{10}), we first set the read reference voltage between states P_i and P_{i+1} to be $REF_{x10}=(\mu_{x10}^{P(i)}+\mu_{x10}^{P(i+1)})/2$. Here, $\mu_{x10}^{P(i)}$ and $\mu_{x10}^{P(i+1)}$ are the mean values of the conditional distribution of x_{10} in the P_i state and the P_{i+1} state respectively. Then, we read flash memory using this read reference voltage. We only compare the data read out from the cells that have direct-neighbor aggressors with value 10 to the data originally programmed to those cells to get the raw BER of reading with the conditional distribution of x_{10}. We repeat the same procedure three more times by setting the read reference voltages between two neighboring states to $REF_{x11}=(\mu_{x11}^{P(i)}+\mu_{x11}^{P(i+1)})/2$, $REF_{x00}=(\mu_{x00}^{P(i)}+\mu_{x00}^{P(i+1)})/2$ and $REF_{x01}=(\mu_{x01}^{P(i)}+\mu_{x01}^{P(i+1)})/2$ to get the raw BER of reading with conditional distributions of x_{11}, x_{00} and x_{01} respectively. The final raw BER of reading with conditional distributions is the equally weighted sum of the raw BER obtained after reading with each of the conditional distributions (with a weight of 0.25 for each BER as direct-neighbor cells have equal probabilities of being programmed to 00, 01, 10, 11). We define these four sets of read reference voltages as the *local optimum read reference voltages*. Based on Equation (18) and our empirical measurements, the read reference voltages obey the following property:

$$REFx_{11} < REFx_{00} < REFx < REFx_{01} \approx REFx_{10} \quad (20)$$

The threshold voltage distributions before and after random data are programmed in neighbor cells are illustrated in Figure 5(a) and 5(b) respectively. The relative location of the global and local optimum read reference voltages are illustrated in Fig. 5(b). N11, N00, N01 and N10 illustrate the conditional distributions of those cells with direct-neighbor aggressor cell programed to 11, 00, 01 and 10 respectively. The optimum BERs of reading with the overall distribution and the conditional distributions are shown on the fourth row of Table 1. Reading the flash memory cells using the set of the local optimum read reference voltages for each conditional distribution leads to a raw BER of $4.25x10^{-5}$ (equal to 0.25 times the sum of the raw BER values for each state in Table 1) whereas reading the same cells with the global optimum set of read reference voltages based on only the overall distribution leads to a raw BER of $3x10^{-4}$. As such, using the conditional distributions to determine read reference voltages to read flash memory leads to an 86% reduction in minimum raw bit error rate.

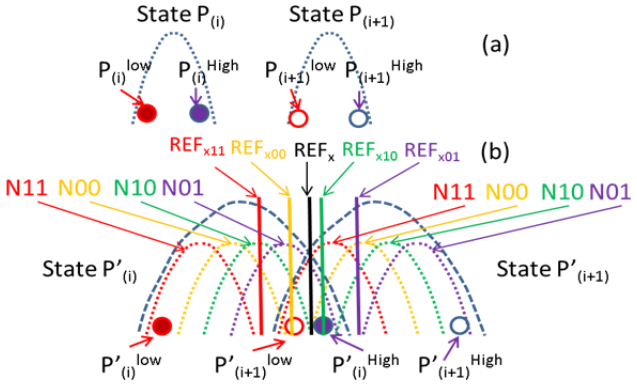

Fig. 5 (a) Threshold voltage distributions of two neighboring states *before* direct-neighbor cells are programmed; (b) Overall and conditional threshold voltage distributions, the global optimum read reference voltage of the overall distribution (REFx), and local optimum read reference voltage of each conditional distribution (REF_{x11}, REF_{x00}, REF_{x10}, REF_{x01}) *after* neighbor pages are fully programmed.

5.6 Summary

Our experimental results (Sec. 5.4-5.5) validate our statistical analyses (Sec. 5.1-5.3) in that: 1) there exists an optimum read reference that can achieve the minimum raw BER; 2) the minimum raw BER decreases as signal-to-noise ratio increases; 3) the distance (signal) of the overall distribution between neighboring states is close to that of each of the conditional distributions; 4) the variance (noise) of each conditional distribution is smaller than that of the overall distribution; 5) the variances of different conditional distributions are close; 6) the signal-to-noise ratio of the conditional distribution is larger than that of the overall distribution; 7) the minimum raw BER obtained after reading with the conditional distribution is much smaller than that obtained after reading with the overall distribution.

6. Neighbor-cell Assisted Error Correction

In Section 5, we have established that the bit error rate obtained by reading flash memory with the set of *local optimum read reference voltages* for each conditional threshold voltage distribution dependent on the direct-neighbor cell value is significantly lower than the bit error rate obtained by reading flash memory with the conventional set of *global optimum read reference voltages* determined based on the overall threshold voltage distribution of *all* flash cells. In this section, we build upon this observation to introduce two mechanisms that both aim to minimize bit error rate: ***Neighbor-cell Assisted Reading (NAR)*** and ***Neighbor-cell Assisted Error Correction (NAC)***.

We have actually already described the key idea of ***Neighbor-cell Assisted Reading (NAR)*** in Sec. 5.5. **NAR** classifies the cells to be read in a wordline into N types based on the values stored in the corresponding direct-neighbor aggressor cells (N is equal to 4 in 2-bit MLC flash because the direct-neighbor can take 4 possible values). To read the cells of each type, a different set of local optimum read reference voltages (that minimizes the bit error rate) is used (i.e., REF_{x11}, REF_{x00}, REF_{x10}, REF_{x01} in Sec. 5.5). Thus, determining the values of cells in an entire flash page is an iterative operation that takes N reading steps: in every step, the values of those cells with the same common direct-neighbor value are determined, which corresponds to approximately 1/N-th of the entire page's contents. For example, first the page is read to determine the values of cells that have direct neighbor values 11 using the set of local optimum read reference voltages corresponding to the conditional distribution assuming neighbor values of 11. Afterwards, the step is repeated for cells with neighbor values 00, 10, 01, respectively. At the end of all steps, the cell values determined in each step are combined to get the complete page. This reduces the bit error rate because each type of cell is read more accurately using its corresponding local optimum read reference voltage.

Unfortunately, NAR significantly degrades read latency because determining the contents of a page requires N+2 read operations (whereas using the global optimum read reference voltage requires only one read operation): NAR first needs to read the LSB page and the MSB page of the direct-neighbor cells to determine the contents of direct-neighbors, which requires two read operations. After that, NAR needs to read the selected page N times to determine the contents of the cells classified into N types based on direct-neighbor cell values. The increase in the number of read operations from 1 to N+2 (i.e., 6 in the case of 2-bit MLC flash memory) degrades read latency to by N+2 times and thus can severely degrade system performance.

To mitigate such performance degradation with NAR, we propose ***Neighbor-cell Assisted Error Correction (NAC***). The idea is to first read the page with the global optimum read reference voltage (which requires only one read operation) and trigger NAR only if ECC fails to correct the page data. Doing so requires a total of $(1+P_{fail} \times (N+2))$ read operations, where P_{fail} is ECC failure rate when reading with the global optimum reference voltage. If the ECC failure rate is low, average read latency would still be low. Extending this reasoning, we further optimize NAC by checking whether or not ECC can correct the current set of data after each step that reads 1/N-th of the page. After a set of local optimum read reference voltages is used to read 1/N-th of the page more accurately, ECC is applied to the entire page. If ECC can correct the entire page, then there is no need to go to the next step, i.e., apply the next set of local optimum read reference voltages to correct the next 1/N-th of the page. The next step is triggered only if ECC fails in the previous step. Thus, number of read operations is further reduced.

6.1 NAC Flow and System Operation

We briefly describe how NAC fits into the flash-based SSD controller architecture (shown in Fig. 6). The SSD controller usually contains an internal buffer to store requests and the data to be programmed and recently read. To correct raw bit errors in flash media, binary BCH codes [12], which can correct multiple bit errors, are widely deployed. The data is encoded before programming and decoded after reading. To increase error correction capability, we add two NAC-specific blocks to the controller: 1) the engine that performs NAC, 2) we reserve a small portion of the controller's internal buffer as the *NAC-buffer*, which stores the current page and neighbor pages (Fig. 6).

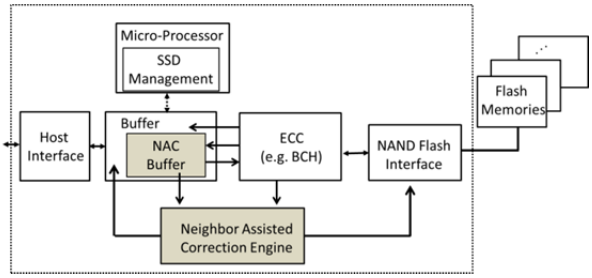

Fig. 6 Flash controller with Neighbor Assisted Error Correction.

Fig. 7 shows the flow of a page read request with NAC. When the controller starts a read operation, it first checks whether the requested page is already in the NAC-buffer (or in the internal flash buffer, as done in a state-of-the-art controller) before requesting the page from the flash memory. If so and if error-free, the page is returned to the requestor (e.g., the host machine). If not, the NAND flash interface fetches the requested page from flash memory and sends it to the ECC engine. If ECC successfully corrects all errors, flash controller sends the corrected page to the requestor, and also stores it in the NAC-buffer. If ECC fails, NAC is triggered for further correction. When NAC is triggered, it requires the direct-neighbor LSB and MSB pages above the requested page in order to classify the cells of the requested page. If these direct-neighbor pages are not in the NAC-buffer, the flash controller reads the neighbor pages out from flash memory. After ECC attempts to correct these neighbor pages, they are saved into the NAC-buffer, either error-free or erroneously (which is indicated by a bit for each page). When both neighbor pages are in the NAC-buffer, NAC is ready to correct the requested page. NAC iteratively applies a set of local optimum read reference voltages to correct 1/N-th of the requested page. ECC attempts to correct the page after every step and when ECC passes, NAC iterations stop and the requested page data is sent to the requestor. If ECC fails after all NAC iterations, the page cannot be corrected even with NAC and an exception is raised.

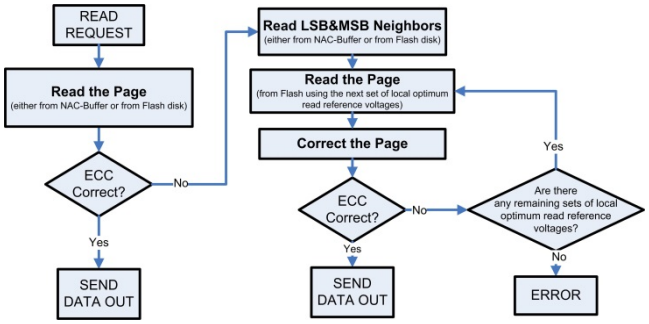

Fig. 7 Flow of the Neighbor Assisted Error Correction (NAC) mechanism

The size of the NAC-buffer is an important parameter. In order to determine the optimum size, we assume the worst-case scenario in which every read page requires NAC correction. It is common that pages are read from the flash memory in page number order due to common sequential access patterns and access to large data chunks. For instance, as in Fig. 1(a), when Page 3 fails and needs to be corrected by NAC, Pages 5 and 8 are saved to NAC-buffer. Pages 5 and 8 are also the neighbors of Page 6, hence, we prefer buffering those pages in anticipation that Page 6 will also be read due to sequential access. However, when Page 5 is read between Pages 3 and 6, two more neighbor pages are added to the NAC-buffer (i.e., Pages 7 and 10). Therefore, we set the size of the NAC-buffer to 5 pages to be able to buffer Pages 5, 6, 7, 8, 10 at the same time. When the buffer is full and a new page needs to be added, we evict the one with the lowest page number. We have

investigated performance sensitivity to NAC buffer size, management policies, and prefetching mechanisms; space limits prevent us from presenting these detailed evaluations. Our empirical results (not shown in this paper) indicate that a NAC-buffer size of 5 pages leads to minimal read latency.

6.2 Prioritized NAC

In this section, we make several observations that improve the effectiveness of NAC by allowing us to apply the N sets of local optimum read reference voltages in an order that minimizes the latency overhead of NAC. In other words, we observe that some sets of local optimum read reference voltages provide better error correction capability (as they are associated with neighbor cell values that are more likely to lead to errors) and we prioritize their use when NAC is triggered such that we can reduce the number of steps in NAC correction. We call this ***Prioritized NAC.***

Error Type Analysis: When a page is read with a single global optimum read reference voltage, REFx (see Fig. 5), the resulting errors can be classified into *eight* types. The errors for cells programmed to a lower-voltage state (P_i state) with direct-neighbor cell value 11, 10, 00 and 01, but misread as belonging to a higher-voltage state (P_{i+1} state) are denoted as $P_i(11)\rightarrow P_{i+1}$, $P_i(10)\rightarrow P_{i+1}$, $P_i(00)\rightarrow P_{i+1}$, $P_i(01)\rightarrow P_{i+1}$. The errors for cells programmed to a higher-voltage state (P_{i+1} state) with direct-neighbor cell value 11, 10, 00 and 01, but misread as belonging to a lower-voltage state (P_i state) are denoted as $P_{i+1}(11)\rightarrow P_i$, $P_{i+1}(10)\rightarrow P_i$, $P_{i+1}(00)\rightarrow P_i$ and $P_{i+1}(01)\rightarrow P_i$. The relative percentages of all eight types of errors are shown in Fig. 8 for 2Y-nm flash under various P/E cycles. We can see that $P_{i+1}(11)\rightarrow P_i$, $P_i(10)\rightarrow P_{i+1}$ and $P_i(01)\rightarrow P_{i+1}$ are the three dominant errors.

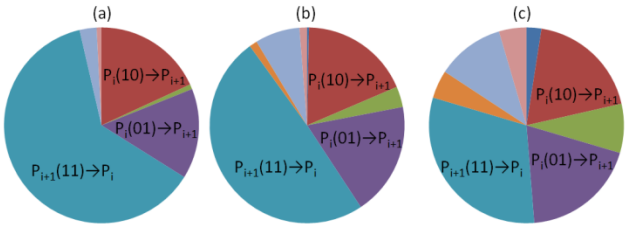

Fig. 8 Error type percentage for all eight types of read errors at (a) 3k (b) 10k and (c) 30k P/E cycles for 2Y-nm flash.

Some types of errors are more dominant because cells with threshold voltage near the border of the neighboring state's threshold voltage distribution tend to have errors as they are the most likely ones to be misclassified using a single reference voltage. To understand this, let us examine Fig. 5(a) and 5(b) that respectively plot threshold voltage distributions of victim cells before and after neighbor cells are programmed. Let us focus on two cells, each of which has an initial threshold voltage that is close to the border of the neighbor state's threshold voltage distribution: $P_{(i+1)}^{low}$ and $P_{(i)}^{high}$ in Fig. 5(a). If $P_{(i+1)}^{low}$'s direct-neighbor cell is programmed to 11, the resulting threshold voltage for this cell, $P'_{(i+1)}^{low}$, will be at the bottom end of the conditional distribution N11 for state $P_{(i+1)}$ (see Fig. 5(b)) because programming the neighbor cell with 11 (ER state) leads to the smallest amount of program interference. Unfortunately, using REFx as the read reference voltage might cause the value of this cell to be misread as belonging to state P_i, leading to the $P_{i+1}(11)\rightarrow P_i$ error type, the most common error type according to Fig. 8. Similarly, if $P_{(i)}^{high}$'s direct-neighbor cell is programmed to 01, the resulting threshold voltage for this cell, will be at the highest end of the conditional distribution N01 for state $P_{(i)}$ (see Fig. 5(b)) because programming the neighbor cell with 01 leads to the highest amount of program interference. Unfortunately, using REFx as the read reference voltage might cause the value of this cell to be misread as belonging to state P_{i+1}, leading to the $P_i(01)\rightarrow P_{i+1}$ error type, second most common error type according to Fig. 8. This example shows that two types of cells are more vulnerable to errors: 1) those that are at the bottom of the distribution of a higher-voltage state and receive the least amount of interference from their direct-neighbor, 2) those that are at the top of the distribution of a lower-voltage state and receive the highest amount of interference from their direct-neighbor. This leads to the dominance of corresponding error types.

Note that the $P_{i+1}(11)\rightarrow P_i$ type error is the most dominant one. This is because the program interference of aggressor cells with values 10 and 01 are close (N01 and N10 almost overlap in Fig. 5(b)). Thus, more cells are located at the top region of the overall distribution of each state. The global optimum read reference voltage therefore deviates toward the higher-voltage state. As a result, more cells of N11 in the higher-voltage state tend to be misread as belonging to the lower-voltage state.

Prioritized NAC: We take advantage of the above analysis to prioritize the selection of the set of local optimum read reference voltages during NAC. After a page that is read using the set of global optimum read reference voltages fails ECC, we select REF_{x11} as the first set of local optimum read reference voltages to read the failed page to fix the dominant $P_{i+1}(11)\rightarrow P_i$ errors. At this step of NAC, the read data of the cells with direct-neighbor values 11 will be replaced with the new reading performed with REF_{x11}. This leads to three cases: 1) The read data of a cell remains unchanged upon the NAC reading. This is the case for cells of type-N11 with threshold voltage larger than REFx or smaller than $REFx_{11}$. 2) The new data read using $REFx_{11}$ is correct and the old data read using REFx was incorrect. This is the case for cells of type-N11 in P_{i+1} state with threshold voltage in the window [$REFx_{11}$, REFx]. 3) The new data read using $REFx_{11}$ is incorrect and the old data read using REFx was correct. This is the case for cells of type-N11 in the P_i state with threshold voltage in the window [$REFx_{11}$, REFx]. Since $REFx_{11}$ is the optimum read reference voltage for cells of type-N11, the number of corrected cells (case 2) is larger than that of mis-corrected cells (case 3). Fig. 9 demonstrates this empirically: the net number of corrected cells (number of corrected minus the number of mis-corrected cells) of type-N11 can reduce the total errors by 58%, 44% and 22% for flash memory at 3k, 10k and 30k P/E cycles respectively. Since the read data of types N00, N01, and N10 are unchanged, the total number of errors is significantly reduced after applying NAC with $REFx_{11}$. ECC may be able to correct the remaining errors without requiring another NAC step. In such a case, system performance is improved due to Prioritized NAC. If ECC still fails after applying $REFx_{11}$, our mechanism tries $REFx_{10}$ next, and if ECC still fails again it applies $REFx_{01}$ as the error correction capability of these are the second and third highest, as Fig. 9 also demonstrates empirically.

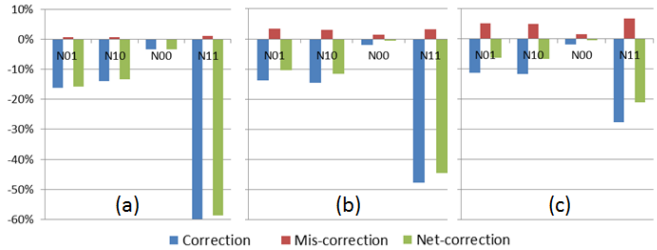

Fig. 9 Percentage of corrected errors, mis-corrected errors and net-corrected errors over total errors for flash memory when local optimum read reference voltages corresponding to N01, N10, N00 and N11 distributions are used at (a) 3k (b) 10k (c) 30k P/E cycles.

6.3 Policy When Neighbor Cells Have Errors

So far, we assumed that the neighbor cell values can be obtained correctly for NAC. However, the neighbor page may fail ECC and may not be correctly read. If the neighbor page fails ECC, we propose to use the uncorrected raw data from the neighbor page to classify cells for NAC. This section analyzes the effect of neighbor failure on the error correction capability of NAC.

Neighbor Page Failure Analysis: First, the neighbor page failure rate is low. Assume P_{fail} is the ECC failure rate. The probability of one of the two neighbor pages failing when the victim page fails is approximately $2(P_{fail})^2$, which is much smaller than P_{fail}. Even when neighbor pages fail ECC, most of the raw values read from neighbor pages are still correct because the raw BER is less than a few percent even at very high P/E cycles [1][2]. As a result, most cells in the page can be correctly classified using correct neighbor values, enabling NAC to correct errors in the victim page. For the neighbor cells that are misread, NAC may still be able fix the errors on the corresponding victim cells although it could also inject additional errors. We demonstrate this with an example. Consider a victim cell of type-N11 with its direct-neighbor cell programmed as 11 but misread as 10. The threshold voltage of this victim cell will be compared to the wrongly-selected local read reference voltage $REFx_{10}$ instead of $REFx_{11}$ during NAC. If the threshold voltage of this victim cell is less than $REFx_{11}$ or larger than $REFx_{10}$, the read value using the wrong reference voltage $REFx_{10}$ is the same as the value if read with the correct reference voltage $REFx_{11}$. Thus, there is no additional error introduced. Only when the threshold voltage of the victim cell is smaller than $REFx_{10}$ or larger than $REFx_{11}$, the read value could be different. For a cell like this, there are still two cases:

1. Cell is actually in state P_i: The correct local read reference voltage $REFx_{11}$ would have misread the cell as belonging to state P_{i+1} while the wrongly-selected $REFx_{10}$ correctly reads it as belonging to state P_i. Hence, the error in the neighbor cell eliminates a potential error when reading the victim cell.
2. Cell is actually in state P_{i+1}: The correct local read reference voltage $REFx_{11}$ would have correctly read the cell as belonging to state P_{i+1}. However, $REFx_{10}$ misreads it as belonging to state Pi. Hence, the error in the neighbor cell introduces a new error when reading the victim cell.

Since the threshold voltages of flash cells in different locations are independent of each other [4], the event that a neighbor cell has an error is independent from the event that its victim cell's threshold voltage falls in the range of [$REFx_{11}$, $REFx_{10}$]. Thus, the additional error rate caused by neighbor failure is equal to:

$$P_{neighbor}^{11->10} \times \left(\int_{REFx11}^{REFx10} f_{11}^{Pi+1}(x)dx - \int_{REFx11}^{REFx10} f_{11}^{Pi}(x)dx\right) \quad (21)$$

$P_{neighbor}^{11->10}$ is the probability that the cell on the neighbor wordline was programmed as 11 but misread as 10. $f_{11}^{Pi}(x)$ and $f_{11}^{Pi+1}(x)$ are the PDF functions of the threshold voltages of type-N11 cells programmed in states P_i and P_{i+1} respectively. $P_{neighbor}^{11->10}$ is less than a few percent at high P/E cycles. $\int_{REFx11}^{REFx10} f_{11}^{Pi+1}(x)dx - \int_{REFx11}^{REFx10} f_{11}^{Pi}(x)dx$ is much smaller than 1 since the region of [$REFx_{11}$, $REFx_{10}$] is far away from the center of the distribution $f_{11}^{Pi+1}(x)$ and $f_{11}^{Pi}(x)$ (see Fig. 5(b)). Similarly to the case of neighbor 11 being misread as 10, the additionally injected errors caused by the other types of neighbor cell failures are also very small.

Measurement of Effect on NAC Error Correction Rate: Fig. 10 plots the fraction of additional errors caused by neighbor failure over the errors that are corrected by NAC under various P/E cycles as measured for 2Y-nm flash devices. We can see that the number of errors corrected by NAC is much larger than the additionally injected errors caused by neighbor failure. The fraction of injected errors due to neighbor failure increases with P/E cycles because the raw error rate of neighbor pages increases with P/E cycles and therefore the probability of errors due to misclassification of victim cells increases. However, we can see that even at 35k P/E cycles, the number of additional errors due to neighbor failures is less than 10% of the errors that are corrected by NAC. We conclude that neighbor cell errors do not significantly reduce the error correction capability of NAC.

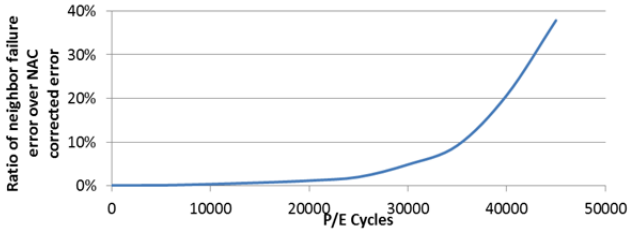

Fig. 10 The fraction of errors due to neighbor cell read failure over total errors that are corrected by NAC.

6.4 Implementation of NAC

Optimum Read Reference Voltage Learning: NAC needs to read the page multiple times using the global optimum and up to N local optimum read reference voltages. The optimum reference voltage is the average of the mean values of the threshold voltage distributions of neighboring states. To learn the mean values of state voltage distributions, we propose to periodically (e.g., every 100 P/E cycles) measure the overall and conditional threshold voltage distribution statistics for a set of sampled wordlines. The details of this periodic sampling based learning mechanism are straightforward and omitted for brevity. The learning overhead is kept low: even for write-intensive applications (e.g., 10 full disk writes per day) [7], 100 P/E cycles will be consumed after at least 10 days. The learning task runs in background with low priority so that it does not interfere with normal operation.

NAC Microarchitecture: Fig. 11 shows the microarchitecture of NAC. It contains four buffers, one comparator (Comp) vector and one pass circuit vector. The neighbor LSB page buffer and MSB page buffer store the neighbor LSB/MSB page data of the current failed page respectively. The *Page-to-be-Corrected* buffer initially contains the data of the page that was read with the global optimum read reference voltage and failed ECC. The *Local-Optimum-Read* buffer is loaded with the page data that is read using the local optimum read reference voltage at each step of NAC. Two bits, Bit1/Bit2, indicate which of the four local optimum read reference voltages (corresponding to four possible values in the neighbor cells) is used to read the current data in the Local-Optimum-Read buffer. In each step of NAC, after the page is read using the local optimum read reference voltage, each comparator circuit compares Bit1/Bit2 to the corresponding MSB/LSB neighbor bits. If there is a match (indicating that the neighbor cell value is the same as the value for which the local optimum read reference voltage is selected), the corresponding pass circuit is enabled, which causes the corresponding data in the *Local-Optimum-Read* buffer to overwrite the corresponding cell in the *Page-to-be-Corrected* buffer, thereby enabling the NAC correction of the cell that has a neighbor value that matches Bit1/Bit2. All other circuitry supporting the NAC flow in Fig. 7 is not shown.

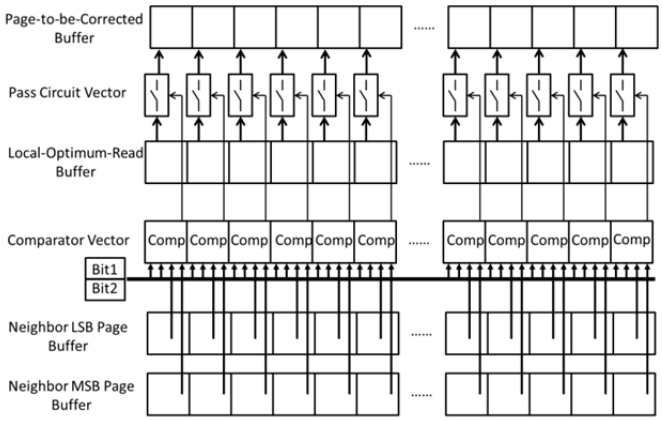

Fig. 11 Microarchitecture of Neighbor Assisted Error Correction.

7. EVALUATION RESULTS

7.1 P/E Cycle Lifetime Evaluation

Since ECCs that can correct 40 bit errors per 1k-Byte data are recommended for 20nm MLC flash memory [17], we chose such ECC as baseline in our evaluations. Following the methodology of [2], the probability of ECC failure can be calculated as:

$$P_{CW} = \sum_{n=E+1}^{N} \binom{N}{n} RBER^{n} \times (1-RBER)^{N-n} \quad (22)$$

Here, N is the number of bits protected by ECC and E is the maximum number of errors that can be corrected. RBER is the raw bit error rate before error correction. Raw BER and ECC failure rate are listed in Table 2. When RBER is larger than $10^{-2.64}$, ECC almost always fails. Also, to guarantee ECC failure rate to be below 10^{-15}, the acceptable RBER should be less than 10^{-3}.

Table 2. Correlation of raw BER and ECC failure

Raw BER	10^{-3}	$10^{-2.9}$	$10^{-2.8}$	$10^{-2.7}$	$10^{-2.65}$	$10^{-2.64}$
ECC FER	10^{-15}	10^{-10}	10^{-6}	10^{-2}	0.07	100%

P/E Cycle Lifetime Evaluation: We program random data into 2Y-nm 2-bit MLC flash memory for up to 50k P/E cycles. Assuming that retention errors can be fixed by using flash refresh techniques [6][7], the programmed data are read out after one-week retention at room temperature. The raw BER values resulting from reading with the global optimum read reference voltage without NAC and reading with different strengths of NAC using respectively one, two, three or four potential sets of local optimum read reference voltages are shown in Fig. 12 over different P/E cycles. To guarantee system reliability, the raw BER must be less than the acceptable raw BER (i.e., 10^{-3}) of the baseline ECC. Thus, the maximum P/E cycle lifetime of the baseline flash memory without NAC is only 18k P/E cycles, as shown in Fig. 12. NAC increases the P/E cycle lifetime by 22% (22k P/E cycles), 33% (24k P/E cycles) and 39% (25k P/E cycles) respectively for different strengths. Since the fourth set of local optimum read reference voltages mainly corrects errors for the conditional distribution that is in the middle of the overall distribution where there are fewer errors (see Fig. 8), its raw BER curve almost overlaps with that of using three sets of local optimum read reference voltages. We conclude that NAC is effective in improving flash memory lifetime and as NAC strength is increased lifetime improvement increases.

NAC Frequency Analysis: As Fig. 12 shows, the extended lifetime due to NAC is divided into three regions based on NAC strength: stage 1, stage 2, and stage 3. At the beginning of the extended lifetime, the raw BER is about 10^{-3} and the ECC failure rate before NAC is 10^{-14}. NAC is seldom triggered. As P/E cycles increase, the raw BER after reading with global optimum read reference voltage increases. Raw BERs without NAC at the end of stage-1, stage-2 and stage-3 are $1.6\text{x}10^{-3}$, $2\text{x}10^{-3}$ and $2.2\text{x}10^{-3}$ respectively. These cause the ECC failure rates without NAC to be 10^{-5}, 10^{-2} and 33% respectively. This means that ECC does not *always* fail during the extended lifetime and hence NAC is not triggered for all requested pages. We use the ECC failure rate without NAC at the end of each stage to estimate the worst-case NAC trigger frequency in that stage. Note that the ECC failure rate within a stage is generally lower than this worst-case value as it increases with P/E cycles. In stage-1, NAC trigger frequency is very low ($<10^{-5}$). However, the ECC failure rate would not satisfy modern storage systems' requirement (less than 10^{-15} failure rate) without NAC protection. In stage-2, NAC is triggered at a rate <1%. Only in the third region, NAC is triggered often (<33%). However, if NAC is applied, the raw BER of all these stages can be reduced to lower than 10^{-3}. Hence, using NAC guarantees the uncorrectable failure rate to be below 10^{-15}, which is the error rate requirement for storage.

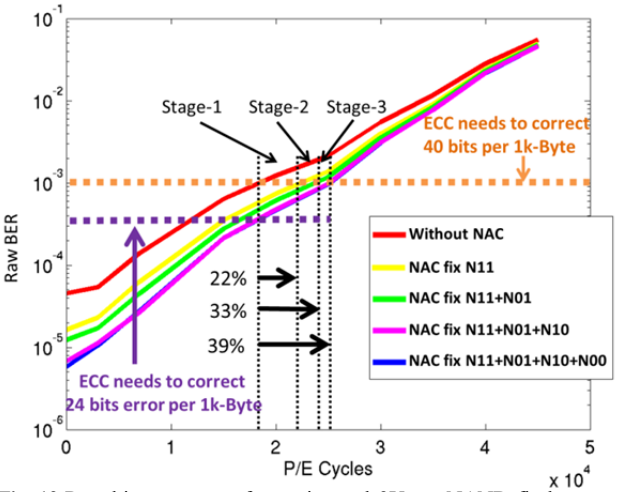

Fig. 12 Raw bit error rate of experimental 2Y-nm NAND flash memory with and without NAC of varying levels of strength.

Reducing the Cost of ECC using NAC: If P/E-cycle lifetime that is desired is the same as that of the baseline without NAC, the employed ECC mechanism can be simplified in the presence of NAC. We explore how much NAC can help reduce ECC cost to achieve the same P/E-cycle lifetime as the baseline ECC mechanism. The maximum P/E cycle lifetime of the employed baseline ECC is 18K P/E cycles, as seen in Fig. 12 (at this lifetime, the raw BER of the baseline reaches 10^{-3}, which is the maximum acceptable raw BER that leads to an ECC failure rate of 10^{-15}, as shown in Table 2). At 18K P/E cycles, using NAC reduces the acceptable raw BER for ECC by 65% from 10^{-3} to $3.5\text{x}10^{-4}$. To satisfy this raw BER, simpler ECC that can correct only 24 bit errors (as opposed to 40 in the baseline) for 1k-Byte data can be used. Since ECC complexity increases linearly with the number of errors that ECC can correct [12], the ECC design cost can be reduced by approximately 40% when NAC is employed. This can lead to significant power and area savings in the system.

P/E Cycle Analysis: NAC provides diminishing returns on raw BER rate as the number of P/E cycles increases. The raw BER reduction with NAC can be more than 90% at the beginning of flash lifetime. However, at high P/E cycles (e.g., 35k P/E cycles), NAC reduces raw BER by only 30%. This is mainly due to two reasons. First, at low P/E cycles program interference related

errors, which are the errors that can be fixed by NAC, constitute a higher fraction of all flash errors, whereas at higher P/E cycles they constitute a smaller fraction [3] and other errors, e.g., those due to P/E cycling noise, which cannot be fixed by NAC, become dominant [1][4]. Note that this is because program interference is due to cell-to-cell coupling and is mainly determined by the geometry of flash cells (e.g., the distance between neighboring flash cells) instead of P/E cycles whereas many other flash error types increase with P/E cycles [3]. Second, the ECC failure rate of, and correspondingly the number of, erroneously read cells on the neighbor pages, which are needed by NAC for error correction, increases over P/E cycles. Neighbor page ECC failure causes misclassification of cells and can partly counteract the errors corrected by NAC (Sec. 5.3). When flash memory scales to smaller geometries and program interference becomes more dominant, we would expect NAC to have even higher benefits in error reduction and relative lifetime improvement.

7.2 Performance Evaluation

We evaluate the performance of NAC by using Disksim [18][19] with SSD extensions [19]. We use I/O traces from various workloads: cello99 [20], postmark [21], MS-Cambridge [22], Financial OLTP [23] and Web Search Engine [23]. Details of these traces can be found in [7]. We configure the simulated flash based SSD with four channels. Each channel has eight flash chips. Each flash chip has 8096 blocks containing 256 pages per block. Each page is 8KB unless specified otherwise.

7.2.1 Workload NAC-Buffer Locality Analysis

To help explain the performance impact of NAC in Sec. 7.2.2, we first analyze the locality behavior of read operations in the workloads in the NAC-buffer. NAC presents time overhead for reading the neighbor LSB and MSB pages of a requested page (called the victim page) that fails ECC. NAC keeps these pages in a 5-entry NAC-buffer, as described in Sec. 6.1. If the victim and neighbor pages are needed while they are in the NAC-buffer, these requests are served from the NAC-buffer instead of flash memory.

We first explore hit rates in a 5-entry NAC-buffer. We study a system that always keeps the 5 last accessed pages in the NAC-buffer (but the system is error-free for the purposes of this study). In Fig. 13, we show the hit rate of different types of pages in the NAC-buffer. The victim page hit rate shows the caching effect of the NAC-buffer: this is the fraction of all flash requests that hit in the 5-entry NAC-buffer. The higher this fraction, the more effective the NAC-buffer as a latency reduction mechanism for *all* requests in the system. The LSB/MSB neighbor page hit rate specifies the fraction of all flash requests for which the LSB/MSB neighbor is also in the NAC buffer when the request is received. The higher this fraction, the less the overhead of a potential NAC correction as there is no need to fetch the LSB/MSB neighbor from flash memory to perform NAC correction when these neighbors are already in the NAC-buffer. For workloads with heavily sequential access patterns, we expect the hit rates of the neighbor pages to be higher since the program may have already fetched neighbor pages before it requests a (victim) page.

We make several observations from Fig. 13. First, the NAC-buffer hit rate is in general low (i.e., less than 5%), which indicates that only a small fraction of pages are accessed repeatedly from the small NAC buffer. This is because, in order to reduce disk accesses, the operating system keeps the recently used pages in main memory and it is unlikely that a page that is recently accessed will be evicted from main memory to be accessed again soon after. However, *Financial* has a remarkably high hit rate in the small NAC-buffer, which affects its performance positively with NAC (see Sec. 7.2.2). Second, we find that the LSB/MSB Neighbor Page hit rates are high for *Cello*, *MS-Cambridge* and *Web-Search* workloads, as these have more than 30% of their read operations as sequential requests. In contrast, *Financial* and *Postmark* workloads have relatively small request sizes; therefore, they feature less sequential read requests, leading to lower hit rates. We expect the overhead of NAC correction to be low for the former type of (sequential-access) workloads. Third, the hit rate of the LSB neighbors is higher than that of the MSB neighbors. This is because the number of pages between the victim page and its MSB neighbor is higher than the number of pages between the victim page and its LSB neighbor. As a result, workloads that only have higher levels of spatial locality, of which there are fewer, can take advantage of the MSB neighbors.

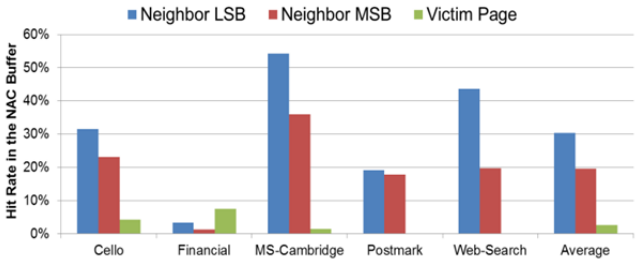

Fig. 13 The hit rate of different page types in the NAC-Buffer during error-free Execution (i.e., when ECC fail rate is zero). Victim Page indicates the overall hit rate of the NAC-buffer for all flash accesses.

7.2.2 NAC Performance Overhead Analysis

Fig. 14 shows the increase in read latency due to NAC at distinct P/E cycles. We activate the NAC mechanism when the first ECC failure is experienced in flash memory. After that, each requested page is first searched in the NAC-buffer before accessing the flash disk. Note that the time spent for this search operation is around 1% of the latency of flash disk access latency. We make three major observations.

First, NAC does not present any performance impact when the flash memory is within the same lifetime as the baseline (<18K P/E cycles).

Second, at relatively low P/E cycles after the extended lifetime, NAC presents either only negligible performance degradation or slight performance improvement. This is because, for especially the workloads with good locality, the 5-entry NAC-buffer behaves as an effective small cache in front of the flash memory, as indicated by the hit ratio of different types of pages in Fig. 13. Even when NAC incurs overhead by fetching the neighbor pages upon an ECC failure from flash, the later requests to these pages made by the workload can hit in the NAC-buffer, thereby potentially hiding the NAC overhead for these neighbor pages.

Third, NAC incurs less than 5% performance degradation while providing a 33% lifetime improvement (i.e., from 18k to 24k P/E cycles). Moreover, this performance degradation is only introduced for the higher P/E cycles during which a normal flash memory without NAC is considered to be non-functional.

Finally, there is a sharp increase in read latency overhead between 24k P/E cycles and 25k P/E cycles. The main reason of this increase is the drastic increase in ECC failure rate from 10^{-2} to 33% between these two points. At 25k P/E cycles, one out of every 3 read operations requires NAC correction with its associated overhead. However, using NAC still enables a high lifetime of 25K P/E cycles, which cannot be achieved by the baseline ECC.

We conclude that NAC is effective at improving flash memory lifetime beyond ECC without significantly degrading system performance (up until the point where ECC failure rate increases drastically).

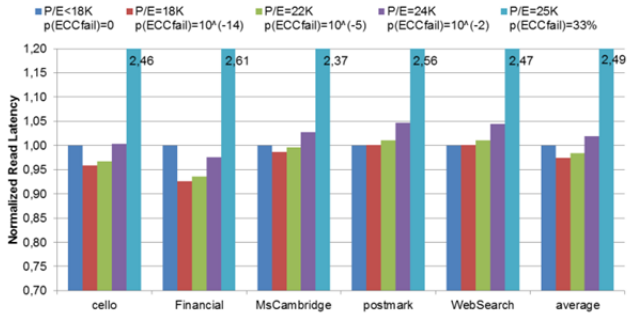

Fig. 14 The read latency overhead of NAC at different P/E cycles with different ECC failure rates in each P/E cycle.

Effect on a System with Prefetching: Fig. 15 shows the read latency overhead of NAC on a system that prefetches 2 or 4 consecutive pages for every read request. We increase the bandwidth between the flash controller and the flash disk so that the prefetched pages can be read to the flash controller in parallel with the requested pages. When the number of pages prefetched is increased, NAC presents less performance degradation at high ECC failure rate, i.e., when p(ECCfail)=33%. This is because a more aggressive prefetcher increases the probability of finding the neighbor pages in the NAC-buffer when NAC needs them. However, at low ECC failure rate, NAC presents more performance degradation when number of pages prefetched is higher. This is because prefetching aggressively consumes NAC-buffer entries and reduces the overall hit rate of the victim pages. We conclude that NAC has even less performance overhead in a state-of-the-art system that employs aggressive prefetching.

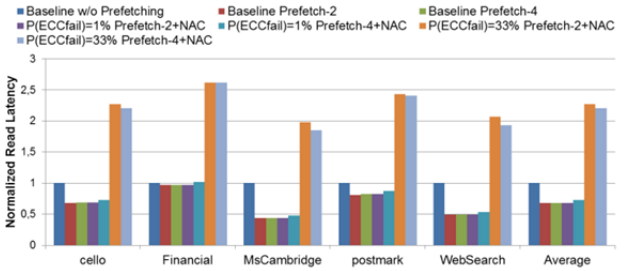

Fig. 15 The performance overhead of NAC in a system with prefetching. Each bar is normalized to the baseline, error-free system that does not employ prefetching.

8. CONCLUSION

We comprehensively analyzed threshold voltage distributions in state-of-the-art 2Y-nm MLC NAND flash memory and both statistically and experimentally demonstrated that bit error rate when reading flash memory can be decreased by classifying flash cells based on their immediately-neighboring cells' values. Building on these analyses, we introduced the first low-overhead and high-accuracy neighbor-cell assisted error correction methods that leverage information from neighbor cells to correct errors in cells that are being read from flash memory. Our experimental evaluations using I/O traces from real workloads and error data obtained from real flash memory chips show that our new error correction methods can significantly reduce the raw bit error rate and improve flash memory lifetime at zero or very modest performance overheads. As flash memory scales down to smaller technology nodes and cell-to-cell interference therefore becomes an even more dominant cause of errors, we expect that the error correction techniques proposed in this paper will become even more important for reliable operation. We also hope that the statistical and experimental analyses provided in this paper can enable the development of even more sophisticated and effective error tolerance mechanisms.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for helpful feedback. This work was partially supported by the Intel Science and Technology Center for Cloud Computing, Data Storage Systems Center (DSSC) at Carnegie Mellon University, NSF Award CCF 12122962, the FP7 ParaDIME Project, grant agreement no. 318693 and by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contracts TIN2008-02055-E and TIN2012-34557.

REFERENCES

[1] Y. Cai et al., "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization and Analysis", DATE 2012.
[2] N. Mielke et al., "Bit Error Rate in NAND Flash Memories", IRPS 2008.
[3] Y. Cai et al., "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation", ICCD 2013.
[4] Y. Cai et al. "Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling", DATE 2013.
[5] 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.
[6] R. Liu et al., "Optimizing NAND Flash-Based SSDs via Retention Relaxation", FAST 2012.
[7] Y. Cai et al. "Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime", ICCD 2012.
[8] Y. Cai et al., "Error Analysis and Retention-Aware Error Management for NAND flash memory," Intel Technology Journal 2013.
[9] H. Shim et al., "Highly Reliable 26nm 64Gb MLC E2NAND Flash Memory with MSP Controller", VLSIT, 2011.
[10] T. Kim et al., "Cell-to-cell Interference Compensation Schemes Using Reduced Symbol Pattern of Interfering Cells for MLC NAND Flash Memory", IEEE Transactions on Magnetics 2013.
[11] G. Dong et al., "Using Data Postcompensation and Predistortion to Tolerate Cell-to-cell Interference in MLC NAND Flash Memory", IEEE Transactions on Circuits and Systems I, 2010.
[12] S. Lin et al., "Error Control Coding (2nd Edition)", Prentice Hall 2004.
[13] Y. Cai et al. "FPGA-Based Solid-State Drive Prototyping Platform", FCCM 2011.
[14] J. Cha et al., "Data Randomization Scheme for Endurance Enhancement and Interference Mitigation of Multilevel Flash Memory Devices", ETRI Journal, 2013.
[15] C. Kim et al., "A 21 nm High Performance 64 Gb MLC NAND Flash Memory with 400 MB/s Asynchronous Toggle DDR Interface", JSSC 2012.
[16] D. Lee et al., "Estimation of NAND Flash Memory Threshold Voltage Distribution for Optimum Soft-Decision Error Correction", IEEE Transactions on Signal Processing 2013.
[17] S. Yasarapu, "Architectural Requirements for MLC based SSDs", FMS 2011.
[18] J. Bucy et al., "The DiskSim Simulation Environment Version 4.0 Reference Manual", Technical Report 2008.
[19] N. Agrawal et al., "Design Tradeoffs for SSD Performance", USENIX 2008.
[20] Open Source at HP Labs, http://tesla.hpl.hp.com/opensource
[21] J. Katcher, "Postmark: a New File System Benchmark", Technical Report, 1997.
[22] SNIA: IOTTA Repository, http://iotta.snia.org/tracetypes/3.
[23] UMass Trace: http://traces.cs.umass.edu/index.php/Storage/Storage