Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory

Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, Onur Mutlu

***Abstract*—NAND flash memory is a widely-used storage medium that can be treated as a *noisy* channel. Each flash memory cell stores data as the *threshold voltage* of a floating gate transistor. The threshold voltage can shift as a result of various types of circuit-level noise, introducing errors when data is read from the channel and ultimately reducing flash lifetime. An accurate model of the threshold voltage distribution across flash cells can enable mechanisms within the flash controller that improve channel reliability and device lifetime. Unfortunately, existing threshold voltage distribution models are either not accurate enough or have high computational complexity, which makes them unsuitable for online implementation within the controller.**

We propose a new, low-complexity flash memory model, built upon a modified version of the Student's t-distribution and the power law, that captures the threshold voltage distribution and *predicts future distribution shifts* as wear increases. Using our experimental characterization of state-of-the-art 1X-nm (i.e., 15–19nm) MLC NAND flash chips, we show that our model is highly accurate (with an average modeling error of 0.68%), and also simple to compute within the flash controller (requiring 4.41x less computation time than the most accurate prior model, with negligible decrease in accuracy). Our model also predicts future threshold voltage distribution shifts with a 2.72% modeling error.

We demonstrate several example applications of our model in the flash controller, which improve flash channel reliability significantly, including a new mechanism to predict the remaining lifetime of a flash device. Our evaluations for two of these applications show that our model (1) helps improve flash memory lifetime by 48.9%, and/or (2) enables the flash device to safely sustain 69.9% more write operations than manufacturer specifications. We hope and believe that the analyses and models developed in this paper can inspire other novel approaches to flash memory reliability and modeling.

1. Introduction

NAND flash memory is widely used today in storage devices, as it provides long-term data integrity guarantees at much higher performance than magnetic hard disks. Data in NAND flash is stored within an array of *flash cells*. Each flash cell is made of a *floating gate transistor*, which stores charge within a *floating gate*. The amount of charge in the floating gate determines the *threshold voltage* of the transistor (i.e., the voltage at which the transistor turns on).

The threshold voltage value is used to represent the data that is stored within a flash cell. The threshold voltage can be programmed to a voltage level within a fixed range. This range is divided up into *voltage windows*. Each window represents a certain binary data value. Older-generation flash devices use *single-level cells* (SLC), where each cell represents a single bit of data. SLC splits the threshold voltage range into two voltage windows: one of the windows represents the data value 0, and the other window represents the data value 1. To enable higher storage densities, many modern flash devices use *multi-level cells* (MLC), where each cell represents *two* bits of data. To do this, the threshold voltage range is split into four smaller windows, one each for the data values 00, 01, 10, and 11. When data is read from a multi-bit flash memory channel, a *read reference voltage* is applied to the control gate of the transistor of each cell being read, in order to sense the threshold voltage of the floating gate within the cell. If the threshold voltage is greater than the read reference voltage, the cell turns on; otherwise, the cell remains off. The flash controller identifies which voltage window a cell belongs to by choosing read reference voltages that fall at the boundaries between voltage windows. Due to variation across flash cells, multiple cells within the same flash memory device that are programmed to the same data value may have different voltage levels (within the voltage window), resulting in a *distribution* of threshold voltages for each data value.

Unfortunately, even when the data stored within a flash cell is *not* modified, the threshold voltage of the cell changes (i.e., shifts) over time. This shift is a result of many different types of circuit-level noise, such as program/erase (or P/E) cycling noise [2,8,26,35], data retention noise [2,3,5,6,26], cell-to-cell program interference noise [2,4,7], read disturbance noise [9, 26], and read noise [2]. Some cells' threshold voltages might shift enough to cross over to neighboring voltage windows. The value of such a cell would be misread on a flash memory channel read, causing an error. If the misread value is *not* correctable by the error-correcting code (ECC) mechanism of the flash controller, either the flash device fails or the data is silently corrupted.

Having online information on the current threshold voltage distribution across all of the flash cells within a flash memory chip (i.e., the *static* distribution), as well as how this distribution changes over time (i.e., the *dynamic* distribution), is important to quantify errors and develop techniques to improve the reliability of the flash device. First, the static distribution can be used to determine the number of errors that would occur for any read reference voltage that is applied. This data can be used by the flash controller to select the read reference voltage that minimizes the error rate. Lowering the error rate increases the lifetime of the flash device, as it delays the time at which the number of errors becomes too large for the built-in ECC mechanism to successfully correct. Second, knowing how the

dynamic distribution changes over time (i.e., as more writes are performed) is important, as it can guide flash controller mechanisms that adjust various flash parameters online (e.g., ECC strength [14, 23, 42], read reference voltages [3], pass-through voltage [9]) to increase the flash memory lifetime. Prior proposals to adjust these parameters (e.g., [3,9]) rely on a trial-and-error approach to select parameters with low error rates, which can be inaccurate, high-latency, and suboptimal in terms of lifetime improvement. In both cases (static and dynamic), the threshold voltage distribution must be determined at runtime by the flash controller. Therefore, it is critical to design a *practical* and *low-complexity* mechanism to determine the distribution and the shifts in the distribution.

There is no practical way to know the *exact* threshold voltage distribution of *all* the cells within a chip, as this requires testing and recording the voltage of *each cell* within the device, which is extremely time-consuming and requires significant storage. Therefore, accurate *models* of this threshold voltage distribution are necessary. In addition to being *accurate*, the models must be *easy to compute*, as they need to be implemented within the flash controller. Prior work models the voltage distribution as a Gaussian distribution [8], but this model has been found to be relatively inaccurate for modern 1X-nm (i.e., 15-19nm) flash memory chips [35]. In contrast, more recent work proposes to use a normal-Laplace distribution to model the threshold voltage distribution more accurately [35], but this model is computationally expensive, taking 10.7x the computation time of the Gaussian model. **Our goal** in this work is to build a new model of threshold voltage distribution that is both *highly accurate* and *computationally efficient*, so that the model can be practically implemented within a flash device to drive various reliability mechanisms. This model must include both *static modeling* of the *current* threshold voltage distribution, to enable accurate reading of data, and *dynamic modeling* of the distribution, to adapt flash mechanisms to predicted distribution changes.

To build our model, we perform an experimental characterization of the threshold voltage distribution on real state-of-the-art 1X-nm (i.e., 15-19nm) MLC NAND flash chips. This characterization is essential to verify that the model we develop accurately captures the behavior of a real, modern device. We make several **key observations** from our characterization. First, we find that the threshold voltage distribution within MLC NAND flash memory has a fatter tail than a Gaussian distribution, rendering the Gaussian-based model significantly inaccurate. Second, we observe that this fat tail is not smooth, and produces a second, smaller peak in the distribution, requiring a model that is based on a multi-peak distribution. Third, we find that as the number of *program/erase (P/E) cycles* (i.e., writes) increases, the threshold voltage distribution shifts in an easy-to-predict manner governed by the power law, enabling us to extend our model to include a simple dynamic component to accurately capture this change.

Based on these observations, we propose a new channel model for the threshold voltage distribution in MLC NAND flash memory chips, which is both highly accurate and practical to implement. We first develop a *static model*, which captures the current threshold voltage distribution at a particular P/E cycle count, based on our modified version of the Student's t-distribution [38]. Our static model achieves a 0.68% average modeling error, which is nearly identical to the accuracy of the normal-Laplace model [35], while requiring 4.41x less computation time for the model. We then extend this into a *dynamic model*, which uses the power law to track how the static model changes with respect to the P/E cycle count. The dynamic model achieves 2.72% modeling error when predicting the threshold voltage distribution at 20K P/E cycles, which the dynamic model extrapolates using only distribution data sampled by invoking the static model between 2.5K and 10K P/E cycles.

We show several *example applications* of our model that improve flash memory reliability. We use the static model to (1) estimate the current raw bit error rate; and (2) estimate the *optimal* read reference voltage (i.e., the voltage at which the read error rate is minimized), showing that it improves flash memory lifetime by 48.9% over using the default read reference voltages. We use the dynamic model to implement a new mechanism that estimates the remaining lifetime of a flash device by predicting the growth in noise (i.e., error rate). This mechanism enables the device to be used safely for much longer than the overly-conservative manufacturer specification, increasing the endurance of the flash device by 69.9%. We also discuss how the dynamic model can be used to estimate the inputs to a sophisticated LDPC (low density parity check) decoder for better error correction, and how our model can be used to improve the performance of flash devices. We conclude that our threshold voltage distribution model can be used for a wide range of purposes, and we expect future work to develop and evaluate several other novel applications of our model that improve flash reliability and performance. We also hope and expect to inspire new online flash channel models that improve upon the model we developed to efficiently provide even higher accuracy.

In this work, we make the following key contributions:

- We provide an experimental characterization of the threshold voltage distribution, and how the distribution changes with wear, for state-of-the-art 1X-nm MLC NAND flash memory chips. Like prior work, we find that program errors can cause the tail of the distribution to fatten significantly, but, unlike prior work, we observe that this fat tail can show up much earlier in the lifetime of the flash device than previously thought.
- Using the findings of our characterization of the threshold voltage distribution, we propose a new, simple and accurate static model for the threshold voltage distribution of MLC NAND flash memory at a particular program/erase (P/E) cycle count, based upon our modified version of the Student's t-distribution. The model is capable of accurately capturing the threshold voltage distribution, with a 0.68% average modeling error, while requiring little computation in the flash controller.
- We propose a new model to dynamically estimate how the threshold voltage distribution shifts as a function of P/E cycles. This model works in conjunction with our proposed static model, and it accurately predicts how the

threshold voltage distribution changes in the future, with an average modeling error of 2.72%.

- We demonstrate several practical uses of our online threshold voltage distribution model in a flash controller, which allows the flash controller to dynamically adapt to threshold voltage shifts and thereby better improve flash memory reliability. We propose a new mechanism to estimate the actual remaining flash lifetime, based on the expected growth in bit error rate. Our mechanisms improve flash memory lifetime by 48.9% and/or enable the flash device to safely endure 69.9% more P/E cycles than the manufacturer specification.

2. Background

In this section, we introduce necessary background information on multi-level cell (MLC) NAND flash memory. Flash memory can be thought of as a noisy multi-bit channel, where each read performed on the channel fetches data from a portion of the flash cell array. We first discuss how data is stored within a flash memory device, then discuss how circuit-level noise can shift these data values. Afterwards, we go over the basic organization of a NAND flash memory device, and discuss read and write operations.

2.1. Threshold Voltage Distribution

NAND flash memory stores data as the threshold voltage of each flash cell, which is made up of a *floating gate transistor*. For a *multi-level cell* (MLC) NAND flash memory, each flash cell stores a two-bit value, and can be programmed to one of four threshold voltage states, which we call the ER, P1, P2, and P3 states. Each state represents a different two-bit value, and is assigned a *voltage window* within the range of all possible threshold voltages. Due to variation across program operations, the threshold voltage of flash cells programmed to the same state is initially distributed across this voltage window.

Figure 1 illustrates the threshold voltage distribution of an MLC NAND flash memory. The x-axis shows the threshold voltage (V_{th}), which spans a certain voltage range. The y-axis shows the probability density at each voltage level across flash memory cells. The threshold voltage distribution of each threshold voltage state can be represented as a probability density curve that spans over the state's voltage window. We label the distribution curve for each state with the name of the state and the corresponding two-bit value. We break down the two-bit values into the most significant bit (MSB) and least significant bit (LSB). The boundaries between neighboring threshold voltage windows are referred to as *read reference voltages*, labeled V_a, V_b, and V_c in Figure 1. These voltages are used by the flash controller to identify the voltage window (i.e., state) of each cell.

2.2. NAND Flash Memory Noise

Over time, the threshold voltage distribution shifts, as shown in Figure 2, and can *overlap* with each other. Shifts happen due to various types of noise in flash memory [2–9, 26, 35]. If the threshold voltage of a flash cell shifts into the voltage

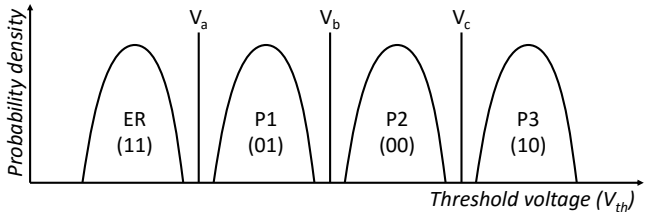


Fig. 1: Illustration of threshold voltage distribution of an MLC NAND flash memory chip.

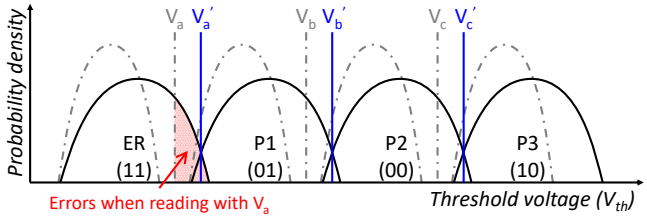


Fig. 2: The threshold voltage distribution after the voltages shift due to noise, resulting in overlapping distributions for each state.

window of a state other than the one the cell was originally programmed in, the cell can be *misread*, leading to a flash *error*. For example, a cell in the rightmost part of the ER state distribution in Figure 2 will be misread as being in the P1 state. Making matters worse, the threshold voltage shifts not only cause neighboring state distributions to overlap, but also shift the actual boundary between two windows away from the original read reference voltages (V_a, V_b, V_c) to new ones (V_a', V_b', V_c' in Figure 2). As such, the raw bit error rate increases as a result of these threshold voltage shifts.

Today's flash controllers deploy strong error-correcting codes (ECC) to detect and correct any raw bit errors [15, 22, 44], thus allowing user data to be stored and retrieved correctly during a channel read. Flash memory *wears out* as more writes (i.e., *program/erase*, or *P/E*, *cycles*) are performed. As flash memory wears out, P/E cycling noise becomes stronger, increasing the magnitude of the threshold voltage shift, as well as the raw bit error rate resulting from that shift [8]. On the other hand, even a strong ECC mechanism can correct only a limited number of raw bit errors within each code word [5]. At some point in the device's lifetime, raw bit errors increase beyond this limit, resulting in an uncorrectable error on the channel. Thus, for ECC to guarantee an acceptable uncorrectable error rate (e.g., 10^{-15} in the JEDEC standard [16]), the raw bit error rate (RBER) must be less than a certain noise threshold (e.g., 10^{-3} [13]), assuming NAND flash is a symmetric memoryless channel (i.e., assuming a uniform random bit flipping probability). Based on the RBER threshold, flash vendors specify a maximum limit to the number of P/E cycles that each flash page can endure. This limit is usually between 3000–10000 cycles in modern consumer MLC NAND flash memory [17].

2.3. NAND Flash Memory Organization

In each MLC flash memory chip, two-bit flash cells are organized as multiple two-dimensional arrays known as *flash*

blocks. Within each flash block, the MSBs of each row of flash cells are logically combined to form an *MSB page*, and the LSBs of each row are logically combined to form an *LSB page*. Each flash block contains 256–512 flash pages, which are typically 8–16KB in size. NAND flash memory supports three basic operations: read, program, and erase. Read and program operations are performed at the page granularity. Erase operations are performed at the block granularity. We provide the basics of these operations next.

2.4. Read Operation

Data can be read from NAND flash memory by applying read reference voltages onto the control gate of each cell, to sense the cell's threshold voltage. To read the LSB of the cell, we only need to distinguish the states with an LSB value of 1 (ER and P1) from those with an LSB value of 0 (P2 and P3). As Figure 1 shows, we need to use only one read reference voltage, V_b, to read the LSB page. To read the MSB page, we need to distinguish the states with an MSB value of 1 (ER and P3) from those with an MSB value of 0 (P1 and P2). Therefore, we need to determine whether or not the threshold voltage of the cell falls between V_a and V_c, requiring us to apply each of these two read reference voltages, with two consecutive read operations, to determine the MSB data.

As we discussed in Section 2.2, the threshold voltage distribution of each state can shift over time, causing some cells to move into neighboring voltage windows. To mitigate the number of errors that occur when a large number of cells experience threshold voltage shifts, modern NAND flash chips support the *read-retry mechanism* [8], which *adapts* the read reference voltages to correspond to the distribution shifts [3,8]. The read-retry operation allows the flash controller to increase or decrease the read reference voltages by multiples of a minimal voltage step (V_{step}), iterating over several potential values for each read reference voltage to *empirically* find the voltage value that yields the lowest raw bit error rate. By retrying to read the same flash page with different read reference voltages after a read failure, the flash controller increases the chances of reading the data correctly. For example, in Figure 2, the original read reference voltage V_a, between the ER and P1 states, misreads many of the flash cells (e.g., many cells actually belonging to the ER state would be incorrectly identified as belonging to the P1 state), but the flash controller can move the read reference voltage to V'_a, using the read-retry mechanism, thereby minimizing the number of errors.

2.5. Erase and Program Operations

In NAND flash, data can be programmed into only an erased flash cell. Due to circuit-level limitations, a flash block must be erased in its *entirety*. The erase operation resets the threshold voltage state of all cells in the flash block to the ER state.

When data is programmed, charge is transferred into the floating gate of a flash cell by repeatedly pulsing the programming voltage, in a procedure known as *incremental-step-pulse programming* (ISPP) [20]. In order to reduce the impact of interference caused by the programming process on the neighboring cells (called *program interference* [4]), *two-step programming* is employed for MLC NAND flash: the LSB is first programmed into a cell, and then the MSB is programmed only after partial data is programmed into neighboring cells [34]. In the first step, a flash cell is *partially programmed* based on its LSB value, either staying in the ER state if the LSB value is 1, or moving to a temporary state (TP) if the LSB value is 0. The TP state has a mean voltage that falls between states P1 and P2. In the second step, the LSB data is first read back into an internal buffer to determine the cell's current threshold voltage state, and then further programming pulses are applied based on the MSB data to increase the cell's threshold voltage to fall within the voltage window of its final state. In between the two steps, threshold voltage shifts can occur on the partially-programmed cell, as several other read and program operations to cells in neighboring pages may take place, causing interference. When this shifted data is read during the second programming step to retrieve the LSB value, the LSB value read may be incorrect, leading to a *program error* when the final cell threshold voltage is programmed [4].

Program errors predominantly shift data that should be in the ER state (11) into the P3 state (10), or data that should be in the P1 state (01) into the P2 state (00). This occurs because the programming that takes place in the second step can only *increase* (and not reduce) the threshold voltage of the cell from its partially-programmed voltage (and thus cannot move a cell that should be in the P3 state into the ER state, or one that should be in the P2 state into the P1 state).

3. Overview

In this paper, our overall goal is to build an *accurate*, and *easy-to-compute* model of the threshold voltage distribution of modern MLC NAND flash memory. This model must be practical to implement, as we intend to use it *online* to design flash controllers that can adapt to the changing NAND flash memory behavior. Our model can (1) *statically* determine the threshold voltage distribution at a given level of wear-out (i.e., a given P/E cycle count), and (2) *dynamically* predict how this threshold voltage distribution shifts over time as a result of the P/E cycling effect. In this section, we first discuss the steps required to construct our model (Section 3.1), and then we describe how we collect the experimental characterization data from real MLC NAND flash chips to drive our model construction (Section 3.2).

3.1. Constructing the Model

To build our threshold voltage distribution model, we first characterize the threshold voltage distribution under different P/E cycles using real 1X-nm MLC NAND flash chips. Second, we construct a static threshold voltage distribution model that can fit the characterized distribution under any given P/E cycle count. Third, we construct a dynamic P/E cycling model that predicts how each parameter of the static distribution model changes after some number of further P/E cycles. Finally, we demonstrate several example use cases in the flash controller that utilize the complete model to enhance the performance and reliability of the NAND flash memory device.

In order for the model to be useful to help flash controller algorithms, it should have certain properties. First, the model needs to be *accurate* for all threshold voltages and at all P/E cycles. This is because inaccurate information can lead a flash controller to make suboptimal decisions, hurting lifetime improvements. Second, the model needs to be *easy to compute*, because the flash controller has only limited computational resources. While we base our model off of a distribution that is easy to compute, we can further reduce online computation with the dynamic component of our model, by performing only a few online static characterizations of the threshold voltage distribution, and then using the simpler dynamic model to predict shifts in these initial characterizations at very low cost over P/E cycles.

3.2. Characterization Methodology

We collect experimental characterization data on the threshold voltage distribution using an FPGA-based NAND flash testing platform [1] with state-of-the-art 1X-nm MLC NAND flash chips. We use the read-retry technique [3, 8] (described in Section 2.4) to sweep *all possible* read reference voltages and determine the threshold voltage value for each cell. We program and erase these blocks to 11 different wear levels, up to 20K P/E cycles, using known pseudo-random data. The manufacturer-specified P/E cycle endurance for the tested flash chips is 3000 P/E cycles. All tests are performed at room temperature with a 5-second dwell time.[1]

Figure 3 shows the threshold voltage distribution for each of the cell states. The read-retry capability on the MLC NAND flash memory chip allows us to fine-tune each read reference voltage (V_a, V_b, and V_c) to one of 101 different steps (a total of 303 read reference voltage steps, labeled as V_1 to V_{303} from left to right). Note that V_1 does not extend all the way to the lowest possible threshold voltage for the ER state, and V_{303} does not extend to the highest possible threshold voltage for the P3 state. In this paper, we normalize the threshold voltage values such that the distance between most of the adjacent read reference voltage steps is one, as the exact values are proprietary information. The distances between steps V_{101} and V_{102} and between steps V_{202} and V_{203} are much larger than the typical distance between voltage steps,[2] as shown in Figure 3. As a result, the voltage step V_{303} has a normalized voltage value that is greater than 303. Overall, the threshold voltage range is divided by these read reference voltages into 304 bins, labeled as bin_0 to bin_{303}. Each flash cell can be classified into one of these bins based on the threshold voltage value read from the cell. If the read reference voltage is higher than the threshold voltage of the cell, the value read out from the flash device is 1, otherwise the value read out is 0. For a cell whose threshold voltage falls between two neighboring read reference voltages (V_k and V_{k+1}), the cell is placed into bin_k, as illustrated in Figure 3.

[1] *Dwell time* is the time duration between an erase operation and the next program operation to the same flash cell.

[2] Some flash vendors choose to provide fewer read reference voltages near the peak of the distribution of each state. This is because flash cells near the have threshold voltages far away from the default read reference voltage, and hence are less likely to have errors.

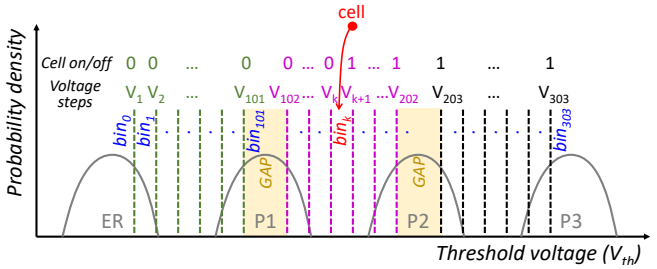


Fig. 3: Methodology for finding the threshold voltage of an MLC NAND flash memory cell.

After classifying every cell using the methodology described above, we count the number of flash cells with state $X \in \{\text{ER}, \text{P1}, \text{P2}, \text{P3}\}$ in bin k as $H_k(X)$. Equation 1 shows how we then normalize the bin counts as the probability density of each bin, $P_k(X)$. Note that in our characterization, we assign each flash cell to the threshold voltage distribution of the *correct state* that it was originally programmed to, as we know the data value that we programmed. The characterized bin density can be viewed as a discretized version of the measured distribution, which our model is constructed to fit.

$$P_k(X) = \frac{H_k(X)}{\Sigma_{i=0}^{303} H_i(X)} \quad (1)$$

4. Static Distribution Model

We construct a *static* threshold voltage distribution model that can fit the characterized threshold voltage distribution well under any P/E cycle count, based on data collected using the methodology described in Section 3.2. Recall that this model needs to be (1) accurate for all threshold voltages and at any given P/E cycle, and (2) easy to evaluate within the flash controller. While a more complex model can satisfy the accuracy requirements, it can be difficult to compute the model on the fly given the limited computational resources in a flash controller. In this section, we first describe two state-of-the-art models, each of which meets only one of our two requirements. The first previously-proposed model [8], based on a *Gaussian distribution*, is simple and easy to compute, but is not accurate enough for raw bit error rate estimation (Section 4.1). The second previously-proposed model [35], based on a *normal-Laplace distribution*, is accurate, but requires significant computational resources, taking 10.7x the computation time of the Gaussian-based model (Section 4.2). We propose a new model, based on our modified version of the *Student's t-distribution* [38], which satisfies both of our requirements, maintaining the accuracy of the normal-Laplace-based model while requiring 4.41x less computation time (Section 4.3). Finally, we validate and compare the three models (Section 4.4).

4.1. Gaussian-based Model

The Gaussian-based model assumes that the threshold voltage distribution of each state follows a Gaussian (i.e., normal) distribution [8]. Equation 2 shows how the Gaussian-based

model estimates the probability density for state X (i.e., ER, P1, P2, and P3) in each bin k, denoted as $G_k(X)$:

$$G_k(X) = GCDF(V_k, \mu_X, \sigma_X) - GCDF(V_{k-1}, \mu_X, \sigma_X) \quad (2)$$

The density $G_k(X)$ is calculated as the difference between the Gaussian cumulative distribution function ($GCDF$) of the bin's two boundaries, V_k and V_{k-1}. The Gaussian-based model has two variables for each state: μ_X is the mean of the distribution, and σ_X is the standard deviation of the distribution. In total, the Gaussian threshold voltage distribution model has eight parameters.

The intuition behind using a Gaussian distribution is twofold. First, the threshold voltage distribution is a result of physical noise and manufacturing process variation, which naturally follow a Gaussian distribution. During a program operation, the flash controller uses ISPP (see Section 2.5), iteratively increasing the threshold voltage until the desired threshold voltage level is achieved. Each programming step increases the threshold voltage of a cell by a small random amount. As programming subjects the cell to random physical noise, the threshold voltage distribution of each state naturally approximates a Gaussian distribution [8].

Second, the Gaussian-based model can be computed quickly, and is easily implementable in the flash controller hardware if we use a *z-table*, a lookup table that stores the *precomputed* cumulative distribution function of the standard Gaussian distribution. Equation 3 shows how the z-table simplifies the computation of $GCDF$. First, we calculate the z-scores $Z = \frac{V-\mu}{\sigma}$ for V_k and V_{k-1}. Then, we calculate $\Phi(Z)$, the precomputed cumulative distribution function of Z, by looking up the z-score in the z-table. The two z-scores (one each for V_k and V_{k-1}) are then combined to get $G_k(X)$, using Equation 2.

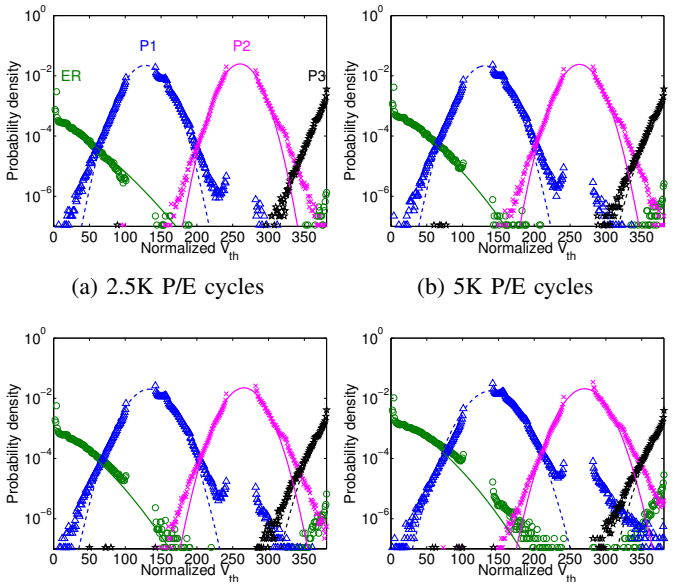
$$GCDF(V, \mu, \sigma) = \Phi(Z) = z\text{-}table(Z) \quad (3)$$

The goal of static modeling is to fit the estimated distribution $G_k(X)$ to the measured distribution $P_k(X)$. We use Kullback-Leibler divergence [19] to estimate the accuracy of the model (i.e., the error between the estimated and measured distributions). The Kullback-Leibler divergence between the measured and the estimated probability density for each bin (P_k and G_k, respectively) can be mathematically defined as:

$$D_{K-L} = \sum_{k=1}^{N_{bin}} P_k \log(\frac{P_k}{G_k}) \quad (4)$$

We use the Nelder-Mead simplex method [30] to minimize the error, in order to learn the model under different P/E cycles. We use a reasonable initial guess of the parameters for the Nelder-Mead simplex method, allowing us to quickly approach the best fit.

Figure 4 shows the distribution measured by our experimental characterization using markers, and shows how the Gaussian-based model (the curves depicted with solid or dashed lines) fits to this data at different P/E cycle counts. The x-axis is the normalized threshold voltage, and the y-axis is the probability density function at each normalized threshold voltage in log scale. In this figure, from left to right, we show the threshold voltage distribution of the ER state, the P1 state, the P2 state, and the P3 state. We show the modeled distributions of the ER and P2 states using solid lines, and the modeled distributions of the P1 and P3 states using dashed lines.



(a) 2.5K P/E cycles (b) 5K P/E cycles

(c) 10K P/E cycles (d) 20K P/E cycles

Fig. 4: Gaussian-based model (solid/dashed lines) vs. data measured from real NAND flash chips (markers) under different P/E cycle counts.

We observe that the Gaussian-based model has two limitations, which are demonstrated in Figure 4. First, the threshold voltage distribution of each state as measured from real flash chips has a *fatter tail* than that of a Gaussian distribution, and the left and right tails of each state have different sizes. We observe the fat tail by comparing the measured distribution of each state to the modeled distribution when the probability density is low (i.e., less than 10^{-4}), and find that the measured distribution has a much greater density than the modeled distribution at the tail. This is because the Gaussian distribution has only two parameters for each state, which capture only the center (μ) and the width (σ) of the distribution. We observe the asymmetric tail by comparing the densities of the left and right tails of the P2 state distribution. Unfortunately, the Gaussian distribution has no way to fine tune the ratio between the left and right tails, or the ratio between the tails and the body of the distribution.

Second, the measured distribution demonstrates large second peaks in the distributions of the ER and P1 states, which are not captured by the Gaussian-based model. These second peaks are evidence of a significant number of program errors (see Section 2.5). Figure 4 shows that the ER state distribution (the leftmost distribution) has a *second* peak that shows up under the P3 state distribution, and that the P1 state distribution (the second distribution from the left) has a second peak under the P2 state distribution. These second peaks occur as a result of the two-step programming mechanism used in MLC NAND flash memory. As we discuss in Section 2.5, *program errors* can be introduced for the ER and P1 states as a result of

intermediate operations that take place while a cell is partially programmed, which causes the LSB to be misread.

As we observe in Figure 4, both types of inaccuracies occur *throughout all P/E cycle counts* (from 2.5K to 20K), and are not, as prior work had shown [35], exclusive to high wear-out scenarios (e.g., when the P/E cycle count is higher than the vendor-specified lifetime). The magnitudes of the fatter tails and program error peaks increase as wear-out (i.e., P/E cycle count) increases. As we can see, even though the Gaussian-based model captures the general trend for the threshold voltage distribution and is easy to compute, it is limited in its accuracy, especially at higher P/E cycles.

Section 4.4 quantifies the modeling error and computational requirements of the Gaussian-based model.

4.2. Normal-Laplace-based Model

To overcome the limitations of the Gaussian-based model, prior work [35] proposes to modify the model to increase its accuracy. This modified threshold voltage distribution model *assumes* that the distribution of each state follows a normal-Laplace distribution, and accounts for the peaks that result from misprogramming some cells that should be in the ER and P1 states into the P3 and P2 states, respectively [35].

The normal-Laplace distribution combines the normal (Gaussian) distribution with the Laplace distribution, which adds an exponential component to both tails of the distribution. As we observe in Figure 4, this is similar to the measured behavior of the threshold voltage distribution. Note that the figure is in log scale, and as a result, the exponential component at the tails of the model appears as a straight line in the figure. By combining the two probability distributions, we can maintain the Gaussian distribution at the center of the distribution, and also model the fat tail more accurately.

However, computing the normal-Laplace distribution becomes much more complex than the Gaussian distribution, as the normal-Laplace distribution is *not* a simple superposition of the Gaussian and Laplace distributions. Equation 5 shows how we compute the cumulative distribution function for the normal-Laplace distribution [37]:

$$NCDF(V,\mu,\sigma,\alpha,\beta,\lambda) = \Phi(Z) - \phi(Z)\frac{\beta R(\alpha\sigma - Z) - \alpha R(\beta\sigma + Z)}{\alpha + \beta} \quad (5)$$

This distribution adds two new parameters, α and β, which can be adjusted to model the right and left tail sizes, respectively. In Equation 5, $Z = \frac{V-\mu}{\sigma}$ is the z-score; Φ and ϕ are the cumulative distribution function and probability density function of the standard Gaussian distribution, respectively; and $R(x) = \frac{1-\Phi(x)}{\phi(x)}$ is Mills' ratio for the Gaussian distribution. $\Phi(Z)$ can be obtained by looking up the z-table, as was done for Equation 3. $\phi(Z)$ can be approximated as $\phi(Z) = \frac{\Phi(Z+\delta)-\Phi(Z-\delta)}{2\delta}$.

The normal-Laplace-based model adds two further parameters, λ_{ER} and λ_{P1}, to model the probability of program errors occurring for cells programmed to the ER and P1 states, respectively. This model assumes that the threshold voltage distribution of the cells with program errors has the same shape (i.e., the same parameters) as the distribution of the state the cells were incorrectly programmed into (e.g., the cells that should be in the ER state but were programmed into the P3 state will have a distribution with the same shape as the correct cells in the P3 state). This is because once the cells are incorrectly programmed to another state, they are treated as if they belong to that other state, and thus it is natural for them to follow the same distribution as the correct cells in that state. Equation 6 shows how the normal-Laplace model estimates the probability density for state X being misprogrammed to state Y in bin k, which is denoted as $N_k(X)$:

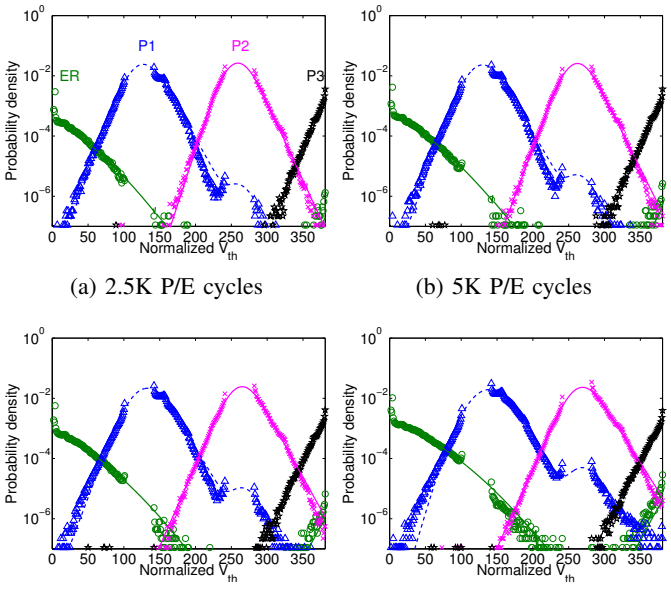
$$\begin{aligned} N_k(X) = {} & (1-\lambda_X)NCDF(V_k,\mu_X,\sigma_X,\alpha_X,\beta_X) \\ & + \lambda_X NCDF(V_k,\mu_Y,\sigma_Y,\alpha_Y,\beta_Y) \\ & - (1-\lambda_X)NCDF(V_{k-1},\mu_X,\sigma_X,\alpha_X,\beta_X) \\ & - \lambda_X NCDF(V_{k-1},\mu_Y,\sigma_Y,\alpha_Y,\beta_Y) \end{aligned} \quad (6)$$

This density is calculated as the difference of the $NCDF$ at the bin's two boundaries, V_k and V_{k-1}. The normal-Laplace-based model allows each state to have at most five parameters (20 parameters over all four states). μ and σ are the mean and standard deviation, respectively; α and β are the tail sizes; and λ_X is the probability that a cell that should actually be in state X is incorrectly programmed.

Following prior work [35], we eliminate four unnecessary parameters of the model, which include λ_{P2}, λ_{P3}, β_{ER}, and α_{P3}. λ_{P2} and λ_{P3} are estimated as zero, as program errors for cells that should be in the P2 or P3 states seldom occur. We also assume that the left and right tails are the same size for the ER and P3 states (i.e., $\beta_{ER} = \alpha_{ER}$ and $\alpha_{P3} = \beta_{P3}$), because the read-retry mechanism prevents us from measuring the left tail of the ER state and the right tail of the P3 state. As we did in Section 4.1, and following prior work [35], we use Kullback-Leibler divergence error [19] as the objective function, and we use the Nelder-Mead simplex method [30] with a reasonable initial guess to learn the best parameters under different P/E cycles.

Figure 5 shows the modeled distribution of each state as curves with solid or dashed lines, and shows the distribution measured from real chips using markers. As we can see, *the normal-Laplace-based model fits the measured distribution much better than the Gaussian-based model*. The modeled tails for the ER, P2, and P3 states follow the measured distribution very closely, thanks to the tail size parameters. Also, the distributions of the ER and P1 states take the program error rate into account, and allow the model to correctly include two peaks for the distributions of the ER and P1 states.

Unfortunately, although the normal-Laplace model is based on the Gaussian model, the computational requirements of the model are much more complex. This is not only because the model adds three more parameters for each state, but also because we now cannot eliminate μ and σ using the z-score. Thus, directly computing the model requires many more floating point operations than the Gaussian model (as we demonstrate in Section 4.4). As such, even though the normal-Laplace model fits the measured threshold voltage distribution accurately, it is less practical to implement.



(a) 2.5K P/E cycles (b) 5K P/E cycles

(c) 10K P/E cycles (d) 20K P/E cycles

Fig. 5: Normal-Laplace-based model (solid/dashed lines) vs. data measured from real NAND flash chips (markers) under different P/E cycle counts.

Section 4.4 quantifies the modeling error and computational requirements of the normal-Laplace-based model.

4.3. Student's t-based Model

Recall that we need a threshold voltage model that is both accurate and easy to compute. As we can see, the Gaussian-based model is simple and fast, but does not meet the accuracy requirement. In contrast, the normal-Laplace-based model fixes the accuracy problem, but uses significantly more complex calculations. We thus aim to develop a model that meets both of our requirements at the same time.

We propose to modify the Student's t-distribution [38] so that it can be used to model the threshold voltage distribution. The Student's t-distribution is a well-known distribution used in statistics that describes samples drawn from a normally-distributed population. The Student's t-distribution is typically used to estimate the true mean of a large, normally-distributed population whose standard deviation is unknown, using only a small sample from the population. Compared to the standard normal distribution, the Student's t-distribution uses an extra parameter, ν, to represent the *degrees of freedom* (i.e., the ratio of the sample size relative to the population size). As ν increases (i.e., the sample size becomes larger), the Student's t-distribution moves closer to a standard normal distribution. However, instead of using the distribution for its original purpose, we use this distribution for a completely different role. We find that ν can be used to tune the size of the distribution tail. When $\nu \to +\infty$, the Student's t-distribution becomes a standard Gaussian distribution, which has a smaller tail. When $\nu \to 0$, the distribution instead has a fatter tail. We generalize the standard Student's t-distribution using the z-score $Z = \frac{V-\mu}{\sigma}$, such that the center and the width of the distribution can be scaled (as was done for the Gaussian distribution). We also allow the left and right tails of the distribution to have different values of ν, which we denote as β and α for the left and right tails, respectively. Thus, our modified Student's t-distribution can fit our measured threshold voltage distribution better than the original Student's t-distribution.

We use *precomputation* to simplify the calculation of the cumulative distribution function for our modified Student's t-distribution ($TCDF$). Similar to the precomputed z-tables available for the Gaussian-based model, we look up values in the precomputed *t-tables* commonly available for the Student's t-distribution to determine the $TCDF$ values. Each t-table contains $TCDF$ values over a range of Z values *for a single* ν.[3] Equation 7 shows how we calculate $TCDF$ using the precomputed t-table:

$$TCDF(V,\mu,\sigma,\alpha,\beta) = \begin{cases} t\text{-}table_{\beta}(Z) & V \le \mu \\ t\text{-}table_{\alpha}(Z) & V > \mu \end{cases} \quad (7)$$

We first compare V with μ to observe whether V is on the left side or the right side of the distribution. Then, depending on the result of the comparison, we use the corresponding tail parameter α or β as ν to select the correct t-table. Finally, we compute the z-score Z to look up $TCDF$ in the selected t-table.

Equation 8 shows how our Student's t-based model estimates the density for cells that should be in state X but are incorrectly programmed to state Y in bin k, which is denoted as $T_k(X)$:

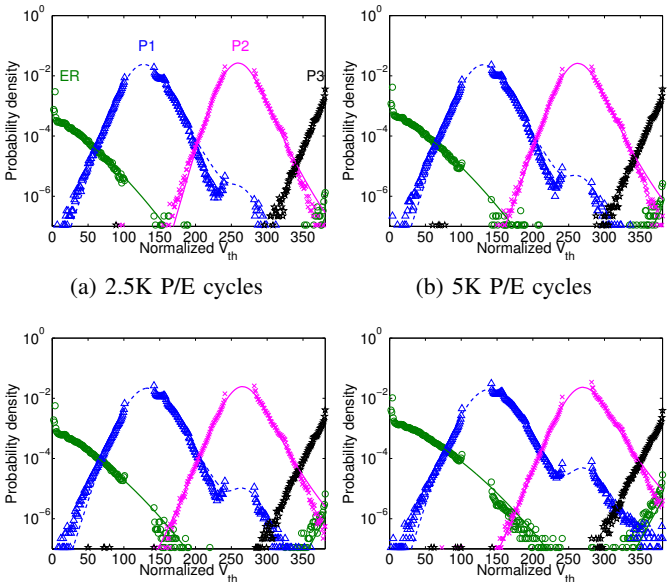
$$\begin{aligned} T_k(X) = {} & (1-\lambda_X)TCDF(V_k,\mu_X,\sigma_X,\alpha_X,\beta_X) \\ & + \lambda_X TCDF(V_k,\mu_Y,\sigma_Y,\alpha_Y,\beta_Y) \\ & - (1-\lambda_X)TCDF(V_{k-1},\mu_X,\sigma_X,\alpha_X,\beta_X) \\ & - \lambda_X TCDF(V_{k-1},\mu_Y,\sigma_Y,\alpha_Y,\beta_Y) \end{aligned} \quad (8)$$

Similar to the normal-Laplace-based model (Section 4.2), our Student's t-based model uses λ to estimate such program errors caused by the two-step programming mechanism (see Section 2.5). Again, like the normal-Laplace-based model, our Student's t-based model assumes that the distribution of these cells has the same parameters as the cells correctly programmed into state Y.

We set λ_{P2} and λ_{P3} to zero, $\beta_{ER} = \alpha_{ER}$, and $\alpha_{P3} = \beta_{P3}$ for the same reasons as the normal-Laplace-based model (see Section 4.2). Putting everything together, we use Kullback-Leibler divergence error [19] as our objective function, and use the Nelder-Mead simplex method [30] with a reasonable initial guess to learn the best parameters for the model under different P/E cycles, as described in Section 4.1.

Figure 6 shows our modeled Student's t-based distribution as curves with solid or dashed lines, once again showing the distribution measured from real chips with markers. The figure shows that our Student's t-based model fits perfectly when the probability density is greater than 10^{-4}. The differences between our Student's t-based model and the measured

[3] The t-table can also be thought of as a two-dimensional array, where each entry corresponds to a unique pair of (Z, ν) values.



(a) 2.5K P/E cycles (b) 5K P/E cycles

(c) 10K P/E cycles (d) 20K P/E cycles

Fig. 6: Our new Student's t-based model (solid/dashed lines) vs. data measured from real NAND flash chips (markers) under different P/E cycle counts.

distribution are within 10^{-6}. The difference becomes non-trivial only for the left tail of the P1 state and the right tail of P2 state. The accuracy improvements over the Gaussian model are similar to those of the normal-Laplace-based model. This shows that our Student's t-based model, like the normal-Laplace-based model, makes good use of its extra parameters (both models have 16 parameters) to cater to the program errors and fat tails that the measured distribution has.

4.4. Model Validation and Comparison

Accuracy. To quantitatively compare the accuracy of each model and validate them, we compute the Kullback-Leibler (K-L) divergence [19] between the modeled and the measured distributions, as K-L divergence measures the difference between two distributions (see Section 4.1). Table 1 and Figure 7 show the modeling error of the three models across a range of P/E cycle counts. We observe two types of behavior shared by all three models. First, as the P/E cycle count increases, the modeling error increases. Second, the increase in modeling error is more rapid at *smaller* P/E cycle counts, and slower at higher P/E cycle counts. As we see in Section 5, this is because the threshold voltage distribution is affected by the P/E cycling effect more significantly at smaller P/E cycle counts.

Comparing the three models in Figure 7, we make two observations. First, the average Kullback-Leibler divergence error for the Gaussian-based model is 2.64%, which is 4.32x and 3.88x greater than the error of the normal-Laplace-based and our Student's t-based models, respectively. We also see that this error can be as large as 8.7%, leading to high inaccuracy. This is mainly due to two limitations of the Gaussian-based model. As mentioned in Section 4.1, the Gaussian-based model (1) cannot adjust its tail size to fit with the fat and asymmetric

P/E Cycles	0	2.5K	5K	7.5K	10K	12K	14K	16K	18K	20K	AVG
Gaussian	.99%	1.8%	1.6%	1.8%	1.9%	2.4%	3.1%	8.7%	2.1%	2.3%	**2.6%**
Normal-Laplace	.34%	.46%	.55%	.61%	.63%	.67%	.68%	.70%	.67%	.67%	**.61%**
Student's t	.37%	.51%	.61%	.68%	.70%	.76%	.76%	.78%	.76%	.78%	**.68%**

TABLE 1: Modeling error of the evaluated threshold voltage distribution models, at various P/E cycle counts.

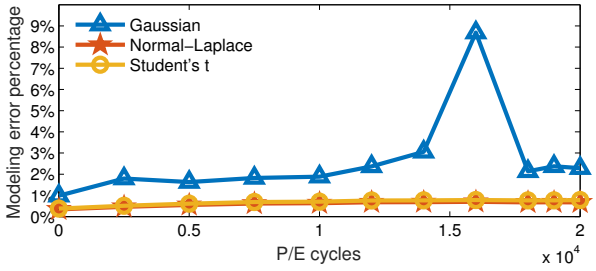


Fig. 7: Modeling error of the evaluated threshold voltage distribution models, at various P/E cycle counts.

tails of the observed distributions of the voltage states, and (2) does not account for misprogrammed cells that form a second peak in the distributions of the ER and P1 states.

Second, the modeling errors of the normal-Laplace-based and our Student's t-based models are very close (averaged across all tested P/E cycles, 0.61% for the normal-Laplace-based model, and 0.68% for our Student's t-based model). The maximum difference in error between these two models is 0.11%, at 20K P/E cycles (already well beyond the rated lifetime of the flash chip, which is 3K P/E cycles).

Complexity. All three of the models require online *iterative computation* whenever the flash controller needs to generate a characterization of the threshold voltage distribution at a new P/E cycle count. For each iterative computation, hundreds to thousands of iterations of the model computation algorithm must be executed before the model reaches high accuracy (i.e., the model converges, or in other words, reaches *convergence*). As we alluded to in Section 4.2, while the normal-Laplace-based model is accurate, it requires significant computation during each iteration, and cannot be precomputed and stored in a lookup table, making it less practical for use within a flash controller. To compare the complexity of the three models, we summarize their computation overhead in terms of the number of floating-point operations and table lookups performed for each iteration, as well as their storage overhead in terms of lookup table size. Table 2 compares the three models. As we can see, the normal-Laplace-based model requires 91,200 operations per iteration (which involves computing $N_k(X)$ for four states in each of the 304 threshold voltage bins). Assuming that each floating-point operation takes the same number of cycles, the normal-Laplace-based model is 10.71x slower than the Gaussian-based model. In contrast, our Student's t-based model takes 4.41x less computation time than the normal-Laplace-based model, with near-identical accuracy. Our Student's t-based model is only 2.43x slower than the Gaussian-based model, but has a 74% smaller modeling error.

Model	Gaussian	Normal-Laplace	Student's t
Operations	8512	91200	20672
Storage	640B	3.84KB	25.6KB

TABLE 2: Computation and storage complexity comparison for the three evaluated threshold distribution models.

The third row of the table shows the storage overhead for the z-table or t-tables used by each model (which are populated in the flash controller's DRAM when the flash device is powered up). The Gaussian-based model needs only 640B to store the useful range of the z-table. The normal-Laplace-based model requires a larger lookup range for the z-table, increasing the storage overhead to 3.84KB. Our Student's t-based model requires storing multiple t-tables (one table per value of ν), and uses 25.6KB of storage in total. We find that all three storage overhead values are negligible, as these tables are easily stored within the flash controller's DRAM, which is usually sized to be a fixed fraction of the flash storage capacity (e.g., 1GB memory for a 512GB drive).

Latency. The flash controller builds a threshold voltage distribution model in two steps: *characterization* and *model computation*. First, we identify the threshold voltages of each cell in a sampled flash wordline by performing 303 read operations, one read for each read reference voltage level (using the approach described in Section 3.2). This *characterization* takes 30.3 ms for the wordline, assuming a typical read latency of 100μs. Second, once characterization is complete, the controller *computes* the model, using a combination of the precomputed tables stored in DRAM and the characterized voltages. To calculate the overhead, we assume that each of the models takes 1000 iterations to converge, and that computation is performed on a 1GHz embedded processor that completes one instruction per cycle.

Figure 8 shows the overall latency for the three models we evaluate, broken down into characterization latency (which is the same for all three models) and model computation latency. The computation overhead of the normal-Laplace-based model dominates its overall latency, while the computation overhead of our Student's t-based model is much smaller than the characterization latency. As a result, our Student's t-based model has a 58.0% lower overall latency than the normal-Laplace-based model. Since the fixed characterization latency dominates overall latency in both our Student's t-based model and the Gaussian-based model, our model is only 31.3% slower in overall latency than the Gaussian-based model, while it reduces modeling error by 74%.

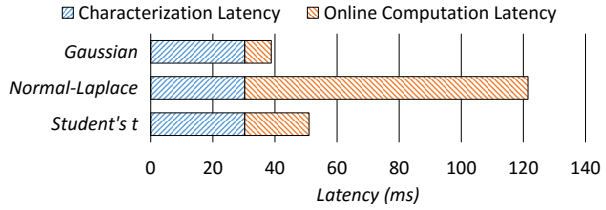


Fig. 8: Overall latency breakdown of the three evaluated threshold voltage distribution models for static modeling.

The frequency with which the characterization and modeling procedure is triggered depends purely on the application making use of the threshold voltage distribution model. Note that the choice of model should not change the frequency at which the procedure is executed (as each model provides an equivalent end result). As an example, we can determine the amortized overhead per 4KB read/write operation for one application of our model, which predicts the optimal read reference voltage (see Section 6.2). The prediction mechanism requires us to repeat the characterization and modeling procedure only once every 1000 P/E cycles. For a flash device with 512 pages per block, if we conservatively assume a read-to-write ratio of 1:1, the average overhead amortized over each read/write operation is 49.8ns using our static Student's t-based model [25].

Summary. In summary, the majority of the accuracy improvement over the Gaussian-based model comes from (1) accounting for the program errors for the erased and P1 states, and (2) accounting for the fat tails of each state. Our Student's t-based model, as well as the previously-proposed normal-Laplace-based model, both contain these improvements, and hence achieve similar accuracy.

Our Student's t-distribution based model has much lower complexity than the normal-Laplace-based model due to its ability to exploit precomputation. We show in Section 4.3 that the CDF of the Student's t-distribution can be simplified into a simple table lookup using the z-score, Z, and the degrees of freedom, ν. We are unaware of a similar precomputation-based approach that can be applied to the normal-Laplace model.

We conclude that our new Student's t-based model achieves the high accuracy of the normal-Laplace-based model while requiring significantly less complexity and latency to compute. As such, we believe that our Student's t-based model meets the requirements of accuracy and simplicity, and is a practical model for implementation within the flash controller.

5. Dynamic Modeling

We now construct a *dynamic* threshold voltage distribution model, building off of our Student's t-based static model in Section 4.3, to capture how the threshold voltage distribution *changes* as the program/erase (P/E) cycle count increases. Again, we must ensure that this dynamic model is accurate, and that it is easy to compute, as we aim to implement the model within the flash controller. To construct the model, we first analyze how each of the individual parameters making up our Student's t-based model change over the P/E cycle count (Section 5.1). By analyzing the meaning of each parameter and observing how it changes, we gain new insights on how the threshold voltage shifts with increasing P/E cycle count. We then use these new insights to construct a model using the power law, which can successfully predict the *future* changes to each of these parameters based on the *current* threshold voltage distribution (Section 5.2). Finally, we validate this model (Section 5.3).

5.1. Static Model Trends Over P/E Cycles

In order to analyze and observe how the parameters for our Student's t-based model change as P/E cycle count increases,

we first need to understand what each parameter means. As we discuss in Section 4.3, our Student's t-based model has 16 parameters. Four of them are the mean values for each state X's threshold voltage distribution (μ_X). Another four parameters are the standard deviation values of the threshold voltage distribution of each state X (σ_X). Three of them are the left tail sizes of the P1, P2, and P3 state distributions (β_X), and another three are the right tail sizes of the ER, P1, and P2 state distributions (α_X). (Recall from Section 4.2 that the left tail of the ER state and the right tail of the P3 state cannot be observed experimentally, so we assume that they equal the right tail of the ER state and the left tail of the P3 state, respectively.) The remaining two parameters are the probability of program errors, occurring for cells programmed into the ER and P1 states (λ_X).

Mean. The mean value of each state represents the center of the distribution. In our Student's t-based model, the majority of the mass of the threshold voltage distribution for each state is near the center. Thus, a change in the mean reflects how the P/E cycle count generally affects the threshold voltages of *all* cells in each state.

Figure 9 plots the mean values obtained from sample Student's t-based models constructed over a range of 20K P/E cycles, shown as circles. The x-axis shows the P/E cycle count, while the y-axis shows the normalized threshold voltage of the mean. Each graph plots the mean value for a different state, which is labeled at the top of the graph. We make three observations from this figure. First, the mean value of each state's distribution increases monotonically with P/E cycle count. Second, the mean value increases faster at lower P/E cycle counts, then slows down to a constant rate of increase after 5K P/E cycles. Third, the mean value shifts more quickly for lower threshold voltage states (ER, P1).

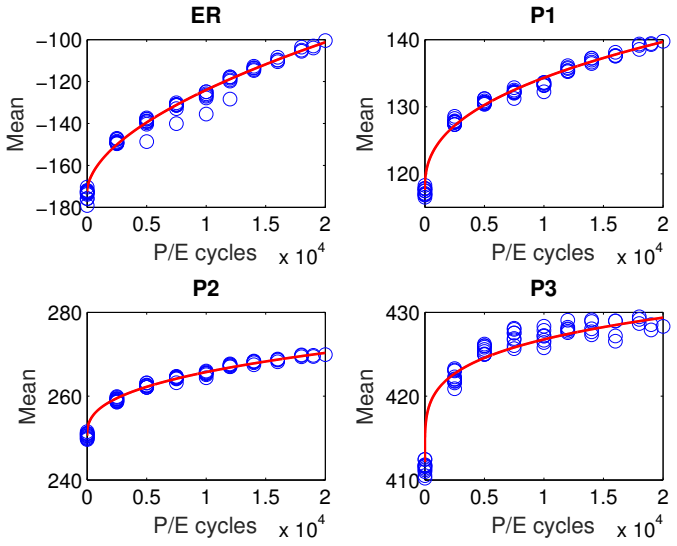


Fig. 9: Change in mean value of each state's threshold voltage distribution as P/E cycle count increases, for the static Student's t-based model (blue circles) and the dynamic model (red line).

Standard Deviation. The standard deviation of each state represents the width of the distribution. Similar to the Gaussian distribution, the Student's t-distribution contains the vast majority (~95%) of its mass within two standard deviations. Thus, the change in standard deviation reflects how P/E cycle count affects the threshold voltage variation among flash cells.

Figure 10 plots the standard deviation values obtained from our Student's t-based model as circles. For this figure, the y-axis shows the standard deviation in terms of normalized threshold voltage. We make three observations from this figure. First, the standard deviation of each state's distribution increases monotonically with P/E cycle count. Second, the standard deviations of the P1 and P2 states increase linearly with P/E cycle count. Third, like the mean, the standard deviation increases faster at lower P/E cycle counts, then slows down to a constant rate of increase after 5K P/E cycles.

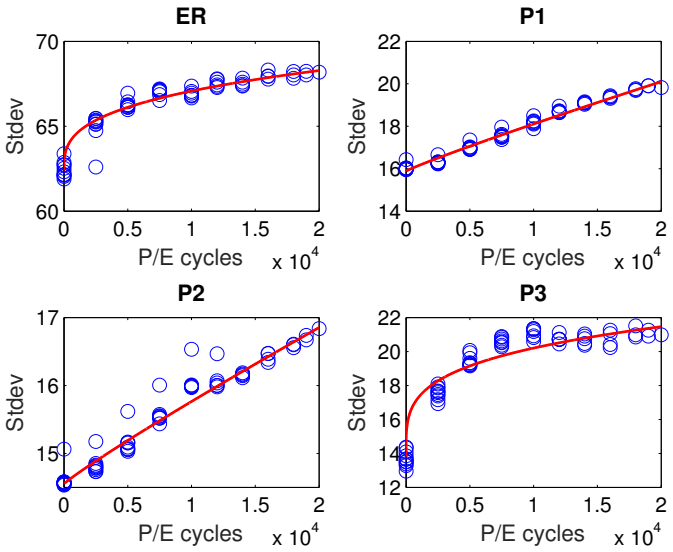


Fig. 10: Change in standard deviation of each state's threshold voltage distribution as P/E cycle count increases, for the static Student's t-based model (blue circles) and the dynamic model (red line).

Tail Values. The tail values of each state represent the size and shape of the distribution tail. Recall from Section 4.3 that we use ν, which actually represents the degrees of freedom, to control how fat the tail of the model is. Thus, the tail value reflects how the P/E cycle count affects the number of outlier cells (i.e., the number of cells that lie at the tail).

Figure 11 plots the tail values obtained from our Student's t-based model as circles. In this figure, the y-axis shows the value of ν, where a lower value of ν corresponds to a fatter tail. We make three observations. First, the range of values for the tail sizes of the ER and P3 states is much smaller in comparison to the tail sizes of the distributions of the other states. Second, the sizes of both tails for the P1 state increase with P/E cycle count. Third, the tail sizes of the P2 state decrease as P/E cycle count increases.

Probability of Program Errors. The program error probability λ_X represents the percentage of cells that should be programmed into state X, but are instead misprogrammed to

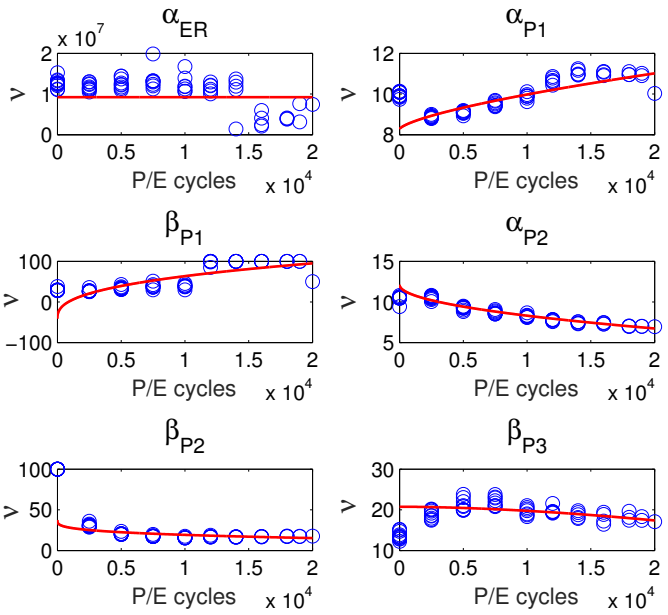


Fig. 11: Change in tail values (v) of each state's threshold voltage distribution as P/E cycle count increases, for the static Student's t-based model (blue circles) and the dynamic model (red line).

a different state, as a result of two-step programming (see Section 2.5). In our model, we assume that only certain types of program errors exist (ER→P3 and P1→P2), as program errors flip the value of only the LSB within a cell and can only *increase* the threshold voltage.

Figure 12 plots the program error probability obtained from our Student's t-based model as circles. For this graph, the y-axis shows the $\log_{10}$ value of the program error probability. We make two observations. First, the program error rate increases with P/E cycle count. Second, the number of program errors increases more rapidly at lower P/E cycle counts, and then slows down to a constant rate of increase at higher P/E cycle counts.

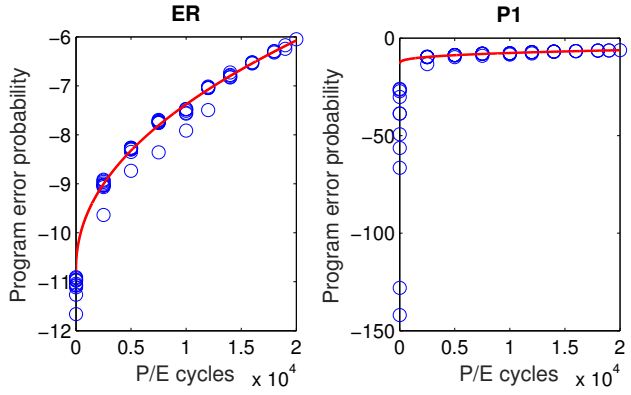


Fig. 12: Change in log value of the program error probability as P/E cycle count increases, for the static Student's t-based model (blue circles) and the dynamic model (red line).

5.2. Power Law-based Model

Now that we have characterized how each of the parameters for our Student's t-based model changes with respect to the P/E cycle count, we use this characterization to develop a *dynamic* model of the threshold voltage distribution. A dynamic model can reduce the total computation effort for the threshold voltage distribution significantly, by requiring as little as a *single* static model *characterization* for the entire lifetime of the flash device. The dynamic model takes the static characterization-based model(s) generated in the past, and simply adjusts the model parameters at higher P/E cycle counts based on its prediction of how each parameter would change with P/E cycle count (without requiring any further characterization). Without the dynamic model, a static model of the characterization must be generated *every time* a new threshold voltage distribution is requested by the controller (e.g., after a fixed number of P/E cycles have occurred), with each characterization requiring a large number of read-retry operations (see Section 4.4). These read-retry operations increase the accuracy of the model, but interfere with and slow down host commands. Our goal is to build a dynamic model that is accurate and easy to compute (such that it requires only a small number of characterizations), so that it can be used within the flash controller.

In Section 5.1, we observe that all of the parameters can increase, decrease, or remain relatively constant. We also observe that the rate at which increases and decreases occur differs between lower P/E cycle counts and higher P/E cycle counts. Our dynamic model must be able to represent all of these behaviors. We find that the *power law* satisfies all of these characteristics. Equation 9 shows the power law function, which models each parameter from our Student's t-based model, Y, as a function of the P/E cycle count (x):

$$Y = a \times x^b + c \tag{9}$$

The power, b, can be set to a positive value to represent an increasing trend, or can be set to a negative value to represent a decreasing trend. b can also control the difference in slope at different P/E cycle counts. For example, when $b < 1$, the modeled parameter Y changes faster at *lower* P/E cycle counts, and when $b > 1$, Y changes faster at *higher* P/E cycle counts.

To observe how well the power law models changes to the parameters of our Student's t-based model, we fit the power law to the values of each of the parameters as measured over several P/E cycle counts (see Section 5.1).[4] We use mean squared error (MSE) to estimate the error, where the divergence between the measured and estimated parameters (Y_i and $\hat{Y}_i$, respectively) can be mathematically defined as: $MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$. We use the Nelder-Mead simplex method [30], with a reasonable initial guess, to fit the trend.

Figures 9, 10, 11, and 12 show the power law-based models fit to the trends of each of our parameters as solid lines. We fit the power law to the static model parameter values generated over a range of 20K P/E cycles. We observe that the predictions from the power law fit very well with the actual

[4]We exclude 0 P/E cycle results when modeling, as they show a completely different behavior than results at any other P/E cycle count.

parameters measured from our Student's t-based model, which are shown as blue circles. We next quantify the accuracy of our power law-based dynamic model.

5.3. Model Validation

We validate our dynamic model by using it to predict the threshold voltage distribution at 20K P/E cycles. We perform threshold voltage distribution characterizations at 2.5K, 5K, 7.5K, and 10K P/E cycles, and use these parameters to predict the distribution at 20K P/E cycles. Figure 13 shows the comparison between the actual characterized distribution (markers) and the distribution predicted by our dynamic model (solid or dashed curves) at 20K P/E cycles. The modeling error for the dynamic model is only 2.72%, which is close to the modeling error of directly using a static Gaussian-based model at 20K P/E cycles. The dynamic model avoids the need to perform the extensive read-retry characterization that all static models, including the Gaussian-based model, would require.

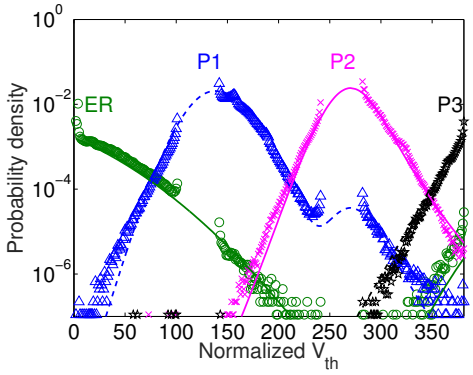


Fig. 13: Threshold voltage distribution as predicted by our dynamic model for 20K P/E cycles, using characterization data from 2.5K, 5K, 7.5K, and 10K P/E cycles, shown as solid/dashed lines. Markers represent data measured from real NAND flash chips at 20K P/E cycles.

Figure 14 shows how the modeling error of our dynamic model decreases for a prediction at 20K P/E cycles as the number of characterized data points increases. The number of characterized data points represents the N earliest static models out of a range that consists of static models for 2.5K, 5K, 7.5K, 10K, 12K, 14K, 16K, 18K, and 19K P/E cycles. Note that we start with three characterization data points, which allows the dynamic model to observe a trend in the change of each parameter. This figure shows that the error rate decreases rapidly as we increase the number of data points used to train the dynamic model.

We conclude that our dynamic model successfully predicts an accurate threshold voltage distribution for a P/E cycle count it has not observed, based on only prior characterization data, and is thus practical for use in the flash controller.

6. Example Applications

Now that we have developed our threshold voltage distribution model, we demonstrate three example applications

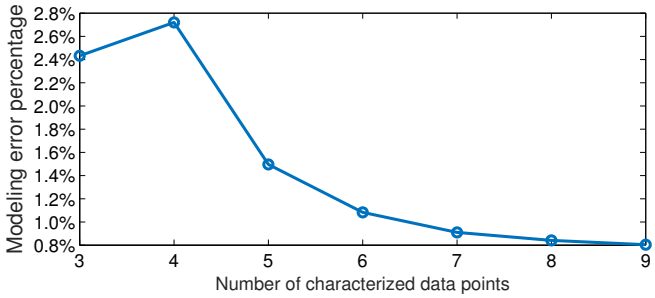


Fig. 14: Modeling error of predicted threshold voltage distribution for our dynamic model at 20K P/E cycles, using characterization data from N different P/E cycles.

within the flash controller that take advantage of the model to enhance the reliability of the flash device. The first application, described in Section 6.1, uses our model to accurately estimate the raw bit error rate. The second application, described in Section 6.2, uses our model to accurately predict the optimal read reference voltage. The third application, described in Section 6.3, uses our model to estimate the expected lifetime of the flash memory device, to safely achieve higher P/E cycle endurance than manufacturer specification.

6.1. Raw Bit Error Rate Estimation

The raw bit error rate (i.e., the probability of reading an incorrect state for a flash cell), or RBER, is important not only because it is a measure of the reliability of a flash device, but because it also can be used to determine the lifetime and performance of the flash drive [3]. The raw bit error rate can be used to enable several optimizations in the flash controller. For example, accurately estimating the current raw bit error rate allows us to safely utilize the *currently unused* ECC correction capability to accelerate program operations [18], relax the retention time [3,23], and reduce the effects of read disturbance [9]. Accurate estimation of the raw bit error rate enables other optimizations, such as predicting the optimal read reference voltage [4] or performing error rate based wear-leveling.

To estimate the raw bit error rate based on the static threshold voltage distribution model, we use the static model to calculate the cumulative distribution function (CDF) for each state at each of our read reference voltages (V_a, V_b, and V_c), and use this data to determine how many cells are misread. For example, if there are cells in the distribution of the ER state whose threshold voltages are greater than V_a, they will be misread (see Figure 2). By calculating the ER state CDF up to V_a, we know what percentage of cells will be *correctly* read. We subtract this value from 1 to obtain the percentage of cells that will be *misread* (and will thus contribute to the raw bit error rate).

Figure 15 shows the actual measured raw bit error rate and the modeled raw bit error rates using the three static models from Section 4, for different P/E cycle counts. The x-axis shows P/E cycle count, and the y-axis shows the measured or model-predicted raw bit error rate. The three graphs show the average error rate for only the LSB pages, only the MSB

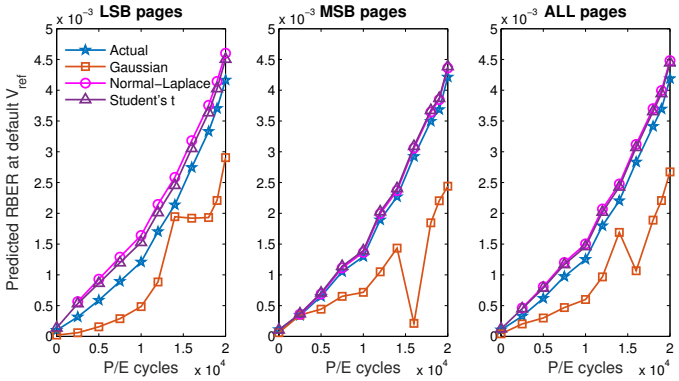


Fig. 15: Actual and modeled raw bit error rate using the three evaluated threshold voltage distribution models when reading with fixed *default* read reference voltages (V_{ref}), across different P/E cycle counts.

pages, and for all of the pages. We make two observations from this data. First, the normal-Laplace-based and our Student's t-based models give a much better estimate of the raw bit error rate than the Gaussian-based model. Averaged across all P/E cycle counts, our Student's t-based model estimates the RBER for all pages within 13.0% of the actual measured RBER, while the normal-Laplace-based model is within 14.9% and the Gaussian-based model is only within 44.7%. This is due to the limitations of the Gaussian-based model, as it cannot adjust the tail size or take program errors into account. Second, the normal-Laplace-based and our Student's t-based models tend to overestimate the error rate, which is usually safe for the purposes of many optimizations, because overestimation results in more than adequate ECC correction capability to remain available for these errors. In contrast, the Gaussian-based model always *underestimates* the raw bit error rate, which, if used for an optimization that relies on an RBER estimation, can cause the number of errors to exceed the correction capability of ECC, resulting in uncorrectable errors during reads. We conclude that our Student's t-based model is effective at providing an accurate estimate of the raw bit error rate for use by the flash controller.

6.2. Optimal Read Reference Voltage Prediction

As we discussed in Section 2.4, when the threshold voltage distribution shifts, it is important to move the read reference voltage to the point where the number of read errors is minimized. After the shift occurs, the threshold voltage distributions of each state may overlap with each other, causing many of the cells within the overlapping regions to be misread. The number of errors due to misread cells can be minimized by setting the read reference voltage to be at the point where the distributions of two neighboring states *intersect*, which we call the *optimal read reference voltage* (V_{opt}) [4]. Once the optimal read reference voltage is applied, the raw bit error rate is minimized, improving the reliability of the device. Furthermore, since fewer errors are corrected, and fewer read-retries are needed, read latency is also significantly reduced [3].

Prior work proposes to learn and record the optimal read reference voltage periodically [3, 32, 39] by sampling the threshold voltages of *some* of the cells in each flash block, but this sampling requires time and storage overhead. With our new distribution model, we can determine the optimal read reference voltage from the model and predict how it changes, without having to exhaustively learn it for each block. From our threshold voltage distribution model, we can predict the optimal read reference voltage by finding the point at which the probability density functions of the distributions of two neighboring states are the same (i.e., the intersection of the two distributions).

Figure 16 plots the actual measured and modeled optimal read reference voltage using the three static models from Section 4, at different P/E cycle counts.[5] Each graph shows the voltage chosen for one of the three read reference voltages (V_a, V_b, and V_c) used to distinguish between the distributions of two neighboring states (see Figure 1). The x-axis shows the P/E cycle count, while the y-axis shows the normalized optimal read reference voltage. We make three observations from this result. First, the normal-Laplace-based and our Student's t-based models slightly overestimate all three optimal read reference voltages. Second, the Gaussian-based model underestimates the optimal read reference voltages in most cases, and has glitches of underestimation as large as 17 voltage steps. We suspect that this is because the Gaussian-based model cannot capture the asymmetric tail sizes of the distribution. Third, at 0 P/E cycles, the read reference voltages predicted using the normal-Laplace-based model deviate significantly from the actual optimal read reference voltages. We find that the normal-Laplace-based model has difficulty converging to a good value at 0 P/E cycles, while our Student's t-based model does not experience any such difficulty.

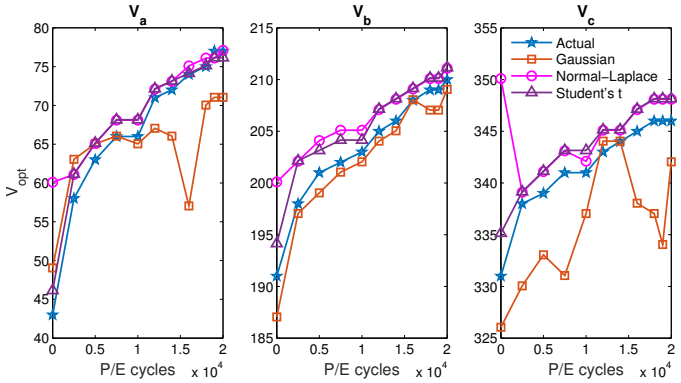


Fig. 16: Actual and modeled *optimal* read reference voltages (V_{opt}) using the three evaluated threshold voltage distribution models at different P/E cycle counts.

Figure 17 shows the RBER when we use the actual optimal read reference voltage to read data, as well as the RBER when we use the optimal read reference voltages predicted by each

[5]Note that the default read reference voltages are (V_a,V_b,V_c) = (50,190,330). We observe that the actual optimal read reference voltage can be higher than the default read reference voltage by as much as 27 voltage steps.

of the three static models from Section 4, at different P/E cycle counts. As we did for Figure 15, we show the average error rate for only the LSB pages, only the MSB pages, and for all of the pages. We observe that the prediction generated from the Gaussian-based model results in a significantly higher MSB error rate than the actual optimal voltage. The normal-Laplace-based and our Student's t-based models generate read reference voltage predictions that result in near-optimal RBER (within 1.5% and 1.1%, respectively, of the optimal RBER), despite some difference between the actual optimal read reference voltage and the model-predicted voltages.

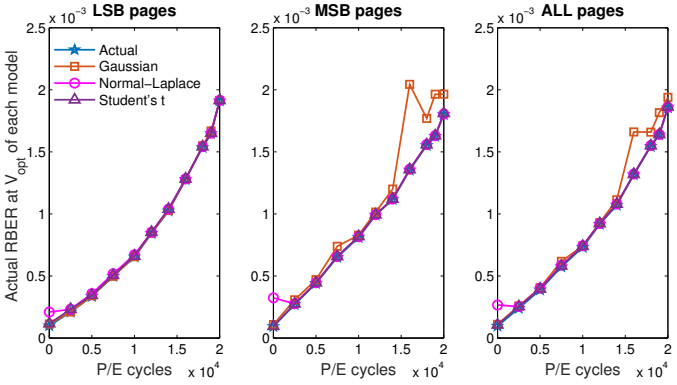


Fig. 17: RBER achieved by actual and modeled *optimal* read reference voltages (V_{opt}) using the three evaluated threshold voltage distribution models at different P/E cycle counts.

We evaluate how using the optimal voltages predicted by each model can improve flash lifetime compared to using the default read reference voltages. We assume that we have a state-of-the-art LDPC decoder, which can tolerate a raw bit error rate as high as 5×10^{-3} [13] while still keeping the required uncorrectable error rate below 10^{-15} [16] during the flash device's lifetime.[6] We also assume that our flash device refreshes its data every three weeks, limiting the number of retention and read disturb errors that occur [5]. Using the actual (i.e., ideal) optimal read reference voltage, flash lifetime improves by 50.6%. Both our Student's t-based model and the normal-Laplace-based model come very close to the ideal improvement, providing a *48.9% lifetime improvement* with our Student's t-based model. Due to its lower accuracy, the Gaussian-based model achieves only a 38.5% improvement.

6.3. Expected Lifetime Estimation

Due to the increasing raw bit error rate at higher P/E cycle counts, flash memory can endure only a limited number of writes. To make sure that all data stored in the flash drive is reliable over the course of a predefined device lifetime (typically several years), enterprise users limit the number of writes to each flash drive. Due to process variation, different flash chips can have different raw bit error rates and thus different P/E cycle endurance. However, flash vendors conservatively set the flash drive's P/E cycle endurance to the *worst case* (i.e., to the lowest endurance value out of all of the chips that they produce), as they do not know how fast each individual flash chip wears out over time. In fact, prior work has tested six commercial flash drives, and found that they all surpassed their official endurance specifications by an average of 81% [12].

If the flash controller can monitor how fast each flash chip wears out due to flash writes, the users can determine the *actual* endurance limit of the flash drive, and write more data to it without worrying about prematurely wearing out the drive and losing data. The model proposed in this paper enables raw bit error rate prediction for *future P/E cycle counts*. Thus, the controller can predict the endurance limit of each flash chip by iterating through our dynamic model to predict the point at which the raw bit error rate exceeds the ECC correction capability (i.e., when the lifetime *actually* ends). The flash controller then communicates this prediction to the file system to allow higher write intensity to the flash drive.

We estimate the lifetime improvements using this technique, with the same assumptions we made in Section 6.2 and the data shown in Section 6.1. With our dynamic model, we safely achieve *69.9% higher P/E cycle endurance* than manufacturer specification. This translates to 69.9% more tolerable writes per day, if we assume that the flash device will be used for the same number of years (i.e., lifetime) as before.

7. Related Work

To our knowledge, this paper is the first to (1) propose a threshold voltage distribution model that is both highly accurate and computationally efficient, (2) propose a dynamic threshold voltage distribution model that predicts how the parameters of this model change with increasing program/erase cycle count, and (3) demonstrate several new practical uses of this threshold voltage distribution model within a flash controller to improve flash memory reliability.

We have already comprehensively compared our Student's t-based static model to the two most relevant models based on real characterization results, the Gaussian-based model [8, 27] and the normal-Laplace-based model [35], in Sections 4.1, 4.2, and 6. We show that our Student's t-based model has an error rate within 0.11% of the error rate of the highly-accurate normal-Laplace model, while requiring 4.41x less computation time. Several prior works fit the threshold voltage distribution to other models that are either less accurate or more complex, such as the beta distribution [8], gamma distribution [8], log-normal distribution [8], Weibull distribution [8], and beta-binomial probability distribution [40]. Other prior works model the threshold voltage distribution based on idealized circuit-level models [10, 27, 31]. These models capture some of the desired threshold voltage distribution behavior, but are less accurate than those derived from real characterization.

A few works also propose dynamic models of the threshold voltage distribution shifts based on the power law [4, 8, 35]. While these models are sufficient for offline analysis, they are unsuitable for deployment in today's flash controllers, as they fail to achieve high accuracy and low computational complexity at the same time. Our dynamic model also uses the power

[6] To tolerate variation in the raw bit error rate, we assume that 10% of the total ECC correction capability is reserved, lowering the maximum tolerable raw bit error rate.

law, but is based on our new, accurate, and low-complexity Student's t-based static model. We show that our model has an error rate of only 2.72% when estimating the distribution at 20K P/E cycles, even though it uses characterization data collected at only four different P/E cycle counts from the past (up to 10K P/E cycles). While other dynamic models based on idealized circuit models exist [10, 31], they are not validated with real characterization data, and cannot achieve the same accuracy as our model.

Prior works propose and evaluate techniques for raw bit error rate estimation [35, 36], optimal read reference voltage estimation [3, 32, 33, 39], and LDPC soft decoding [10, 41, 44]. These works utilize a threshold voltage distribution model only offline, or do not utilize a threshold voltage distribution model at all. We show that, by utilizing our model, we can effectively and practically guide such flash reliability mechanisms *online* in the flash controller. We also provide a new mechanism to *exploit* process variation for *higher* flash endurance, by predicting and safely utilizing the remaining lifetime of a flash device online. Prior works propose to only *tolerate* error rate variation and process variation to improve flash lifetime [21, 28, 29].

We note that several prior works have already extensively studied the impact of retention behavior on the threshold voltage distribution using real hardware [2, 3, 5, 6]. They show that commonly-employed refresh mechanisms in flash devices can successfully mitigate most of the impact of retention on the threshold voltage distribution [5, 6, 24]. As a result, we expect that even without capturing the effects of retention, our proposed threshold voltage distribution model will work well in practice.

8. Future Work

We believe that there are many other applications of our online threshold voltage distribution model, in addition to the examples we present in Section 6. For example, we envision future work that attains further flash lifetime improvements by using our online model to estimate soft information for a commonly-used error correction technique known as low-density parity check (LDPC) codes [11] (Section 8.1). We also envision future work that demonstrates the potential performance benefits of the applications we discussed in Section 6, as well as work that develops new applications of our online model to further improve performance (Section 8.2). We briefly discuss a high-level overview of these two directions in this section.

8.1. Soft Information Estimation for LDPC Codes

To tolerate flash errors more efficiently, today's flash controllers use LDPC codes to detect and correct multiple raw bit errors in the data read from the flash memory channel [10, 41, 44]. An LDPC code can use *soft information* about each bit to increase the probability of correcting the raw bit errors. This soft information is provided by the flash controller, which estimates the probability of each bit being a 1 or a 0 using the threshold voltage of a cell. A modern flash controller typically obtains this probability from a Gaussian-based model for the threshold voltage distribution, since the soft information can be computed as a quadratic function of the threshold voltage. However, as we have shown in Section 4, the Gaussian-based model underestimates the probability density at the tail of the distribution, and does not model program errors. Thus, the model can provide inaccurate information to the LDPC decoder. This compromises the error correction capability of the LDPC codes, thus reducing the reliability and performance of the flash drive.

With the models proposed in this paper, we now have an accurate threshold voltage distribution model that adapts to the P/E cycle of each block, and can be implemented within the flash controller. Using our Student's t-based model, we can accurately and efficiently compute the probability density for any threshold voltage range to provide *accurate* soft information to the flash controller. By increasing the accuracy of this soft information, we effectively increase the error correction capability of the LDPC code, which can lead to longer flash lifetime and better read performance [10,41,44]. We leave the precise implementation of such a mechanism for future work.

8.2. Improving Flash Performance

While the applications of our threshold voltage distribution model that we have discussed in Sections 6 and 8.1 aim to improve reliability, they can also improve flash performance. For example, by predicting and applying the optimal read reference voltage (Section 6.2), we can greatly lower the probability that read-retries need to be performed for a read operation, which also reduces the number of ECC decoding iterations, both of which lead to a lower read latency [3]. Other applications can also take advantage of our model to improve flash performance. For example, we can minimize the ECC decoding latency by adaptively applying a weaker ECC code when the raw bit error rate indicated by our model is low [13, 14, 42, 43]. We expect and hope future work to evaluate the performance benefits of these applications, and to propose other new applications of our online model that can improve flash performance.

9. Conclusion

This paper introduces a new threshold voltage distribution model for modern NAND flash memory devices. Our model is based on a new experimental characterization of the threshold voltage distribution and how it shifts over time using state-of-the-art 1X-nm MLC NAND flash chips. Our characterization shows that the threshold voltage distribution can be approximated using our modified version of the Student's t-distribution, and that the amount by which the distribution shifts as the P/E cycle count increases is governed by the power law. Our new model, which combines these two observations in its static and dynamic components, is capable of accurately capturing the current and predicting the future threshold voltage distribution of flash memory cells. We show that our model achieves low modeling error, and is computationally simple enough to implement online in a

flash controller. We demonstrate various applications of our model in a flash controller. We show that these applications improve flash lifetime by 48.9% and/or enable the flash device to safely utilize 69.9% more P/E cycles than manufacturer specification. We conclude that our proposed threshold voltage distribution model for modern MLC NAND flash memory devices is practical and effective. We hope that our paper inspires future work to improve upon our online flash channel model, and to develop and evaluate new techniques that take advantage of such a model to increase flash memory reliability and performance.

Acknowledgments

We thank the anonymous reviewers for their feedback. This work is partially supported by the Intel Science and Technology Center, the CMU Data Storage Systems Center, NSF grants 1212962 and 1320531, and gifts from Intel and Seagate.

References

[1] Y. Cai, E. F. Haratsch, M. P. McCartney, and K. Mai, "FPGA-Based Solid-State Drive Prototyping Platform," in *FCCM*, 2011.

[2] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," in *DATE*, 2012.

[3] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai, and O. Mutlu, "Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery," in *HPCA*, 2015.

[4] Y. Cai, O. Mutlu, E. F. Haratsch, and K. Mai, "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation," in *ICCD*, 2013.

[5] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, and K. Mai, "Flash Correct and Refresh: Retention Aware Management for Increased Lifetime," in *ICCD*, 2012.

[6] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, A. Cristal, O. Unsal, and K. Mai, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," *Intel Technology Journal (ITJ)*, 2013.

[7] Y. Cai, G. Yalcin, O. Mutlu, E. F. Haratsch, O. Unsal, A. Cristal, and K. Mai, "Neighbor Cell Assisted Error Correction in MLC NAND Flash Memories," in *SIGMETRICS*, 2014.

[8] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis, and Modeling," in *DATE*, 2013.

[9] Y. Cai, Y. Luo, S. Ghose, E. F. Haratsch, K. Mai, and O. Mutlu, "Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery," in *DSN*, 2015.

[10] G. Dong, N. Xie, and T. Zhang, "Enabling NAND Flash Memory Use Soft-Decision Error Correction Codes at Minimal Read Latency Overhead," *IEEE Trans. Circuits Syst. I, Reg. Papers*, 2013.

[11] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, 1962.

[12] G. Gasior, "The SSD Endurance Experiment: They're All Dead," http://techreport.com/review/27909/the-ssd-endurance-experiment-theyre-all-dead. 2015.

[13] E. F. Haratsch, "LDPC Code Concepts and Performance on High-Density Flash Memory," in *Flash Memory Summit*, 2014.

[14] E. F. Haratsch, "Controller Concepts for 1y/1z nm and 3D NAND Flash," in *Flash Memory Summit*, 2015.

[15] Q. Huang, S. Lin, and K. A. Abdel-Ghaffar, "Error-Correcting Codes for Flash Coding," *IEEE Trans. Inf. Theory*, 2011.

[16] JEDEC Solid State Technology Assn., "Failure Mechanisms and Models for Semiconductor Devices," *JEDEC Publication JEP122-B*, 2003.

[17] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Advanced Flash Technology Status, Scaling Trends & Implications to Enterprise SSD Technology Enablement," in *Flash Memory Summit*, 2012.

[18] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Lifetime Improvement of NAND Flash-Based Storage Systems Using Dynamic Program and Erase Scaling," in *FAST*, 2014.

[19] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, 1951.

[20] C. Lee, S.-K. Lee, S. Ahn, J. Lee, W. Park, Y. Cho, C. Jang, C. Yang, S. Chung, I.-S. Yun *et al.*, "A 32-Gb MLC NAND Flash Memory with Vth Endurance Enhancing Schemes in 32 nm CMOS," *JSSC*, 2011.

[21] J. Li, K. Zhao, J. Ma, and T. Zhang, "Realizing Unequal Error Correction for NAND Flash Memory at Minimal Read Latency Overhead," *IEEE Trans. Circuits Syst. II, Express Briefs*, 2014.

[22] Q. Li, H. Chang, A. Jiang, and E. F. Haratsch, "Joint Decoding of Content-Replication Codes for Flash Memories," in *Allerton*, 2015.

[23] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing NAND Flash-Based SSDs via Retention Relaxation," in *FAST*, 2012.

[24] Y. Luo, Y. Cai, S. Ghose, J. Choi, and O. Mutlu, "WARM: Improving NAND Flash Memory Lifetime with Write-Hotness Aware Retention Management," in *MSST*, 2015.

[25] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "An Accurate and Practical Threshold Voltage Distribution Model for Modern MLC NAND Flash Memory," Carnegie Mellon Univ., SAFARI Research Group, Tech. Rep. 2016-006, 2016.

[26] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit Error Rate in NAND Flash Memories," in *IRPS*, 2008.

[27] R. Motwani, "Estimation of Flash Memory Level Distributions Using Interpolation Techniques for Optimizing the Read Reference," in *GLOBECOM*, 2015.

[28] R. Motwani and C. Ong, "Design of LDPC Coding Schemes for Exploitation of Bit Error Rate Diversity Across Dies in NAND Flash Memory," in *ICNC*, 2013.

[29] R. Motwani and C. Ong, "Soft Decision Decoding of RAID Stripe for Higher Endurance of Flash Memory Based Solid State Drives," in *ICNC*, 2015.

[30] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, 1965.

[31] Y. Pan, G. Dong, and T. Zhang, "Exploiting Memory Device Wear-Out Dynamics to Improve NAND Flash Memory System Performance," in *FAST*, 2011.

[32] N. Papandreou, T. Parnell, H. Pozidis, T. Mittelholzer, E. Eleftheriou, C. Camp, T. Griffin, G. Tressler, and A. Walls, "Using Adaptive Read Voltage Thresholds to Enhance The Reliability of MLC NAND Flash Memory Systems," in *GLSVLSI*, 2014.

[33] N. Papandreou, T. Parnell, H. Pozidis, T. Mittelholzer, E. Eleftheriou, C. Camp, T. Griffin, G. Tressler, and A. Walls, "Enhancing the Reliability of MLC NAND Flash Memory Systems by Read Channel Optimization," *ACM TODAES*, 2015.

[34] K.-T. Park, M. Kang, D. Kim, S.-W. Hwang, B. Y. Choi, Y.-T. Lee, C. Kim, and K. Kim, "A Zeroing Cell-to-Cell Interference Page Architecture with Temporary LSB Storing and Parallel MSB Program Scheme for MLC NAND Flash Memories," *JSSC*, 2008.

[35] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, "Modelling of the Threshold Voltage Distributions of Sub-20nm NAND Flash Memory," in *GLOBECOM*, 2014.

[36] A. Prodromakis, S. Korkotsides, and T. Antonakopoulos, "MLC NAND Flash Memory: Aging Effect and Chip/Channel Emulation," *Microprocessors and Microsystems*, 2015.

[37] W. J. Reed, "The Normal-Laplace Distribution and Its Relatives," in *Advances in Distribution Theory, Order Statistics, and Inference*. Springer, 2006.

[38] M. Speigel, "Theory and Problems of Probability and Statistics," *McGraw-Hill*, 1992.

[39] H. Tabrizi, B. Peleato, R. Agarwal, and J. Ferreira, "Improving NAND Flash Read Performance Through Learning," in *ICC*, 2015.

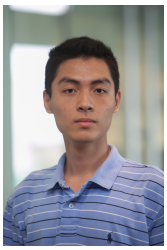
[40] V. Taranalli, H. Uchikawa, and P. H. Siegel, "Channel Models For Multi-Level Cell Flash Memories Based on Empirical Error Analysis," 2016. http://arxiv.org/abs/1602.07743

[41] J. Wang, K. Vakilinia, T.-Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel, "Enhanced Precision Through Multiple Reads for LDPC Decoding in Flash Memories," *JSAC*, 2014.

[42] G. Wu, X. He, N. Xie, and T. Zhang, "DiffECC: Improving SSD Read Performance Using Differentiated Error Correction Coding Schemes," in *MASCOTS*, 2010.

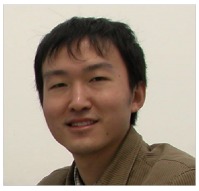
[43] L. Yuan, H. Liu, P. Jia, and Y. Yang, "An Adaptive ECC Scheme for Dynamic Protection of NAND Flash Memories," in *ICASSP*, 2015.

[44] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "LDPC-in-SSD: Making Advanced Error Correction Codes Work Effectively in Solid State Drives," in *FAST*, 2013.



Yixin Luo received his B.S.E. in Computer Engineering from the University of Michigan, Ann Arbor, and his B.S.E. in Electrical Engineering from Shanghai Jiao Tong University, in 2012. He is a Ph.D. candidate at Carnegie Mellon University (CMU). His research at CMU mainly focuses on DRAM and flash reliability, and datacenter reliability and cost optimization. He received the best paper award and the best paper runner-up award from the IEEE International Symposium on High-Performance Computer Architecture in 2012 and in 2015, respectively.

Saugata Ghose is a Systems Scientist in the Department of Electrical and Computer Engineering at Carnegie Mellon University. He is a member of the SAFARI Research Group, led by Dr. Onur Mutlu. His current research interests include application- and system-aware memory and storage systems, flash reliability, architectural solutions for large-scale systems, GPUs, and emerging memory technologies. He received his Ph.D. and M.S. from Cornell University, where he was the recipient of the NDSEG Fellowship and the ECE Director's Ph.D. Teaching Assistant Award, and received dual B.S. degrees in Computer Science and in Computer Engineering from Binghamton University.



Yu Cai received his Ph.D. from Carnegie Mellon University in Computer Engineering. He obtained his M.S. degree from Tsinghua University in Electronic Engineering and his B.S. degree from Beijing University of Posts and Telecommunications in Telecommunication Engineering. He has worked as a solid-state disk system architect at SK Hynix, Seagate Technology, Avago Technologies, and LSI Corporation. Prior to that, he worked on wireless communications at the Hong Kong Applied Science and Technology Research Institute (ASTRI), Alcatel-Lucent and Microsoft Research Asia (MSRA). He holds 30+ US patents, and is the author of 20+ peer-reviewed papers. He received the best paper runner-up award from the IEEE International Symposium on High-Performance Computer Architecture in 2015.

Erich F. Haratsch is Director of Engineering, Firmware Architecture for Flash Controllers at Seagate Technology. He leads the development of advanced features for best-in class SSD performance, endurance, NAND flash management, signal processing and error correction coding for solid-state drive controllers. Before joining Seagate, Haratsch was Director of Engineering at LSI, where he pioneered advanced signal processing and LDPC-based error correction algorithms for solid-state drive controllers. Earlier in his career, Haratsch developed signal processing and error correction technologies for several generations of HDD controllers at LSI and Agere Systems, which shipped in more than one billion chips. He previously worked at Bell Labs, where he invented new equalizer and decoder architectures for Gigabit Ethernet over copper and optical communications. Haratsch is a frequent speaker at leading industry events, is the author of over 40 peer-reviewed journal and conference papers, and holds more than 100 US patents. He earned his M.S. and Ph.D. degrees from the Technical University of Munich (Germany).

Onur Mutlu is a Full Professor of Computer Science at ETH Zurich. He is also a faculty member at Carnegie Mellon University, where he previously held the William D. and Nancy W. Strecker Early Career Professorship. His current broader research interests are in computer architecture, systems, and bioinformatics. He is especially interested in interactions across domains and between applications, system software, compilers, and microarchitecture, with a major current focus on memory and storage systems. He obtained his Ph.D. and M.S. in ECE from the University of Texas at Austin and B.S. degrees in Computer Engineering and Psychology from the University of Michigan, Ann Arbor. His industrial experience spans starting the Computer Architecture Group at Microsoft Research (2006-2009), and various product and research positions at Intel Corporation, Advanced Micro Devices, and VMware. He received the inaugural IEEE Computer Society Young Computer Architect Award, the inaugural Intel Early Career Faculty Award, faculty partnership awards from various companies, and a healthy number of best paper or "Top Pick" paper recognitions at various computer systems and architecture venues. His computer architecture course lectures and materials are freely available on YouTube, and his research group makes software artifacts freely available online. For more information, please see his webpage at http://www.ece.cmu.edu/~omutlu.