Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery

Yu Cai, Yixin Luo, Saugata Ghose, Erich F. Haratsch*, Ken Mai, Onur Mutlu
Carnegie Mellon University, *Seagate Technology
yucaicai@gmail.com, {yixinluo, ghose, kenmai, onur}@cmu.edu

***Abstract*—NAND flash memory reliability continues to degrade as the memory is scaled down and more bits are programmed per cell. A key contributor to this reduced reliability is *read disturb*, where a read to one row of cells impacts the threshold voltages of *unread* flash cells in different rows of the same block. Such disturbances may shift the threshold voltages of these unread cells to different logical states than originally programmed, leading to read errors that hurt endurance.**

For the first time in open literature, this paper experimentally characterizes read disturb errors on state-of-the-art 2Y-nm (i.e., 20-24 nm) MLC NAND flash memory chips. Our findings (1) correlate the magnitude of threshold voltage shifts with read operation counts, (2) demonstrate how program/erase cycle count and retention age affect the read-disturb-induced error rate, and (3) identify that lowering pass-through voltage levels reduces the impact of read disturb and extend flash lifetime. Particularly, we find that the probability of read disturb errors increases with both higher wear-out and higher pass-through voltage levels.

We leverage these findings to develop two new techniques. The first technique mitigates read disturb errors by dynamically tuning the pass-through voltage on a per-block basis. Using real workload traces, our evaluations show that this technique increases flash memory endurance by an average of 21%. The second technique recovers from previously-uncorrectable flash errors by identifying and probabilistically correcting cells susceptible to read disturb errors. Our evaluations show that this recovery technique reduces the raw bit error rate by 36%.

Keywords*—*NAND flash memory; read disturb; error tolerance

1. Introduction

NAND flash memory currently sees widespread usage as a storage device, having been incorporated into systems ranging from mobile devices and client computers to datacenter storage, as a result of its increasing capacity. Flash memory capacity increase is mainly driven by aggressive transistor scaling and multi-level cell (MLC) technology, where a single flash cell can store more than one bit of data. However, as its capacity increases, flash memory suffers from different types of circuit-level noise, which greatly impact its reliability. These include program/erase cycling noise [2,3], cell-to-cell program interference noise [2,5,8], retention noise [2,4,6,7,23,24], and read disturb noise [11,14,24,33]. Among all of these types of noise, *read disturb* noise has largely been understudied in the past for MLC NAND flash, with no open-literature work available today that characterizes and analyzes the read disturb phenomenon.

One reason for this neglect has been the heretofore low occurrence of read-disturb-induced errors in older flash technologies. In single-level cell (SLC) flash, read disturb errors were only expected to appear after an average of one million reads to a single flash block [10,14]. Even with the introduction of MLC flash, first-generation MLC devices were expected to exhibit read disturb errors after 100,000 reads [10,15]. As a result of process scaling, some modern MLC flash devices are now prone to read disturb errors after as few as 20,000 reads, with this number expected to drop even further with continued scaling [10,15]. The exposure of these read disturb errors can be exacerbated by the uneven distribution of reads across flash blocks in contemporary workloads, where certain flash blocks experience high temporal locality and can, therefore, more rapidly exceed the read count at which read disturb errors are induced.

Read disturb errors are an intrinsic result of the flash architecture. Inside each flash cell, data is stored as the *threshold voltage* of the cell, based on the logical value that the cell represents. During a read operation to the cell, a *read reference voltage* is applied to the transistor corresponding to this cell. If this read reference voltage is higher than the threshold voltage of the cell, the transistor is turned *on*. Within a flash block, the transistors of multiple cells, each from a different flash page, are tied together as a single *bitline*, which is connected to a single output wire. Only one cell is read at a time per bitline. In order to read one cell (i.e., to determine whether it is turned *on* or *off*), the transistors for the cells *not being read* must be kept *on* to allow the value from the cell being read to propagate to the output. This requires the transistors to be powered with a *pass-through voltage*, which is a read reference voltage guaranteed to be higher than *any* stored threshold voltage. Though these other cells are *not* being read, this high pass-through voltage induces electric tunneling that can shift the threshold voltages of these unread cells to higher values, thereby *disturbing the cell contents on a read operation to a neighboring page*. As we scale down the size of flash cells, the transistor oxide becomes thinner, which in turn increases this tunneling effect. With each read operation having an increased tunneling effect, it takes fewer read operations to neighboring pages for the unread flash cells to become disturbed (i.e., shifted to higher threshold voltages) and move into a different logical state.

In light of the increasing sensitivity of flash memory to read disturb errors, our goal in this paper is to (1) develop a thorough understanding of read disturb errors in state-of-the-art MLC NAND flash memories, by performing experimental characterization of such errors on existing commercial 2Y-nm (i.e. 20-24 nm) flash memory chips, and (2) develop mechanisms that can tolerate read disturb errors, making use of insights gained from our read disturb error characterization. The *key findings* from our quantitative characterization are:

- The effect of read disturb on threshold voltage distributions and raw bit error rates increases with both the number of reads to neighboring pages and the number of program/erase cycles on a block (Sec. 3.2 and 3.3).
- Cells with lower threshold voltages are more susceptible to errors as a result of read disturb (Sec. 3.2).
- As the pass-through voltage decreases, (1) the read disturb effect of each individual read operation becomes smaller, but (2) the read errors can increase due to reduced ability in allowing the read value to pass through the unread cells (Sec. 3.4, 3.5, and 3.6).
- If a page is recently written, a significant margin within the ECC correction capability is unused (i.e., the page can still tolerate more errors), which enables the page's pass-through voltage to be lowered safely (Sec. 3.7).

We exploit these studies on the relation between the read disturb effect and the pass-through voltage (V_{pass}), to design two mechanisms that reduce the impact of read disturb. First, we propose a low-cost dynamic mechanism called V_{pass} *Tuning*, which, for each block, finds the lowest pass-through voltage that retains data correctness. V_{pass} *Tuning* extends flash endurance by exploiting the finding that a lower V_{pass} reduces the read disturb error count (Sec. 4). Second, we propose *Read Disturb Recovery* (RDR), a mechanism that exploits the differences in the susceptibility of different cells to read disturb to extend the effective correction capability of error-correcting codes (ECC). RDR probabilistically identifies and corrects cells susceptible to read disturb errors (Sec. 5).

To our knowledge, this paper is the first to make the following *contributions*:

- We perform a detailed experimental characterization of how the threshold voltage distributions for flash cells get distorted due to the read disturb phenomenon.
- We propose a new technique to mitigate the errors that are induced by read disturb effects. This technique dynamically tunes the pass-through voltage on a per-block basis to minimize read disturb errors. We evaluate the proposed read disturb mitigation technique on a variety of real workload I/O traces, and show that it increases flash memory endurance by 21%.
- We propose a new mechanism that can probabilistically identify and correct cells susceptible to read disturb errors. This mechanism can reduce the flash memory raw bit error rate by up to 36%.

2. Background and Related Work

In this section, we first provide some necessary background on storing and reading data in NAND flash memory. Next, we discuss read disturb, a type of error induced by neighboring read operations, and describe its underlying causes.

2.1. Data Storage in NAND Flash

NAND Flash Cell Threshold Voltage Range. A flash memory cell stores data in the form of a *threshold voltage*, the lowest voltage at which the flash cell can be switched *on*. As illustrated in Fig. 1, the threshold voltage (V_{th}) range of a 2-bit MLC NAND flash cell is divided into four regions by three reference voltages, V_a, V_b, and V_c. The region in which the threshold voltage of a flash cell falls represents the cell's current state, which can be ER (or erased), P1, P2, or P3. Each state decodes into a 2-bit value that is stored in the flash cell (e.g., 11, 10, 00, or 01). We represent this 2-bit value throughout the paper as a tuple (LSB, MSB), where LSB is the least significant bit and MSB is the most significant bit. Note that the threshold voltage of all flash cells in a chip is bounded by an upper limit, V_{pass}, which is the *pass-through voltage*.

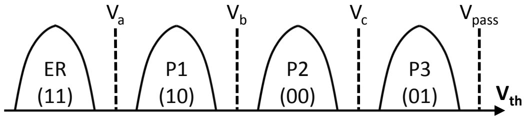

Fig. 1. Threshold voltage distribution in 2-bit MLC NAND flash. Stored data values are represented as the tuple (LSB, MSB).

NAND Flash Block Organization. A NAND flash memory chip is organized as thousands of two-dimensional arrays of flash cells, called *blocks*. Within each block, as illustrated in Fig. 2a, all the cells in the same row share a *wordline* (WL), which typically spans 32K to 64K cells. The LSBs stored in a wordline form the *LSB page*, and the MSBs stored in a wordline form the *MSB page*. Within a block, all cells in the same column are connected in series to form a *bitline* or *string* (BL in Fig. 2a). All cells in a bitline share a common ground on one end, and a common sense amplifier on the other for reading the threshold voltage of one of the cells when decoding data.

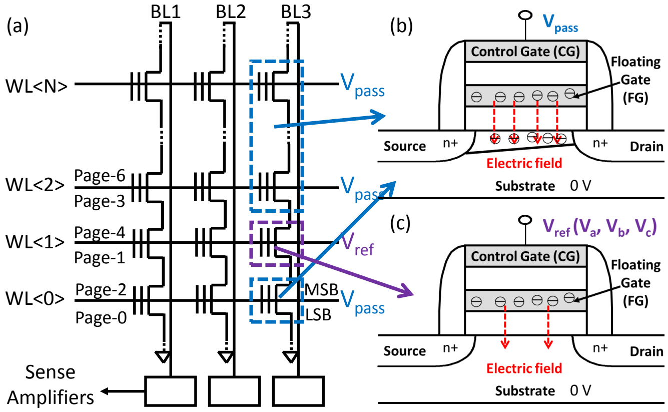

Fig. 2. (a) NAND flash block structure. (b/c) Diagrams of floating gate transistors when different voltages (V_{pass}/V_{ref}) are applied to the wordline.

NAND Flash Read Operation. A NAND flash read operation is performed by applying a *read reference voltage* V_{ref} one or more times to the wordline that contains the data to be read, and sensing whether the cells on the wordline are switched *on* or not. The applied V_{ref} is chosen from the reference voltages V_a, V_b, and V_c, and changes based on which page (i.e., LSB or MSB) we are currently reading.

To read an LSB page, only one read reference voltage, V_b, needs to be applied. If a cell is in the ER or P1 state, its threshold voltage is lower than V_b, hence it is switched *on*. If a cell is in the P2 or P3 state, its threshold voltage is higher than V_b, and the cell is switched *off*. The sense amplifier can then determine whether the cell is switched *on* or *off* to read the data in this LSB page. To read the MSB page, two read reference voltages, V_a and V_c, need to be applied in sequence to the wordline. If a cell turns *off* when V_a is applied *and* turns *on* when V_c is applied, we determine that the cell contains a threshold voltage V_{th} where $V_a < V_{th} < V_c$, indicating that it is in either the P1 or P2 state and holds an MSB value of 0 (see Fig. 1). Otherwise, if the cell is *on* when V_a is applied *or off* when V_c is applied, the cell is in the ER or P3 state, holding an MSB value of 1.

As we mentioned before, the cells on a bitline are connected in series to the sense amplifier. In order to read from a single cell on the bitline, *all of the other cells* on the same bitline must switched *on* to allow the value being read to propagate through to the sense amplifier. We can achieve this by applying the *pass-through voltage* onto the wordlines of *unread cells*. Modern flash memories guarantee that *all* unread cells are *passed through* (i.e., the maximum possible threshold voltage, V_{pass}, is applied to the cells) to minimize errors during the read operation. We will show, in Sec. 3.6, that this choice is conservative: applying a single worst-case pass-through voltage to *all* cells is not necessary for correct operation.

2.2. Read Disturb

Read disturb is a well-known phenomenon in NAND flash memory, where reading data from a flash cell can cause the threshold voltages of other (unread) cells in the same block to shift to a higher value [2, 11, 14, 15, 24, 33]. While a single threshold voltage shift is small, such shifts can accumulate over time, eventually becoming large enough to alter the state of some cells and hence generate *read disturb errors*.

The failure mechanism of a read disturb error is similar to the mechanism of a normal *program* operation. A *program* operation applies a high programming voltage (+10V) to the cell to shift its threshold voltage to the desired range. Similarly, a read operation applies a high pass-through voltage (∼+6V) to all other cells that share the same bitline with the cell being read. Although the pass-through voltage is not as high as the programming voltage, it still generates a "weak programming" effect on the cells it is applied to, which can unintentionally shift their threshold voltages.

2.3. Circuit-Level Impacts of Read Disturb

At the circuit level, as illustrated in Fig. 2b and 2c, a NAND flash memory cell is essentially a floating gate transistor with its control gate (CG) connected to the wordline, and its source and drain connected to (or shared with) its neighboring cells. A floating gate transistor, compared to an ordinary transistor, adds a floating gate (FG, as shown in Fig. 2b and 2c) beneath the CG. The amount of charge stored in the FG determines the threshold voltage of the transistor.

Electrical charge is injected to the FG during a read disturb or a program operation through an effect called *Fowler-Nordheim (FN) tunneling* [12], which creates an electric tunnel between the FG and the substrate. The FN tunnel is triggered by the electric field passing through the tunnel (E_{ox}). Note that the strength of this electric field is proportional to the voltage applied on the CG and the amount of charge stored in the FG. The current density through the FN tunnel (J_{FN}) can be modeled as [12]:

$$J_{FN} = \alpha_{FN} E_{ox}^2 e^{-\beta_{FN}/E_{ox}} \tag{1}$$

We observe from Eq. (1)[1] that the FN tunneling current increases with E_{ox} super-linearly. Since the pass-through voltage is much lower than the programming voltage, the tunneling current induced by a single read disturb is much smaller than that of a program operation. With a lower current, each individual read disturb injects charge into the FG at a lower rate, resulting in a slower threshold voltage shift than during a program operation.

Unfortunately, the actual effect of read disturb is exacerbated by the accumulation of read counts within the same block. Today's flash devices are fast enough to sustain more than 100,000 read operations in 1 minute [30]. The threshold voltage change generated by each read operation within the same block can accumulate to lead to a read disturb error. Also, a single read operation can disturb *all other pages within the same block*. As the block size increases further in the future, read disturb errors are more likely to happen [15].

2.4. Related Work on Read Disturb

To date, the read disturb phenomenon for NAND flash has not been well explored in openly-available literature. Prior work on mitigating NAND flash read disturb errors has proposed to leverage the flash controller, either by caching recently read data to avoid a read operation [32], or by maintaining a cumulative per-block read counter and rewriting the contents of a block whenever the counter exceeds a predetermined threshold [13]. The Read Disturb-Aware FTL identifies those pages which incur the most reads using the flash translation layer (FTL), and moves these pages to a new block [15].

Two mechanisms are currently being implemented within Yaffs (Yet Another Flash File System) to handle read disturb errors, though they are not yet available [10]. The first mechanism is similar to the Read Disturb-Aware FTL [15], where a block is rewritten after a fixed number of page reads are performed to the block (e.g., 50,000 reads for an MLC chip). The second mechanism periodically inserts an additional read (e.g., a read every 256 block reads) to a page within the block, to check whether that page has experienced a read disturb error, in which case the page is copied to a new block.

All of these proposals are orthogonal to our read disturb mitigation techniques, and can be combined with our work for even greater protection. None of these works perform device-level experimental characterization of the read disturb phenomenon, which we provide extensively in this paper.[2]

3. Read Disturb Characterization

In this section, we describe a series of observations and characterizations that were performed using commercially-available 2Y-nm MLC NAND flash chips. We first identify trends directly related to the magnitude of perturbations that take place during read disturb (Sec. 3.2). Next, we determine the frequency at which errors occur in modern flash devices as a result of the read disturb phenomenon (Sec. 3.3). We then examine the effect of changing the pass-through voltage, V_{pass}, on the voltage shifts that result from read disturb (Sec. 3.4). We also identify other errors that can result from changing V_{pass} (Sec. 3.6), and show how many of these errors can be tolerated by error correction mechanisms in modern flash devices (Sec. 3.7). These characterizations are used in Sec. 4 to drive our read disturb mitigation mechanism that tunes V_{pass}, and in Sec. 5 for our read disturb error recovery mechanism.

3.1. Characterization Methodology

We use an FPGA-based NAND flash testing platform in order to characterize state-of-the-art flash chips [1]. We use the *read-retry* operation present within MLC NAND flash devices to accurately read the cell threshold voltage [3, 4, 6, 29]. As threshold voltage values are proprietary information, we present our results using a *normalized threshold voltage*, where the nominal value of V_{pass} is equal to 512 in our normalized scale, and where 0 represents GND.

One limitation of using commercial flash devices is the inability to alter the V_{pass} value, as no such interface currently exists. We work around this by using the read-retry mechanism, which allows us to change the read reference voltage V_{ref} one wordline at a time. Since both V_{pass} and V_{ref} are applied to wordlines, we can mimic the effects of changing V_{pass} by instead changing V_{ref} and examining the impact on the wordline being read. We perform these experiments on one wordline per block, and repeat them over ten different blocks.

3.2. Quantifying Read Disturb Perturbations

Our first goal is to measure the amount of threshold voltage shift that takes place inside a flash cell due to read disturb. These measurements are performed by first programming known pseudo-randomly generated data values into a selected flash block. Using read-retry techniques [3, 29], the initial threshold voltages are measured for all flash cells in the block. Then, we select a single page from the block to read, and perform N repeated read operations on it. After the N reads, we measure the threshold voltage for every flash cell in the

[1] α_{FN} and β_{FN} are material-specific constants.

[2] Recent work experimentally characterizes and proposes solutions for read disturb errors in DRAM [19]. The mechanisms for disturbance and techniques to mitigate them are different between DRAM and NAND flash due to device-level differences.

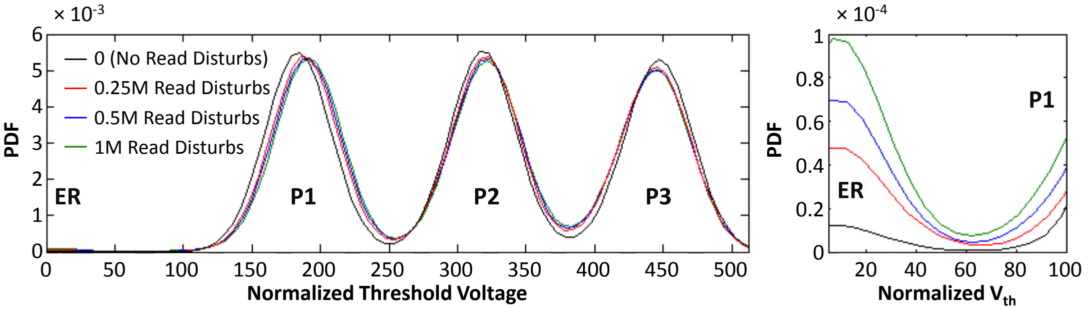

Fig. 3. (a) Threshold voltage distribution of all states before and after read disturb; (b) Threshold voltage distribution between erased state and P1 state.

block to determine how much the threshold voltage for each cell shifted. We repeat this process to measure the distribution shift over an increasing number of read disturb occurrences.

Fig. 3a shows the distribution of the threshold voltages for cells in a flash block after 0, 250K, 500K, and 1 million read operations. Fig. 3b zooms in on this to illustrate the distribution for values in the ER state.[3] We observe that *states with lower threshold voltages are slightly more vulnerable to shifts than states with higher threshold voltages*. This is due to applying the *same* voltage (V_{pass}) to *all* cells during a read disturb operation, regardless of their threshold voltages. A lower threshold voltage on a cell induces a larger voltage difference $(V_{pass} - V_{th})$ through the tunnel, and in turn generates a stronger tunneling current, making the cell more vulnerable to read disturb.

The degree of the threshold voltage shift is broken down further in Fig. 4, where we group cells by their initially-programmed state. The figure demonstrates the shift in mean threshold voltage for each group, as the number of read disturb occurrences increases due to more reads being performed to the block over time. Fig. 4a shows that *for cells in the ER state, there is a systematic shift of the cell threshold voltage distribution to the right* (i.e., to higher values), *demonstrating a significant change as a result of read disturb*. In contrast, the increases for cells starting in the P1 (Fig. 4b) and P2 (Fig. 4c) states are much more restricted, showing how the read disturb effect becomes less prominent as V_{th} increases (as explained above). For the P3 state, as shown in Fig. 4d, we actually observe a decrease in the mean V_{th}. This decrease is due to the effects of *retention loss* arising from charge leakage. As data is held within each flash cell, the stored charge slowly leaks over time, with a different rate of leakage across different flash cells due to both process variation and uneven wear. For cells in the P3 state, the effects of read disturb are minimal, and so we primarily see the retention-caused drop in threshold voltage (which is small).[4] For cells starting in other states, the read disturb phenomenon outweighs leakage due to retention loss, resulting in increases in their means. Again, cells in the ER state are most affected by read disturb.

Fig. 5 shows the change in the standard deviation of the threshold voltage, again grouped by the initial threshold voltage of the cell, after an increasing number of read disturb occurrences. For cells starting in the P1, P2, and P3 states, we observe an increased spread in the threshold voltage distribution, a result of both uneven read disturb effects and uneven retention loss. For the ER state, we actually observe a slight reduction in the deviation, which is a result of our measurement limitations: cells in the ER state often have a negative V_{th}, but we can only measure non-negative values of V_{th}, so the majority of these cells do not show up in our distributions.

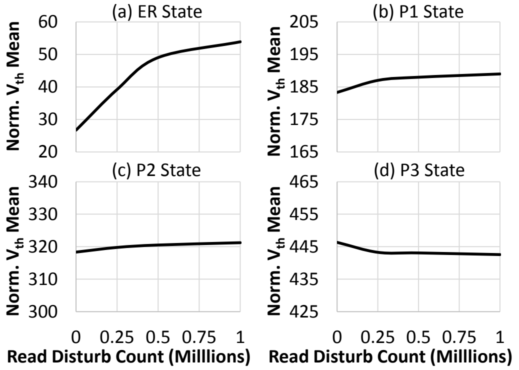

Fig. 4. Mean value of normalized cell threshold voltage, as the read disturb count increases over time. Distributions are separated by cell states.

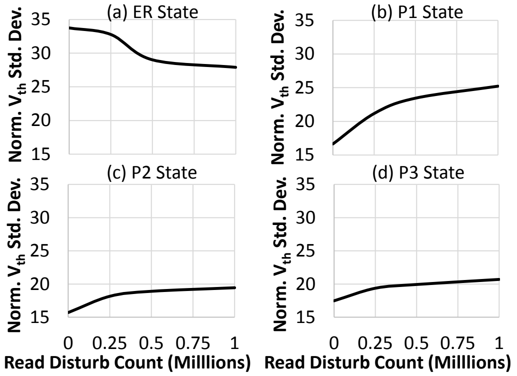

Fig. 5. Standard deviation of normalized cell threshold voltage, as the read disturb count increases over time. Distributions are separated by cell states.

We conclude that *the magnitude of the threshold voltage shift for a cell due to read disturb (1) increases with the number of read disturb operations, and (2) is higher if the cell has a lower threshold voltage*.

3.3. Effect of Read Disturb on Raw Bit Error Rate

Now that we know how much the threshold voltage shifts due to read disturb effects, we aim to relate these shifts to the *raw bit error rate* (RBER), which refers to the probability of reading an incorrect state from a flash cell. We see that *for a given amount of P/E cycle wear on a block, the raw bit error rate increases roughly linearly with the number of read disturb operations*. Fig. 6 shows the RBER over an increasing number of read disturb operations for different amounts of P/E cycle wear on flash blocks. Each level shows a linear RBER increase as the read disturb count increases.

[3] For now, we use a flash block that has experienced 8,000 program/erase (P/E) cycles. We will show sensitivity to P/E cycles in Sec. 3.3.

[4] Retention loss effects are observable in these results because it takes approximately two hours to perform 200K read operations, due to the latency between the flash device and the FPGA host software.

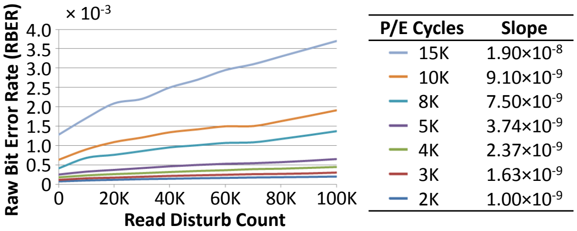

Fig. 6. Raw bit error rate vs. read disturb count under different levels of P/E cycle wear.

We also observe that *the effects of read disturb are greater for cells that have experienced a larger number of P/E cycles*. In Fig. 6, the *derivative* (i.e., slope) of each line grows with the number of P/E cycles at roughly a quadratic rate. This is an effect of the wear caused with each additional P/E cycle, where the probability of charge getting trapped within the transistor oxide increases and the insulating abilities of the dielectric degrade [26]. As a result, when V_{pass} is applied to the transistor gate during a read disturb operation, the degraded dielectric allows additional electrons to be injected through the tunnel into the floating gate. This results in a greater degree of threshold voltage shift for each read disturb operation.

It is important to note that flash correct-and-refresh mechanisms [6, 7, 22, 23, 25, 28] can provide *long-term* correction of read disturb errors. These refresh mechanisms periodically take the contents of a flash block and program them to a new block, in effect resetting the impact of retention loss and read disturbs. However, the refresh frequency is typically limited, as each refresh operation forces an additional erase and program operation on a block, thereby increasing wear. For the purposes of our studies, we assume that refreshes take place after a retention period of one week (i.e., one week after programming) [6,7], and thus we focus on the number of read disturb errors that can occur over the course of seven days.

3.4. Pass-Through Voltage Impact on Read Disturb

As we saw in Sec. 3.2, the effects of read disturb worsen for cells whose threshold voltages are further from V_{pass}. In fact, when we observe the raw bit errors that result from read disturb, we find that the majority of these errors are from cells that were programmed in the ER state but shift into the P1 state due to read disturb. We have already discussed that a lower value of V_{th} increases the impact of read disturb, assuming a fixed value of V_{pass}. In this subsection, we will quantitatively show how the difference $(V_{pass} - V_{th})$ affects the magnitude of FN tunneling that takes place, which directly correlates with (and affects) the magnitude of the threshold voltage shift due to read disturb.

Fig. 7a shows the internal design of the floating gate cell in NAND flash. The floating gate holds the charge of a flash cell, which is set to a particular threshold voltage V_{th} when the floating gate is programmed. The control gate is used to read or reprogram the value held within the floating gate. The control gate and floating gate are separated by an insulator, reoxidized nitrided SiO$_2$ (ONO), which has an effective capacitance of C_{ono} and a thickness of t_{ono}. Between the floating gate and the substrate lies the tunneling oxide, whose effective capacitance is C_{ox} and whose thickness is t_{ox}. The substrate has a constant intrinsic voltage, which we refer to as V_{thi}.

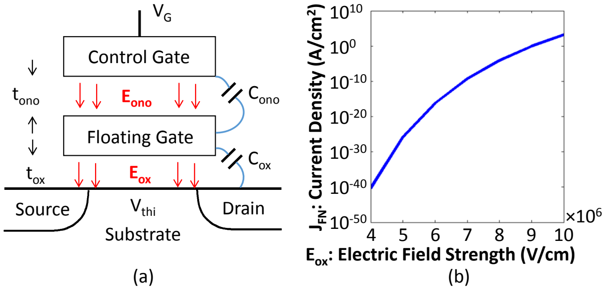

Fig. 7. (a) Electrical parameters within a flash cell; (b) Correlation between J_{FN} (current in tunnel oxide) and E_{ox} (electric field strength) from Eq. (1).

When a positive voltage (V_G) is applied to the control gate, two electric fields are induced: one flowing from the control gate to the floating gate (E_{ono}), and another flowing from the floating gate to the substrate (E_{ox}). As we mentioned in Sec. 2.3, the electric field E_{ox} through the tunnel oxide is a function of both the voltage applied at the control gate and the charge stored inside the floating gate:

$$E_{ox} = \frac{C_{ono}}{C_{ono} + C_{ox}} \times [(V_G - V_{thi}) - V_{th}] \times \frac{1}{t_{ox}} \quad (2)$$

We derive E_{ox} by determining the component of the electrical field induced due to the voltage differential between the control gate and the floating gate, by using the voltage equations $V = Et$ and $Q = VC$. During a read disturb operation, $V_G = V_{pass}$. As a result, the strength of the electrical field E_{ox} is a linear function of $(V_{pass} - V_{th})$.

Fig. 7b illustrates the relationship between the current density of the FN tunnel (J_{FN}) and E_{ox}, which we derive from Eq. (1). Note that the y-axis is in log scale. The figure shows that J_{FN} grows super-linearly with E_{ox}. As E_{ox} is a linear function of $(V_{pass} - V_{th})$, the key insight is that *either a decrease in V_{th} or an increase in V_{pass} results in a super-linear increase in the current density*, i.e., the tunneling effect that causes read disturb. This relationship demonstrates why voltage threshold shifts are much worse for cells in the erased state in Sec. 3.2 than for cells in the other states, as the erased state has a much higher value of $(V_{pass} - V_{th})$, assuming a fixed V_{pass} for all cells. As a higher $(V_{pass} - V_{th})$ increases the impact of read disturb, we want to reduce this voltage difference. Even a small decrease in $(V_{pass} - V_{th})$ can significantly reduce the tunneling current density (see Fig. 7b), and hence the read disturb effects. We use this insight to drive the next several characterizations, which identify the feasibility and potential of lowering V_{pass} to reduce the effects of read disturb.

To summarize, we have shown that the cause of read disturb can be reduced by reducing the pass-through voltage. Our goal is to exploit this observation to mitigate read disturb effects.

3.5. Constraints on Reducing Pass-Through Voltage

There are several constraints that restrict the range of potential values for V_{pass} in a flash chip. All of these constraints must be taken into account if we are to change the V_{pass} value to reduce read disturb. Traditionally, a single V_{pass} value is used globally for the entire chip, and the value of V_{pass} must be higher than all potential threshold voltages within the chip. Due to the charge leakage that occurs during data retention, the threshold voltage of each cell slowly decreases over time. The specific rate of leakage can vary across flash cells, as a function of both process variation and uneven wear-leveling. If we can identify the *slowest* leaking cell in the *entire* flash chip, we may be able to globally decrease V_{pass} over time to reduce the effects of read disturb.

To observe whether the slowest leaking cell leaks fast enough to yield any meaningful V_{pass} reduction, we perform experiments on a flash block that has incurred 8,000 P/E cycles, and study the drop in threshold voltage over *retention age* (i.e., the length of time for which the data has been stored in the flash block). Unfortunately, in a 40-day study, there was no significant change in normalized threshold voltage for the slowest leaking cell, as shown in Fig. 8. This is despite the fact that the mean threshold voltage for a cell in the P3 state dropped to 437, which is much lower than the lowest observed threshold voltage (503) in Fig. 8. (The slowest leaking cell has a threshold voltage 6σ higher than the mean.)

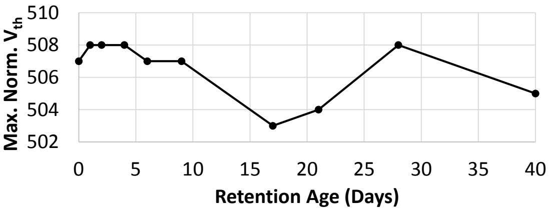

Fig. 8. Maximum threshold voltage within a block with 8K P/E cycles of wear vs. retention age, at room temperature.

In order to successfully lower the value of V_{pass}, we must turn to a mechanism where V_{pass} can be set *individually for each flash block*. The minimum V_{pass} value for a block only needs to be larger than the maximum threshold voltage within that block. This is affected by two things: different blocks are likely to have different maximum threshold voltages because they may have (1) different amounts of P/E cycle wear, or (2) different levels of V_{th} due to process variation effects. Therefore, we conclude that *a mechanism that provides a per-block value of V_{pass} must be able to adjust this value dynamically based on the current properties of the block, to ensure that the V_{pass} selected for each block is greater than the maximum V_{th} in that block.*

3.6. Effect of Pass-Through Voltage on Raw Bit Error Rate

Even when V_{pass} is selected on a per-block basis, it may make sense to reduce V_{pass} to a value *below* the maximum V_{th} within the block, to further reduce the effects of read disturb. Our goal is to characterize and understand how this reduction affects the raw bit error rate.

Setting V_{pass} to a value slightly lower than the maximum V_{th} leads to a tradeoff. On the one hand, it can substantially reduce the effects of read disturb. On the other hand, it causes a small number of unread cells to incorrectly stay *off* instead of passing through a value, potentially leading to a *read error*. Therefore, if the number of read disturb errors can be dropped significantly by lowering V_{pass}, the small number of read errors introduced may be warranted.[5] Naturally, this trade-off depends on the magnitude of these error rate changes. We now explore the gains and costs, in terms of overall RBER, for relaxing V_{pass} *below* the maximum threshold voltage of a block.

We first describe how relaxing V_{pass} increases the RBER as a result of read errors. Fig. 9a demonstrates an example using a three-wordline flash block. For each cell in Fig. 9a, the threshold voltage value of the cell is labeled. When we attempt to read the value stored in the middle wordline, V_{pass} is applied to the top and bottom wordlines. Let us assume that we are performing the first step of the read operation, setting the read reference voltage V_{ref} to V_b (2.5V for this example). The four cells of our selected wordline turn their transistors *off*, *off*, *on*, and *off*, respectively, and we should read the correct data value 0010 from the LSBs. If V_{pass} is set to 5V (higher than any of the threshold values of the block), the transistors for our unread

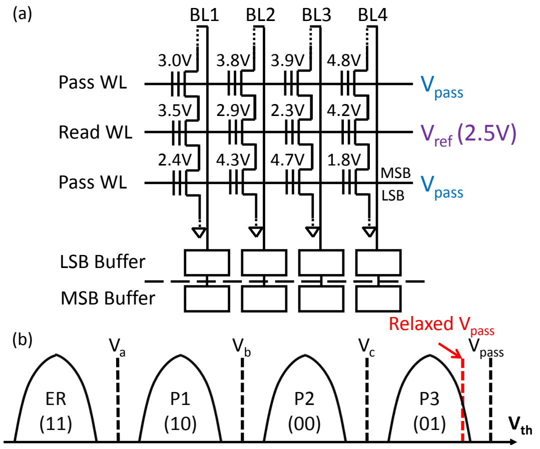

Fig. 9. (a) Example three-wordline flash block with threshold voltages assigned to each cell; (b) Illustration of how bit errors can be introduced when relaxing V_{pass} below its nominal voltage.

cells are all turned *on*, allowing values from the wordline being read to pass through successfully.

Let us explore what happens if we relax V_{pass} to 4.6V, as shown in Fig. 9b. The first two bitlines (BL1 and BL2) in Fig. 9a are unaffected, since all of the threshold voltages on the transistors of their unread cells are less than 4.6V, and so these transistors on BL1 and BL2 still turn *on* (as they should). However, the third bitline (BL3) exhibits an error. The transistor for the bottom cell in BL3 is now turned *off*, since V_{pass} is lower than its threshold voltage. In this case, *a read error is introduced*: the cell in the wordline being read was turned *on*, yet our incorrectly turned *off* bottom cell prevents the value from passing through properly. If we examine the fourth bitline (BL4), the top cell is also turned *off* now due to the lower value of V_{pass}. This case, however, *does not produce an error*, since the cell being read would have been turned *off* anyways (as its V_{th} is greater than V_{ref}). As a result of our relaxed V_{pass}, instead of reading the correct value 0010, we now read 0000. Note that this single-bit error may still be correctable by ECC.

To identify the extent to which relaxing V_{pass} affects the raw bit error rate, we experimentally sweep over V_{pass}, reading the data after a range of different retention ages, as shown in Fig. 10. First, we observe that across all of our studied retention ages, *V_{pass} can be lowered to some degree without inducing any read errors*. For greater relaxations, though, the error rate increases as more unread cells are incorrectly turned *off* during read operations. We also note that, *for a given V_{pass} value, the additional read error rate is lower if the read is performed a longer time after the data is programmed into the flash (i.e., if the retention age is longer)*. This is because of the retention loss effect, where cells slowly leak charge and thus have lower threshold voltage values over time. Naturally, as the threshold voltage of every cell decreases, a relaxed V_{pass} becomes more likely to correctly turn *on* the unread cells.

We now quantify the potential reduction in RBER when a relaxed V_{pass} is used to reduce the effects of read disturb. When performing this characterization, we must work around the current flash device limitation that V_{pass} cannot be altered by the controller. We overcome this limitation by using the read-retry mechanism to emulate a reduced V_{pass} to a single wordline. For these experiments, after we program pseudo-random data to the cells, we set the *read reference voltage* to the relaxed V_{pass} value. We then repeatedly read the LSB page of our selected wordline for N times, where N is the number of neighboring wordline reads we want to emulate (which,

[5] If too many read errors occur, we can always fall back to using the maximum threshold voltage for V_{pass} without consequence; see Sec. 4.4.

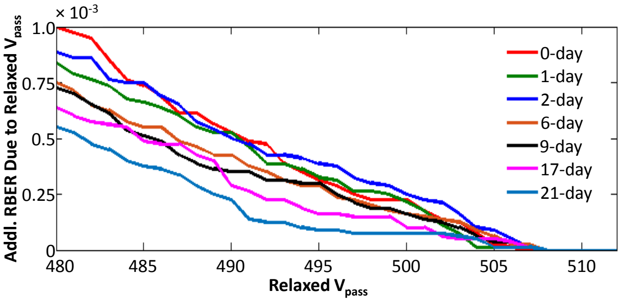

Fig. 10. Additional raw bit error rate induced by relaxing V_{pass}, shown across a range of data retention ages.

in practice, would apply our relaxed V_{pass} to this selected wordline). We then measure the RBER for both the LSB and MSB pages of our selected wordline by applying the default values of read reference voltages (V_a, V_b, and V_c) to it.

Fig. 11 shows the change in RBER as a function of the number of read operations, for selected relaxations of V_{pass}. Note that the x-axis uses a log scale. *For a fixed number of reads, even a small decrease in the V_{pass} value can yield a significant decrease in RBER.* As an example, at 100K reads, lowering V_{pass} by 2% can reduce the RBER by as much as 50%. Conversely, *for a fixed RBER, a decrease in V_{pass} exponentially increases the number of tolerable read disturbs.* This is also shown in Table 1, which lists the increased ratio of read disturb errors a flash device can tolerate in its lifetime (while RBER $\leq 1.0\times10^{-3}$ [6, 7]) with a lowered V_{pass}. This result is consistent with our model in Sec. 3.4, where we find a super-linear relationship between $(V_{pass} - V_{th})$ and the induced tunneling effect (which affects read disturbs). We conclude that *reducing V_{pass} per block can greatly reduce the RBER due to read disturb.*

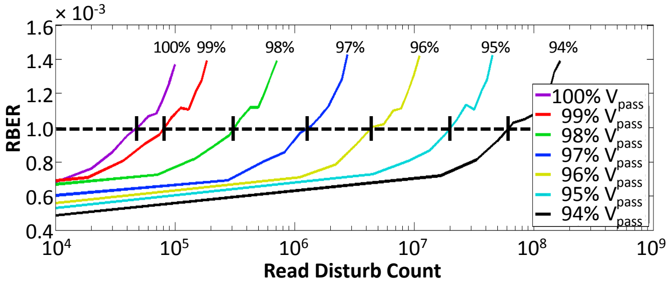

Fig. 11. Raw bit error rate vs. read disturb count for different V_{pass} values, for flash memory under 8K P/E cycles of wear.

Table 1. Tolerable read disturb count at different V_{pass} values, normalized to the tolerable read disturb count for nominal V_{pass} (512).

Pct. V_{pass} Value	100%	99%	98%	97%	96%	95%	94%
Rd. Disturb Cnt.	1x	1.7x	6.8x	22x	100x	470x	1300x

3.7. Error Correction with Reduced Pass-Through Voltage

So far, we have examined how read disturb count and pass-through voltage affect the raw bit error rate. While we have shown in Sec. 3.6 that V_{pass} can be lowered to some degree without introducing new raw bit errors, we would ideally like to further decrease V_{pass} to lower the read disturb impact more. This can enable flash devices to tolerate many more reads, as we demonstrated in Fig. 11.

Modern flash memory devices experience a limited number of raw bit errors, which come from a number of sources: erase errors, program errors, errors caused by program interference from neighboring cells, retention errors, and read disturb errors [2,7,24]. As flash memories guarantee a minimum level of error-free non-volatility, modern devices include error correcting codes (ECC) that are used to fix raw bit errors [21]. Depending on the number of ECC bits used, an ECC mechanism can provide a certain *error correction capability* (i.e., the total number of bit errors it can correct for a single read). If the number of bit errors in a read flash page is below this capability, ECC delivers error-free data. However, if the number of errors exceeds the ECC capability, the correction mechanism cannot successfully correct the data in the read page. As a result, the amount of ECC protection must cover the total number of raw bit errors expected in the device. ECC capability is practically limited, as a greater capability requires additional ECC bits (and therefore greater storage, power consumption, and latency overhead [6, 7]) per flash page.

In this subsection, our goal is to identify how many additional raw bit errors the current level of ECC provisioning in flash chips can sustain. With room to tolerate additional raw bit errors, we can further decrease V_{pass} without fear of delivering incorrect data. A typical flash device is considered to be error-free if it guarantees an uncorrectable bit error rate of less than 10^{-15}, which corresponds to traditional data storage reliability requirements [16, 21]. For an ECC mechanism that can correct 40 bits of errors for every 1K bytes, the acceptable raw bit error rate to meet the reliability requirements is 10^{-3} [6, 7].

Fig. 12 shows how the expected RBER changes over a 21-day period for our tested flash chip *without read disturb*, using a block with 8,000 P/E cycles of wear. Unsurprisingly, as retention age increases, retention errors increase, driving up the RBER [2, 4, 24]. However, when the retention age is low, the retention error rate is also low, as is the overall raw bit error rate, resulting in significant *unused* ECC correction capability.

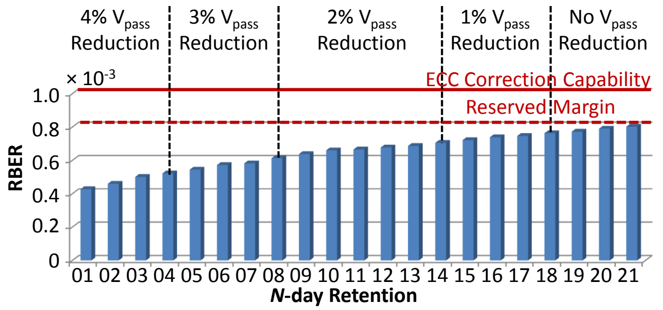

Fig. 12. Overall raw bit error rate and tolerable V_{pass} reduction vs. retention age, for a flash block with 8K P/E cycles of wear.

Based on our analysis in Sec. 3.6, we can fill in the unused ECC correction capability with read errors introduced by relaxing V_{pass}, which would allow the flash memory to tolerate more read disturbs. As we illustrate in Fig. 12, an RBER margin (20% of the total ECC correction capability) is reserved to account for the variation in the distribution of errors and other potential errors (e.g., program and erase errors). For each retention age, we record the maximum percentage of *safe* V_{pass} reduction (i.e., the lowest value of V_{pass} at which all read errors can still be corrected by ECC) compared to the default pass-through voltage (V_{pass} = 512). This percentage is listed on the top of Fig. 12. As we can see, by exploiting the previously-unused ECC correction capability, V_{pass} can be safely reduced by as much as 4% when the retention age is low (less than 4 days). Since the amount of previously-unused ECC correction capability decreases over retention age, V_{pass} must be increased for reads to remain correctable.

Our key insight from this study is that *a lowered V_{pass} can reduce the effects of read disturb, and that the read errors induced from lowering V_{pass} can be tolerated by the built-in error correction mechanism within modern flash controllers*. Using this insight, in Sec. 4, we design a mechanism that can dynamically tune the V_{pass} value, based on the characteristics of each flash block and the age of the data stored within it.

3.8. Summary of Key Characterization Results

From our characterization, we make the following major conclusions: (1) The magnitude of threshold voltage shifts due to read disturb increases for larger values of $(V_{pass} - V_{th})$; hence, minimizing V_{pass} can greatly reduce such threshold voltage shifts; (2) Blocks with greater wear (i.e., more P/E cycles) experience larger threshold voltage shifts due to read disturb; (3) While reducing V_{pass} can reduce the raw bit errors that occur as a result of read disturb, it can introduce other errors that affect reliability; (4) The over-provisioned correction capability of ECC can allow us to reliably decrease V_{pass} on a per-block basis, as long as the decreases are dynamically adjusted as the age of the data grows to tolerate increasing retention errors.

4. Mitigation: Pass-Through Voltage Tuning

In Sec. 3, we made a number of new observations about the read disturb phenomenon. We now propose *V_{pass} Tuning*, a new technique that exploits those observations to mitigate NAND flash read disturb errors, by tuning the pass-through voltage (V_{pass}) for each flash block. The key idea is to reduce the number of read disturb errors by shrinking $(V_{pass} - V_{th})$ as much as possible, where V_{th} is the value stored within a flash cell. Our mechanism trades off read disturb errors for the read errors that are introduced when lowering V_{pass}, but these read errors can be corrected using the unused portion of the ECC correction capability.

4.1. Motivation

NAND flash memory typically uses ECC to correct a certain number of raw bit errors within each page, as we discussed in Sec. 3.7. As long as the total number of errors does not exceed the ECC correction capability, the errors can be corrected and the data can be successfully read. When the retention age of the data is low, we find that the retention error rate (and therefore the overall raw bit error rate) is much lower than the rate at high retention ages (see Fig. 12), resulting in significant unused ECC correction capability.

Fig. 13 provides an exaggerated illustration of how this unused ECC capability changes over the retention period (i.e., the *refresh interval*). At the start of each retention period, there are no retention errors or read disturb errors, as the data has just been restored. In these cases, the large unused ECC capability allows us to design an *aggressive* read disturb mitigation mechanism, as we can safely *introduce correctable errors*. Thanks to read disturb mitigation, we can reduce the effect of each individual read disturb, thus lowering the total number of read disturb errors accumulated by the end of the refresh interval. This reduction in read disturb error count leads to lower *error count peaks* at the end of each refresh interval, as shown in Fig. 13 by the distance between the solid black line and the dashed red line. Since flash lifetime is dictated by the number of data errors (i.e., when the total number of errors exceeds the ECC correction capability, the flash device has reached the end of its life), lowering the error count peaks extends lifetime by extending the time before these peaks exhaust the ECC correction capability.

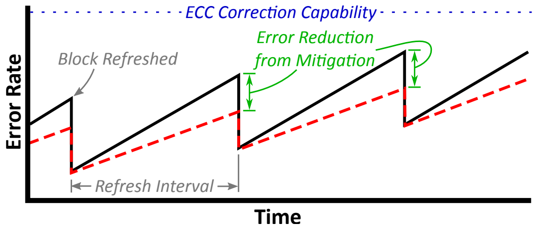

Fig. 13. Exaggerated example of how read disturb mitigation reduces error rate peaks for each refresh interval. Solid black line is the unmitigated error rate, and dashed red line is the error rate after mitigation. (Note that the error rate does not include read errors introduced by reducing V_{pass}, as the unused error correction capability can tolerate errors caused by *V_{pass} Tuning*.)

4.2. Mechanism Overview

We reduce the flash read disturb errors by relaxing V_{pass} when the block's retention age is low, thus minimizing the impact of read disturb. Recall from Sec. 3 that reducing V_{pass} has two major effects: (1) a read operation may fail if V_{pass} is lower than the V_{th} of any cell on the bitline; (2) reducing V_{pass} can significantly decrease the read disturb effect for each read operation. If we aggressively lower V_{pass} when a block has a low retention age (which is hopefully possible without causing uncorrectable read errors due to the large unused ECC correction capability at low retention age), the accumulated read disturb errors are minimal when the block reaches a high retention age. This makes it much less likely for read disturbs to generate an uncorrectable error, thus leading to overall flash lifetime improvement.

To minimize the effect of read disturb, we propose to *learn the minimum pass-through voltage for each block, such that all data within the block can be read correctly with ECC*. Our learning mechanism works online and is triggered on a daily basis. *V_{pass} Tuning* can be fully implemented within the *flash controller*, and has two components:

1. It first finds the size of the ECC margin M (i.e., the unused correction capability within ECC) that can be exploited to tolerate additional read errors for each block. In order to do this, our mechanism discovers the page with *approximately* the highest number of raw bit errors (Sec. 4.3).
2. Once it knows the available margin M, our mechanism calibrates the pass-through voltage V_{pass} *on a per-block basis* to find the lowest value of V_{pass} that introduces no more than M additional raw errors (Sec. 4.4).

4.3. Identifying the Available ECC Margin

To calculate the available ECC margin M, our mechanism must first approximately discover the page with the highest error count. While finding the page in each block with the *exact* highest error count can be costly if performed daily, we can instead statically identify, at manufacture time, a page in each block that will *approximately* have the greatest number of errors. Flash devices generally exhibit two types of errors: those based on dynamic factors (e.g., retention, read disturb) and those based on static factors (e.g., process variation). Within a block, there is likely to be little variation in the number of errors based on dynamic factors, as all pages in the block are of similar retention age and experience similar read disturb counts and P/E cycles. Additionally, modern flash devices randomize their data internally to improve endurance and encrypt their contents [9,18], which leads to the stored data values across the pages to be similar. Therefore, the mitigation mechanism can be simplified to identify the page in each block that exhibits the greatest number of errors occurring due to static factors (as these factors remain relatively constant over the device lifetime), which we call the *predicted worst-case page*.

After manufacturing, we statically find the predicted worst-case page by programming pseudo-randomly generated data to each page within the block, and then immediately reading the page to find the error count, as prior work on error analysis has done [2]. (ECC provides an error count whenever a page is read.) For each block, we record the page number of the page with the highest error count.

While we find the predicted worst-case page only once for each block after the flash device is manufactured, our mechanism must still count the number of errors within this page once daily, to account for the increasing number of errors due to dynamic factors. It can obtain the error count, which we define as our *maximum estimated error* (MEE), by performing *a single read* to this page and reading the error count provided by ECC (once a day).

Since we only *estimate* the maximum error count instead of finding the *exact* maximum, and as new retention and read disturb errors appear within the span of a day, we conservatively reserve 20% of the spare ECC correction capability in our calculations. Thus, if the maximum number of raw bit errors correctable by ECC is C, we calculate the available ECC margin for a block as $M = (1 - 0.2) \times C - MEE$.

4.4. Tuning the Pass-Through Voltage

The second part of our mechanism identifies the greatest V_{pass} reduction that introduces no more than M raw bit errors. The general V_{pass} *identification process* requires three steps:

Step 1: Aggressively reduce V_{pass} to $V_{pass} - \Delta$, where Δ is the smallest resolution by which V_{pass} can change.

Step 2: Apply the new V_{pass} to *all* wordlines in the block. Count the number of 0's read from the page (i.e., the number of bitlines incorrectly switched *off*, as described in Sec. 3.6) as N. If $N \leq M$ (recall that M is the extra available ECC correction margin), the read errors resulting from this V_{pass} value can be corrected by ECC, so we repeat Steps 1 and 2 to try to further reduce V_{pass}. If $N > M$, it means we have reduced V_{pass} too aggressively, so we proceed to Step 3 to roll back to an acceptable value of V_{pass}.

Step 3: Increase V_{pass} to $V_{pass} + \Delta$, and verify that the introduced read errors can be corrected by ECC (i.e., $N \leq M$). If this verification fails, we repeat Step 3 until the read errors are reduced to an acceptable range.

The implementation can be simplified greatly in practice, as the error rate changes are relatively slow over time (as seen in Sec. 3.7).[6] Over the course of the seven-day refresh interval, our mechanism must perform one of two actions each day:

Action 1: When a block is *not* refreshed, our mechanism checks once daily if V_{pass} should *increase*, to accommodate the slowly-increasing number of errors due to dynamic factors (e.g., retention errors, read disturb errors).

Action 2: When a block is refreshed, all retention and read disturb errors accumulated during the previous refresh interval are corrected. At this time, our mechanism checks how much V_{pass} can be *lowered* by.

For Action 1, the error count increase over time is low enough that we need to only increase V_{pass} by at most a single Δ per day (see Fig. 12). This allows us to skip Step 1 of our identification process when a block is *not* refreshed, as the number of errors does not reduce, and only perform Steps 2 and 3 once, to compare the number of errors N from using the current V_{pass} and from using $V_{pass} + \Delta$, thus requiring *no more than two reads per block daily*.

For Action 2, we *at most* need to roll back all the V_{pass} increases from Action 1 that took place during the previous refresh interval, since the number of errors that result from static factors cannot decrease. Since Action 1 is performed daily for six days, *we only need to lower V_{pass} from its current value by at most six Δ*, requiring us to perform Steps 1 and 2 no more than six times, potentially followed by performing Step 3 once. In the worst case, *only seven reads are needed*.

Our mechanism repeats the V_{pass} identification process for each block that contains valid data to learn the *minimum pass-through voltage we can use*. This allows it to adapt to the variation of maximum threshold voltage across different blocks, which results from many factors, such as process variation and retention age variation. It also repeats the entire V_{pass} learning process daily to adapt to threshold voltage changes due to retention loss [5, 8]. As such, the pass-through voltage of *all blocks* in a flash drive can be fine-tuned *continuously* to reduce read disturb and thus improve overall flash lifetime.

Fallback Mechanism. For extreme cases where the additional errors accumulating between tunings exceed our 20% margin of unused error correction capability, errors will be uncorrectable if we continue to use an aggressively-tuned V_{pass}. If this occurs, we provide a *fallback mechanism* that simply uses the default pass-through voltage (V_{pass} = 512) to correctly read the page, as V_{pass} *Tuning* does not corrupt the stored data.

4.5. Overhead

Performance. As we described in Sec. 4.3 and 4.4, only a small number of reads need to be performed for each block on a daily basis. For Action 1, which is performed six times in our seven-day refresh period, *our tuning mechanism requires a total of three reads* (one to find the margin M, and two more to tune V_{pass}). For a flash-based SSD with a 512GB capacity (containing 65,536 blocks, with a 100μs read latency), this process takes 65536×3×100μs = 19.67 sec daily *to tune the entire SSD*. For Action 2, which is performed once at the beginning of a refresh interval, our mechanism requires a maximum of eight reads (one to find M, and up to seven to tune V_{pass}; see Sec. 4.4). Assuming every block within the SSD is refreshed on the same day, the worst-case tuning latency on this day is 65536×8×100μs = 52.43 sec for the entire drive. If we average the daily overhead over all seven days of the refresh interval (assuming *distributed refresh*), *the average daily performance overhead for our 512GB SSD is 24.34 sec.*

These small latencies can be hidden by performing the tuning in the background when the SSD is idle. We conclude that the performance overhead of V_{pass} *Tuning* is negligible.

Hardware. V_{pass} *Tuning* takes advantage of the existing read-retry mechanism (used to control the read reference voltage V_{ref}) [3, 29] to adjust V_{pass}, since both V_{ref} and V_{pass} are applied to the wordlines of a flash block. As a result, our mechanism does not require a new voltage generator. The flash device simply needs to expose an interface by which the V_{pass} value can be set by the flash controller (within which our tuning mechanism is implemented). This interface, like V_{ref}, can be tuned using an 8-bit value that represents 256 possible voltage settings.[7]

[6] While we describe and evaluate one possible pass-through voltage tuning algorithm in this paper, other, more efficient or more aggressive algorithms are certainly possible, which we encourage future work to explore. For example, we can take advantage of the monotonic relationship between pass-through voltage reduction and its resulting RBER increase to perform a binary search of the optimal pass-through voltage that minimizes the RBER.

[7] Due to the smaller range of practical voltage values for V_{pass}, as discussed in Sec. 3.5, we need to allow the selection of only the highest 256 voltage settings (out of the 512 settings possible).

Our mechanism also requires some extra storage for each block, requiring one byte to record our 8-bit tuned V_{pass} setting and a second byte to store the page number of the predicted worst-case page (we assume that each flash block contains 256 pages). For our assumed 512GB SSD, this uses a total of 65536×2B = 128KB storage overhead.

4.6. Methodology

We evaluate V_{pass} *Tuning* with I/O traces collected from a wide range of real workloads with different use cases [17,20,27,31,34], listed in Table 2. To compute flash chip *endurance* (the number of P/E cycles at which the total error rate becomes too large, resulting in an uncorrectable failure) for both the baseline and the proposed V_{pass} *Tuning* technique, we first find the block with the highest number of reads for each trace (as this block constrains the lifetime), as well as the worst-case read disturb count for that block. Next, we exploit our results from Sec. 3.7 (Table 1) to determine the equivalent read disturb count for the block with the worst-case read disturb count *after* V_{pass} *Tuning*. Finally, we use our results from Sec. 3.3 (Fig. 6) to determine the endurance. Our results faithfully take into account the effect of *all* sources of flash errors, including process variation, P/E cycling, cell-to-cell program interference, retention, and read disturb errors.

Table 2. Simulated workload traces.

Trace	Source	Max. 7-Day Read Disturb Count to a Single Block
homes	FIU [20]	511
web-vm	FIU [20]	2416
mail	FIU [20]	23612
mds	MSR [27]	36529
rsrch	MSR [27]	39810
prn	MSR [27]	40966
web	MSR [27]	41816
stg	MSR [27]	49680
ts	MSR [27]	54652
proj	MSR [27]	64480
src	MSR [27]	66726
wdev	MSR [27]	66800
usr	MSR [27]	154464
postmark	Postmark [17]	308226
hm	MSR [27]	343419
cello99	HP Labs [31]	363155
websearch	UMass [34]	611839
financial	UMass [34]	1729028
prxy	MSR [27]	2950196

4.7. Evaluation

Fig. 14 plots the P/E cycle endurance for the simulated traces. For read-intensive workloads (*postmark*, *financial*, *websearch*, *hm*, *prxy*, and *cello99*), the overall flash endurance improves significantly with V_{pass} *Tuning*. Table 2 lists the highest read disturb count for any one block within a refresh interval. We observe that workloads with higher read disturb counts see a greater improvement (in Fig. 14). As we can see in Fig. 14, the absolute value of endurance with V_{pass} *Tuning* is similar across all workloads. This is because the workloads are approaching the minimum possible number of read disturb errors, and are close to the maximum endurance improvements that read disturb mitigation can achieve. On average across all of our workloads, overall flash endurance improves by 21.0% with V_{pass} *Tuning*. We conclude that V_{pass} *Tuning* effectively improves flash endurance without significantly affecting flash performance or hardware cost.

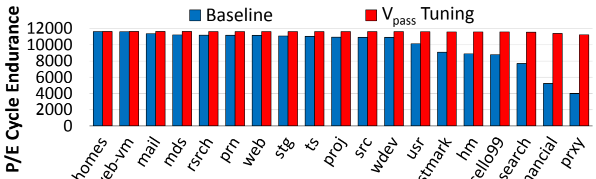

Fig. 14. Endurance improvement with V_{pass} *Tuning*.

5. Read Disturb Oriented Error Recovery

In this section, we introduce another technique that exploits our observations from Sec. 3, called *Read Disturb Recovery* (RDR). This technique recovers from an ECC-uncorrectable flash error by characterizing, identifying, and selectively correcting cells more susceptible to read disturb errors.[8]

5.1. Motivation

In Sec. 3.2, we observed that the threshold voltage shift due to read disturb is the greatest for cells in the lowest threshold voltage state (i.e., the erased state). In Fig. 15, we show example threshold voltage distributions for the erased and P1 states, and illustrate the optimal read reference voltage (V_a) between these two states, both before and after read disturb. Before read disturb occurs, the two distributions are separated by a certain voltage margin, as illustrated in Fig. 15a. In this case, V_a falls in the middle of this margin. After some number of read disturb operations, the relative threshold voltage distributions of the erased state and the P1 state shift closer to each other, eliminating the voltage margin and eventually causing the distributions to overlap, as illustrated in Fig. 15b. In this case, the optimal V_a lies at the intersection of the two distributions, as it minimizes the raw bit errors.

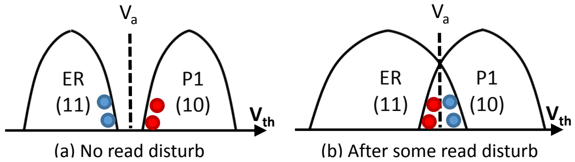

Fig. 15. V_{th} distributions before and after read disturb.

Even when the optimal V_a is applied after enough read disturbs, some cells in the erased state are misread as being in the P1 state (shown as blue cells), while some cells in the P1 state are misread as being in the erased state (shown as red cells). In these cases, errors occur, and, as we have mentioned before, consume some of the ECC error correction capability. Eventually, as these errors accumulate within a page and exceed the total ECC correction capability, the ECC can no longer correct them, resulting in an *uncorrectable flash error*. An uncorrectable flash error is the most critical type of error because (1) it determines the flash lifetime, which is the guaranteed time a flash device can be used without exceeding a fixed rate of uncorrectable errors, and (2) it may result in the permanent loss of important user data.

As we mentioned before, raw bit errors are a combination of read disturb errors and other error types, such as program errors and retention errors. If we were somehow able to correct even a fraction of the read disturb errors with a mechanism other than ECC, those now-removed errors would no longer consume part of the limited ECC correction capability. As a result, the total amount of raw bit errors that the flash device

[8] RDR can perform error recovery either online or offline. We leave the detailed exploration of the benefits and trade-offs of online vs. offline recovery to future work.

can handle would increase. This, in effect, allows previously uncorrectable flash errors to be corrected. *Thus, we would like to develop a new recovery mechanism that can identify and correct such read disturb errors.*

In order to perform such a recovery, we need to first identify *susceptible flash cells* (i.e., cells with a threshold voltage close to a read reference voltage V_{ref}) whose states are *most likely* to have been incorrectly changed due to read disturb. We do this by characterizing the degree of this threshold voltage shift. Second, we need to probabilistically correct these cells based on this threshold voltage shift characterization. To this end, we introduce our proposed mechanism, RDR, which performs these two steps to successfully recover from read disturb errors.

5.2. Identifying and Correcting Susceptible Cells

When threshold voltage distributions of two different logical states overlap due to read disturb related shifts, RDR identifies susceptible cells, and determines a threshold with which to probabilistically estimate the correct logical values of such cells.

Although read disturb is pervasive across all flash cells in a chip, we hypothesize that each cell is affected by read disturb to a different degree, due to effects such as process variation. We verify this hypothesis experimentally for $V_{ref} = V_a$. First, we program known, pseudo-randomly generated data values to a flash block with 8,000 P/E cycles of wear, and increase the read disturb count by repeatedly reading data from the block. After the first round of 250K reads, we identify susceptible cells (in this case, cells whose V_{th} is within the range $V_a \pm \sigma/2$, where σ is the standard deviation of the threshold voltage distribution). Next, we record the threshold voltages of all susceptible cells by sweeping the read reference voltage. Then, we add a second round of 100K reads, and measure the threshold voltage of the susceptible cells again. We compare the difference in threshold voltage (ΔV_{th}) for these susceptible cells between the first and second rounds, and plot the distribution of this *difference* in Fig. 16. The blue line corresponds to susceptible cells originally programmed in the erased state (cells illustrated as blue dots in Fig. 15). The red line corresponds to susceptible cells originally programmed in the P1 state (cells illustrated as red dots in Fig. 15).

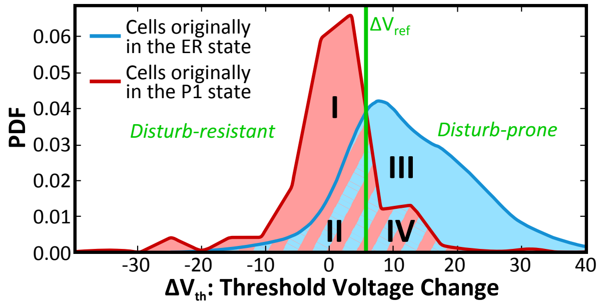

Fig. 16. Probability density function of the threshold voltage change (ΔV_{th}) for susceptible cells with threshold voltages near V_a. Cells in the area under the blue line (regions II, III, IV) were originally in the ER state, and cells in the area under the red line (regions I, II, IV) were originally in the P1 state.

Identification. As Fig. 16 shows, by setting a delta threshold voltage (ΔV_{ref}) at the intersection of the two probability density functions, we can classify all the cells into two categories. Since read disturb tends to increase a cell's threshold voltage (as is shown in Sec. 3.2), we classify cells with a higher threshold voltage change ($\Delta V_{th} > \Delta V_{ref}$; regions III and IV in Fig. 16) as *disturb-prone* cells. We classify cells with a lower or negative threshold voltage change ($\Delta V_{th} < \Delta V_{ref}$; regions I and II in Fig. 16) as *disturb-resistant* cells, as their threshold voltages either do not increase greatly or reduce, and they are therefore not likely to move upwards into a different (and incorrect) state.

Due to this disparity in the cell threshold voltage changes, some *disturb-prone* cells in the erased state are affected more by read disturb. Eventually, their threshold voltages exceed the optimal V_a, and they are misread as being in the P1 state (the blue cells in Fig. 15). In contrast, some *disturb-resistant* cells in the P1 state are affected less by read disturb. Eventually, their threshold voltages are mixed with the *disturb-prone* cells in the erased state, and they are misread as being in the erased state (red cells in Fig. 15).

Correction. After the read to a flash block has failed, if we intentionally induce more read disturbs, we can observe the amount by which the threshold voltage of a cell close to V_a shifts (i.e., we can calculate ΔV_{th} for this cell). As we did in Fig. 16, RDR can classify each cell based on the size of this shift as being either *disturb-prone* or *disturb-resistant*. Based on this classification, RDR makes two predictions. First, it predicts that a *disturb-prone* cell, whose threshold voltage has increased more rapidly, was originally programmed in the ER state, and its threshold voltage incorrectly crossed V_a. Second, it predicts that a *disturb-resistant* cell, whose threshold voltage either did not increase rapidly or decreased, was originally programmed in the P1 state, and its threshold voltage was greater than V_a *before the distributions overlapped*. RDR uses these predictions to correct the values of these susceptible cells before ECC is applied, in effect rolling back the effect of read disturb.

This technique performs a *probabilistic* correction, as we demonstrate using Fig. 16. Note that the red and blue distributions are independent, and that each represents different cells. For cells originally programmed in the ER state, a majority of them have $\Delta V_{th} > \Delta V_{ref}$ (regions III and IV under the blue line), and are hence identified as *disturb-prone*. From our first prediction above, these cells are *correctly* recovered by RDR to the ER state. In contrast, the remaining cells originally programmed in the ER state that have $\Delta V_{th} < \Delta V_{ref}$ (region II under the blue line) are identified as *disturb-resistant*, and are *incorrectly* recovered by RDR to the P1 state.

Similarly, a majority of the cells originally programmed in the P1 state have $\Delta V_{th} < \Delta V_{ref}$ (regions I and II under the red line). From our second prediction above, these cells are *correctly* recovered by RDR to the P1 state. In contrast, the remaining cells originally programmed in the P1 state that have $\Delta V_{th} > \Delta V_{ref}$ (region IV under the red line) are identified as *disturb-resistant*, and are *incorrectly* recovered to the ER state.

As we just described, RDR can sometimes incorrectly recover cells (region II under the blue line; region IV under the red line). However, it still achieves a net reduction in errors (which amounts to the area in regions I and III) because the number of cells that are correctly recovered is much greater. Incorrectly recovered cells can still be corrected later by ECC.

5.3. Mechanism

To recover from uncorrectable flash errors, we propose to use RDR to *identify those cells whose states are most likely to be changed by read disturb, and probabilistically correct those cells to reduce the overall raw bit error rate to a level correctable by ECC*. Our mechanism consists of six steps:

Step 1: When we have an uncorrectable error in a block, back up the valid, readable data in this block to another block.

Step 2: Scan the threshold voltages of the cells in the page containing the data that ECC was unable to correct, using the same methodology described in Sec. 3.1, and save the threshold voltages to another block.

Step 3: Induce additional read disturbs to this page, by repeatedly reading from another page in the same block 100K times.

Step 4: Scan and save the threshold voltages of the cells in the failed page again (same as Step 2) to another block.

Step 5: Select the cells with threshold voltages close to a read reference voltage ($V_{ref} - \sigma/2 < V_{th} < V_{ref} + \sigma/2$, and V_{ref} is set to V_a, V_b, or V_c). Calculate the change in threshold voltage for these cells before (Step 2) and after 100K read disturbs (Step 4). Set ΔV_{ref} equal to the mean of these *differences*.

Step 6: Using the ΔV_{ref} value from Step 5, predict a cell whose threshold voltage changes by more than ΔV_{ref} as *disturb-prone*, and assume it was originally programmed into the lower of the two possible cell states. Predict a cell whose threshold voltage changes by less than ΔV_{ref} as *disturb-resistant*, and assume it was originally in the higher voltage state (see Sec. 5.2). Using these state assumptions, attempt to recover the failed page using ECC.

5.4. Evaluation

We evaluate how the overall RBER changes when we use RDR. Fig. 17 shows experimental results for error recovery in a flash block with 8,000 P/E cycles of wear. When RDR is applied, the *reduction in overall RBER* grows with the read disturb count, from a few percent for low read disturb counts up to 36% for 1 million read disturb operations. As data experiences a greater number of read disturb operations, the read disturb error count contributes to a significantly larger portion of the total error count, which our recovery mechanism targets and reduces. We therefore conclude that RDR can provide a large effective extension of the ECC correction capability.

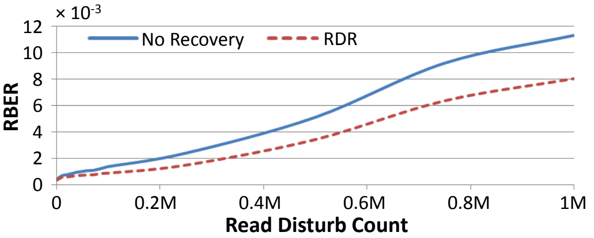

Fig. 17. Raw bit error rate vs. number of read disturb operations, with and without RDR, for a flash block with 8,000 P/E cycles of wear.

6. Conclusion

This paper provides the first detailed experimental characterization of read disturb errors for 2Y-nm MLC NAND flash memory chips. We find that bit errors due to read disturb are much more likely to take place in cells with lower threshold voltages, as well as in cells with greater wear. We also find that reducing the pass-through voltage can effectively mitigate read disturb errors. Using these insights, we propose (1) a mitigation mechanism, called V_{pass} *Tuning*, which dynamically adjusts the pass-through voltage for each flash block online to minimize read disturb errors, and (2) an error recovery mechanism, called *Read Disturb Recovery*, which exploits the differences in susceptibility of different cells to read disturb, to probabilistically correct read disturb errors. We hope that our characterization and analysis of the read disturb phenomenon enables the development of other error mitigation and tolerance mechanisms, which will become increasingly necessary as continued flash memory scaling leads to greater susceptibility to read disturb. We also hope that our results will motivate NAND flash manufacturers to add pass-through voltage controls to next-generation chips, allowing flash controller designers to exploit our findings and design controllers that tolerate read disturb more effectively.

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, and NSF grants 0953246, 1065112, 1212962, and 1320531.

References

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

[2] Y. Cai *et al.*, "Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis," in *DATE*, 2012.

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

[4] Y. Cai *et al.*, "Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery," in *HPCA*, 2015.

[5] Y. Cai *et al.*, "Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation," in *ICCD*, 2013.

[6] Y. Cai *et al.*, "Flash Correct and Refresh: Retention Aware Management for Increased Lifetime," in *ICCD*, 2012.

[7] Y. Cai *et al.*, "Error Analysis and Retention-Aware Error Management for NAND Flash Memory," *Intel Technology Journal (ITJ)*, 2013.

[8] Y. Cai *et al.*, "Neighbor Cell Assisted Error Correction in MLC NAND Flash Memories," in *SIGMETRICS*, 2014.

[9] J. Cha and S. Kang, "Data Randomization Scheme for Endurance Enhancement and Interference Mitigation of Multilevel Flash Memory Devices," *ETRI Journal*, 2013.

[10] Charles Manning, "Yaffs NAND Flash Failure Mitigation," 2012. http://www.yaffs.net/sites/yaffs.net/files/YaffsNandFailureMitigation.pdf

[11] J. Cooke, "The Inconvenient Truths of NAND Flash Memory," *Flash Memory Summit*, 2007.

[12] R. H. Fowler and L. Nordheim, "Electron Emission in Intense Electric Fields," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 1928.

[13] H. H. Frost *et al.*, "Efficient Reduction of Read Disturb Errors in NAND Flash Memory," US Patent No. 7818525. 2010.

[14] L. M. Grupp *et al.*, "Characterizing Flash Memory: Anomalies, Observations, and Applications," in *MICRO*, 2009.

[15] K. Ha *et al.*, "A Read-Disturb Management Technique for High-Density NAND Flash Memory," in *APSys*, 2013.

[16] JEDEC Solid State Technology Assn., "Failure Mechanisms and Models for Semiconductor Devices," Doc. No. JEP122G. 2011.

[17] J. Katcher, "Postmark: A New File System Benchmark," Network Appliance, Tech. Rep. TR3022, 1997.

[18] C. Kim *et al.*, "A 21 nm High Performance 64 Gb MLC NAND Flash Memory with 400 MB/s Asynchronous Toggle DDR Interface," *JSSC*, 2012.

[19] Y. Kim *et al.*, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA*, 2014.

[20] R. Koller and R. Rangaswami, "I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance," *TOS*, 2010.

[21] S. Lin and D. J. Costello, *Error Control Coding*. Prentice Hall, 2004.

[22] R.-S. Liu *et al.*, "Duracache: A Durable SSD Cache Using MLC NAND Flash," in *DAC*, 2013.

[23] R.-S. Liu *et al.*, "Optimizing NAND Flash-Based SSDs via Retention Relaxation," in *FAST*, 2012.

[24] N. Mielke *et al.*, "Bit Error Rate in NAND Flash Memories," in *IRPS*, 2008.

[25] V. Mohan *et al.*, "reFresh SSDs: Enabling High Endurance, Low Cost Flash in Datacenters," Univ. of Virginia, Tech. Rep. CS-2012-05, 2012.

[26] V. Mohan *et al.*, "How I Learned to Stop Worrying and Love Flash Endurance," in *HotStorage*, 2010.

[27] D. Narayanan *et al.*, "Write off-Loading: Practical Power Management for Enterprise Storage," *TOS*, 2008.

[28] Y. Pan *et al.*, "Quasi-Nonvolatile SSD: Trading Flash Memory Nonvolatility to Improve Storage System Performance for Enterprise Applications," in *HPCA*, 2012.

[29] K.-T. Park *et al.*, "A 7MB/s 64Gb 3-Bit/Cell DDR NAND Flash Memory in 20nm-Node Technology," in *ISSCC*, 2011.

[30] R. Smith, "SSD Moving Rapidly to the Next Level," *Flash Memory Summit*, 2014.

[31] Storage Network Industry Assn., "IOTTA Repository: Cello 1999." http://iotta.snia.org/traces/21

[32] T. Sugahara and T. Furuichi, "Memory Controller for Suppressing Read Disturb When Data Is Repeatedly Read Out," US Patent No. 8725952. 2014.

[33] K. Takeuchi *et al.*, "A Negative Vth Cell Architecture for Highly Scalable, Excellently Noise-Immune, and Highly Reliable NAND Flash Memories," *IEEE Journal of Solid-State Circuits*, 1999.

[34] Univ. of Massachusetts, "Storage: UMass Trace Repository." http://tinyurl.com/k6golon