Dynamic Program and Erase Scaling in NAND Flash-based Storage Systems

Jaeyong Jeong*, Sangwook Shane Hahn*, Sungjin Lee†, and Jihong Kim*

*Dept. of CSE , Seoul National University , {jyjeong, shanehahn, jihong}@davinci.snu.ac.kr
†CSAIL , Massachusetts Institute of Technology , chamdoo@csail.mit.edu

Abstract

The cost-per-bit of NAND flash memory has been continuously improved by semiconductor process scaling and multi-leveling technologies (e.g., a 10 nm-node TLC device). However, the decreasing lifetime of NAND flash memory as a side effect of recent advanced technologies is regarded as a main barrier for a wide adoption of NAND flash-based storage systems. In this paper, we propose a new system-level approach, called dynamic program and erase scaling (DPES), for improving the lifetime (particularly, endurance) of NAND flash memory. The DPES approach is based on our key observation that changing the erase voltage as well as the erase time significantly affects the NAND endurance. By slowly erasing a NAND block with a lower erase voltage, we can improve the NAND endurance very effectively. By modifying NAND chips to support multiple write and erase modes with different operation voltages and times, DPES enables a flash software to exploit the new tradeoff relationships between the NAND endurance and erase voltage/speed under dynamic program and erase scaling. We have implemented the first DPES-aware FTL, called autoFTL, which improves the NAND endurance with a negligible degradation in the overall write throughput. Our experimental results using various I/O traces show that autoFTL can improve the maximum number of P/E cycles by 61.2% over an existing DPES-unaware FTL with less than 2.2% decrease in the overall write throughput.

1 Introduction

NAND flash-based storage devices are increasingly popular from mobile embedded systems (e.g., smartphones and smartpads) to large-scale high-performance enterprise servers. Continuing semiconductor process scaling (e.g., 10 nm-node process technology) combined with various recent advances in flash technology (such as a TLC device [1] and a 3D NAND device [2]) is expected to further accelerate an improvement of the cost-per-bit of NAND devices, enabling a wider adoption of NAND flash-based storage systems. However, the poor endurance of NAND flash memory, which deteriorates further as a side effect of recent advanced technologies, is still regarded as a main barrier for sustainable growth in the NAND flash-based storage market. (We represent the NAND endurance by the maximum number of program/erase (P/E) cycles that a flash memory cell can tolerate while preserving data integrity.) Even though the NAND density doubles every two years, the storage lifetime does not increase as much as expected in a recent device technology [3]. For example, the NAND storage lifetime was increased by only 20% from 2009 to 2011 because the maximum number of P/E cycles was decreased by 40% during that period. In particular, in order for NAND flash memory to be widely adopted in high-performance enterprise storage systems, the deteriorating NAND endurance problem should be adequately resolved.

Since the lifetime L_C of a NAND flash-based storage device with the total capacity C is proportional to the maximum number $MAX_{P/E}$ of P/E cycles, and is inversely proportional to the total written data W_{day} per day, L_C (in days) can be expressed as follows (assuming a perfect wear leveling):

$$L_C = \frac{MAX_{P/E} \times C}{W_{day} \times WAF}, \tag{1}$$

where WAF is a write amplification factor which represents the efficiency of an FTL algorithm. Many existing lifetime-enhancing techniques have mainly focused on reducing WAF by increasing the efficiency of an FTL algorithm. For example, by avoiding unnecessary data copies during garbage collection, WAF can be reduced [4]. In order to reduce W_{day}, various architectural/system-level techniques were proposed. For example, data de-duplication [5], data compression [6] and write traffic throttling [7] are such examples. On the other hand, few system/software-level techniques were proposed for actively increasing the maximum number $MAX_{P/E}$ of P/E cycles. For example, a recent study [8] suggests $MAX_{P/E}$ can be indirectly improved by a self-recovery property of a NAND cell but no specific technique was proposed yet.

In this paper, we propose a new approach, called dynamic program and erase scaling (DPES), which can significantly improve $MAX_{P/E}$. The key intuition of our approach, which is motivated by a NAND device physics model on the endurance degradation, is that changing the erase voltage as well as the erase time significantly affects the NAND endurance. For example, slowly erasing a NAND block with a lower erase voltage can improve the NAND endurance significantly. By modifying a NAND device to support multiple write and erase modes (which have different voltage/speed and different impacts on the NAND endurance) and allowing a firmware/software module to choose the most appropriate write and erase mode (e.g., depending on a given workload), DPES can significantly increase $MAX_{P/E}$.

The physical mechanism of the endurance degradation is closely related to stress-induced damage in the tunnel oxide of a NAND memory cell [9]. Since the probability of stress-induced damage has an exponential dependence on the stress voltage [10], reducing the stress voltage (particularly, the erase voltage) is an effective way of improving the NAND endurance. Our measurement results with recent 20 nm-node NAND chips show that when the erase voltage is reduced by 14% during P/E cycles, $MAX_{P/E}$ can increase on average by 117%. However, in order to write data to a NAND block erased with the lower erase voltage (which we call a shallowly erased block in the paper), it is necessary to form narrow threshold voltage distributions after program operations. Since shortening the width of a threshold voltage distribution requires a fine-grained control during a program operation, the program time is increased if a lower erase voltage was used for erasing a NAND block.

Furthermore, for a given erase operation, since a nominal erase voltage (e.g., 14 V) tends to damage the cells more than necessary in the beginning period of an erase operation [11], starting with a lower (than the nominal) erase voltage and gradually increasing to the nominal erase voltage can improve the NAND endurance. However, gradually increasing the erase voltage increases the erase time. For example, our measurement results with recent 20 nm-node NAND chips show that when the initial erase voltage of 10 V is used instead of 14 V during P/E cycles, $MAX_{P/E}$ can increase on average by 17%. On the other hand, the erase time is increased by 300%.

Our DPES approach exploits the above two tradeoff relationships between the NAND endurance and erase voltage/speed at the firmware-level (or the software level in general) so that the NAND endurance is improved while the overall write throughput is not affected. For example, since the maximum performance of NAND flash memory is not always needed in real workloads, a DPES-based technique can exploit idle times between consecutive write requests for shortening the width of threshold voltage distributions so that shallowly erased NAND blocks, which were erased by lower erase voltages, can be used for most write requests. Idle times can be also used for slowing down the erase speed. If such idle times can be automatically estimated by a firmware/system software, the DPES-based technique can choose the most appropriate write speed for each write request or select the most suitable erase voltage/speed for each erase operation. By aggressively selecting endurance-enhancing erase modes (i.e., a slow erase with a lower erase voltage) when a large idle time is available, the NAND endurance can be significantly improved because less damaging erase operations are more frequently used.

In this paper, we present a novel NAND endurance model which accurately captures the tradeoff relationship between the NAND endurance and erase voltage/speed under dynamic program and erase scaling. Based on our NAND endurance model, we have implemented the first DPES-aware FTL, called *autoFTL*, which dynamically adjusts write and erase modes in an automatic fashion, thus improving the NAND endurance with a negligible degradation in the overall write throughput. In autoFTL, we also revised key FTL software modules (such as garbage collector and wear-leveler) to make them DPES-aware for maximizing the effect of DPES on the NAND endurance. Since no NAND chip currently allows an FTL firmware to change its program and erase voltages/times dynamically, we evaluated the effectiveness of autoFTL with the *FlashBench* emulation environment [12] using a DPES-enabled NAND simulation model (which supports multiple write and erase modes). Our experimental results using various I/O traces show that autoFTL can improve $MAX_{P/E}$ by 61.2% over an existing DPES-unaware FTL with less than 2.2% decrease in the overall write throughput.

The rest of the paper is organized as follows. Section 2 briefly explains the basics of NAND operations related to our proposed approach. In Section 3, we present the proposed DPES approach in detail. Section 4 describes our DPES-aware autoFTL. Experimental results follow in Section 5, and related work is summarized in Section 6. Finally, Section 7 concludes with a summary and future work.

2 Background

In order to improve the NAND endurance, our proposed DPES approach exploits key reliability and performance parameters of NAND flash memory during run time. In this section, we review the basics of various reliability parameters and their impact on performance and en-

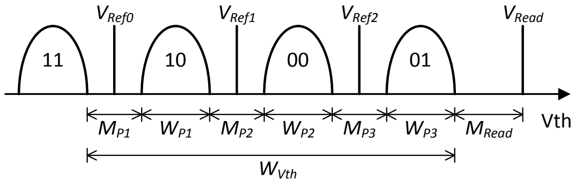


Figure 1: An example of threshold voltage distributions for multi-level NAND flash memory and primary reliability parameters.

durance of NAND cells.

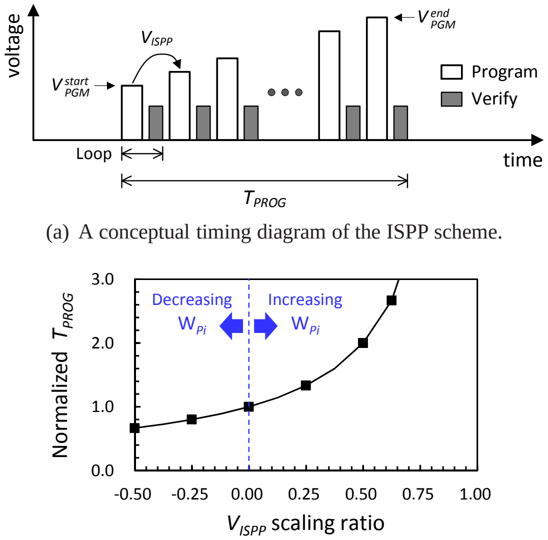
2.1 Threshold Voltage Distributions of NAND Flash Memory

Multi-level NAND flash memory stores 2 bits in a cell using four distinct threshold voltage levels (or states) as shown in Figure 1. Four states are distinguished by different reference voltages, V_{Ref0}, V_{Ref1} and V_{Ref2}. The threshold voltage gap M_{Pi} between two adjacent states and the width W_{Pi} of a threshold voltage distribution are mainly affected by data retention and program time requirements [13, 14], respectively. As a result, the total width W_{Vth} of threshold voltage distributions should be carefully designed to meet all the NAND requirements. In order for flash manufacturers to guarantee the reliability and performance requirements of NAND flash memory throughout its storage lifespan, all the reliability parameters, which are highly inter-related each other, are usually *fixed* during device design times under the worst-case operating conditions of a storage product.

However, if one performance/reliability requirement can be relaxed under specific conditions, it is possible to drastically improve the reliability or performance behavior of the storage product by exploiting tradeoff relationships among various reliability parameters. For example, Liu *et al.* [13] suggested a system-level approach that improves the NAND write performance when most of written data are short-lived (i.e., frequently updated data) by sacrificing M_{Pi}'s which affect the data retention capability[1]. Our proposed DPES technique exploits W_{Pi}'s (which also affect the NAND write performance) so that the NAND endurance can be improved.

2.2 NAND Program Operations

In order to form a threshold voltage distribution within a desired region, NAND flash memory generally uses the incremental step pulse programming (ISPP) scheme. As shown in Figure 2(a), the ISPP scheme gradually increases the program voltage by the V_{ISPP} step until all the memory cells in a page are located in a desired threshold voltage region. While repeating ISPP loops, once NAND cells are verified to have been sufficiently programmed, those cells are excluded from subsequent ISPP loops.



(a) A conceptual timing diagram of the ISPP scheme.

(b) Normalized T_{PROG} variations over different V_{ISPP} scaling ratios.

Figure 2: An overview of the incremental step pulse programming (ISPP) scheme for NAND flash memory.

Since the program time is proportional to the number of ISPP loops (which are inversely proportional to V_{ISPP}), the program time T_{PROG} can be expressed as follows:

$$T_{PROG} \propto \frac{V_{PGM}^{end} - V_{PGM}^{start}}{V_{ISPP}}. \qquad (2)$$

Figure 2(b) shows normalized T_{PROG} variations over different V_{ISPP} scaling ratios. (When a V_{ISPP} scaling ratio is set to x%, V_{ISPP} is reduced by x% of the nominal V_{ISPP}.) When a narrow threshold voltage distribution is needed, V_{ISPP} should be reduced for a fine-grained control, thus increasing the program time. Since the width of a threshold voltage distribution is proportional to V_{ISPP} [14], for example, if the nominal V_{ISPP} is 0.5 V and the width of a threshold voltage distribution is reduced by 0.25 V, V_{ISPP} also needs to be reduced by 0.25 V (i.e., a V_{ISPP} scaling ratio is 0.5), thus increasing T_{PROG} by 100%.

3 Dynamic Program and Erase Scaling

The DPES approach is based on our key observation that slowly erasing (i.e., erase time scaling) a NAND block with a lower erase voltage (i.e., erase voltage scaling) significantly improves the NAND endurance. In this section, we explain the effect of erase voltage scaling on improving the NAND endurance and describe the dynamic program scaling method for writing data to a shallowly erased NAND block (i.e., a NAND block erased with

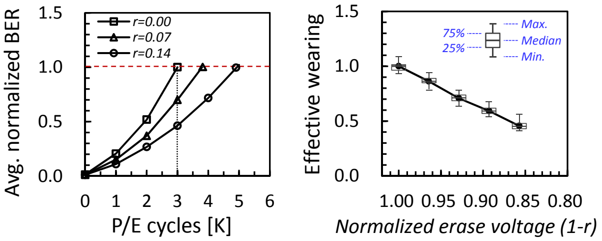
[1] Since short-lived data do not need a long data retention time, M_{Pi}'s are maintained loosely so that the NAND write performance can be improved.

a lower erase voltage). We also present the concept of erase time scaling and its effect on improving the NAND endurance. Finally, we present a novel NAND endurance model which describes the effect of DPES on the NAND endurance based on an empirical measurement study using 20 nm-node NAND chips.

3.1 Erase Voltage Scaling and its Effect on NAND Endurance

The time-to-breakdown T_{BD} of the oxide layer decreases exponentially as the stress voltage increases because the higher stress voltage accelerates the probability of stress-induced damage which degrades the oxide reliability [10]. This phenomenon implies that the NAND endurance can be improved by lowering the stress voltage (e.g., program and erase voltages) during P/E cycles because the reliability of NAND flash memory primarily depends on the oxide reliability [9]. Although the maximum program voltage to complete a program operation is usually larger than the erase voltage, the NAND endurance is mainly degraded during erase operations because the stress time interval of an erase operation is about 100 times longer than that of a program operation. Therefore, if the erase voltage can be lowered, its impact on the NAND endurance improvement can be significant.

In order to verify our observation, we performed NAND cycling tests by changing the erase voltage. In a NAND cycling test, program and erase operations are repeated 3,000 times (which are roughly equivalent to $MAX_{P/E}$ of a recent 20 nm-node NAND device [3]). Our cycling tests for each case are performed with more than 80 blocks which are randomly selected from 5 NAND chips. In our tests, we used the NAND retention BER (i.e., a BER after 10 hours' baking at 125 °C) as a measure for quantifying the wearing degree of a NAND chip [9]. (This is a standard NAND retention evaluation procedure specified by JEDEC [15].) Figure 3(a) shows how the retention BER changes, on average, as the number of P/E cycles increases while varying erase voltages. We represent different erase voltages using an voltage scaling ratio r ($0 \leq r \leq 1$). When r is set to x, the erase voltage is reduced by ($x \times 100$)% of the nominal erase voltage. The retention BERs were normalized over the retention BER after 3K P/E cycles when the nominal erase voltage was used. As shown in Figure 3(a), the more the erase voltage is reduced (i.e., the higher r's), the less the retention BERs[2]. For example, when the erase voltage is reduced by 14% of the nominal erase voltage, the normalized retention BER is reduced by 54% after 3K P/E cycles over the nominal erase voltage case.



(a) Average BER variations over different P/E cycles under varying erase voltage scaling ratios (r's)

(b) Effective wearing over different erase voltage scaling ratios (r's)

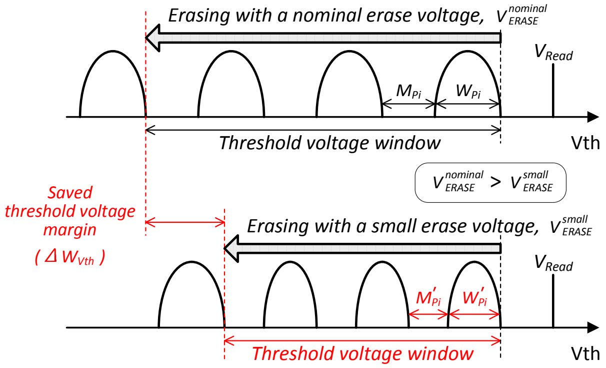
Figure 3: The effect of lowering the erase voltage on the NAND endurance.

Since the normalized retention BER reflects the degree of the NAND wearing, higher r's lead to less endurance degradations. Since different erase voltages degrade the NAND endurance by different amounts, we introduce a new endurance metric, called *effective wearing per PE* (in short, *effective wearing*), which represents the effective degree of NAND wearing after a P/E cycle. We represent the effective wearing by a normalized retention BER after 3K P/E cycles[3]. Since the normalized retention BER is reduced by 54% when the erase voltage is reduced by 14%, the effective wearing becomes 0.46. When the nominal erase voltage is used, the effective wearing is 1.

As shown in Figure 3(b), the effective wearing decreases near-linearly as r increases. Based on a linear regression model, we can construct a linear equation for the effective wearing over different r's. Using this equation, we can estimate the effective wearing for a different r. After 3K P/E cycles, for example, the total sum of the

[2] Since we do not have other NAND chips from different NAND manufacturer, we could not prove that our test results can be generalized. However, since our test results are based on the widely-known device physics which have been investigated by many device engineers/researchers, we are convinced that the consistency of our results would be maintained as long as NAND flash memories use the same physical mechanism (i.e., the FN-tunneling) for program and erase operations.

[3] In this paper, we use a linear approximation model which simplifies the wear-out behavior over P/E cycles for supporting a simple software implementation while achieving a reasonable accuracy. Our current linear model can overestimate the effective wearing under low erase voltage scaling ratios while it can underestimate the effective wearing under high erase voltage scaling ratios. The upper bound of the estimation error caused by our model can be calculated from the relative wearing for each extreme case as follows: (1) If we always erase a NAND block with the normalized erase voltage of 1.00, the NAND endurance is underestimated by up to 20% (2) If we always use the normalized erase voltage of 0.86 (i.e., r is 0.14), the NAND endurance is overestimated by up to 23%. However, overestimating the effective wearing is not a problem because the endurance gain will be underestimated (scheme will be more effective in practice than outlined here). So, we can focus on how much wearing our model underestimates. The lower the erase voltage used, the more endurance gain can be achieved while the probability of estimation error is also increased. Since we set the reliability margin so conservative (as will be described in the next section), the minimum available erase voltage is only 0.89 (i.e., r is 0.11) and the lower bound of the relative total wearing is 0.91. As a result, the maximum overestimation in the endurance gain is only 9.9% (=1/0.91). This allows us to conclude that our device model does not overestimate the endurance gain by more than 10%.



Erasing with a nominal erase voltage, $V_{ERASE}^{nominal}$
V_{Read}
M_{Pi}
W_{Pi}
Threshold voltage window
Vth
$V_{ERASE}^{nominal}$ > V_{ERASE}^{small}
Saved threshold voltage margin
(ΔW_{Vth})
Erasing with a small erase voltage, V_{ERASE}^{small}
V_{Read}
M'_{Pi}
W'_{Pi}
Threshold voltage window
Vth

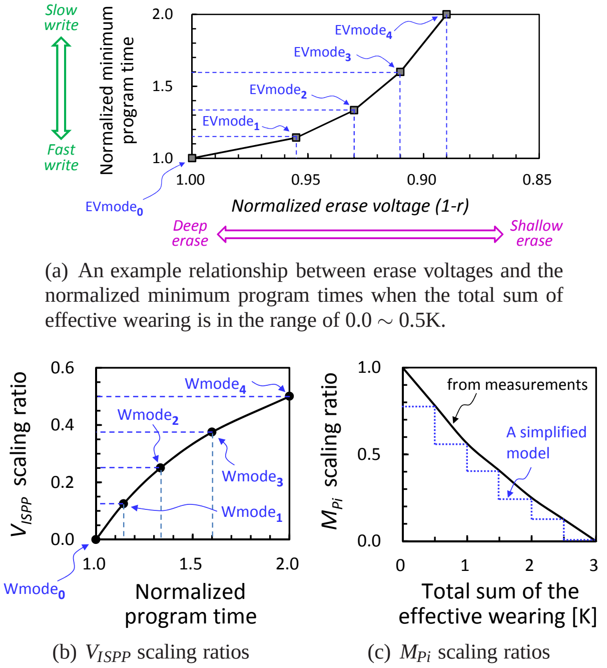
Figure 4: An example of program voltage scaling for writing data to a shallowly erased NAND block.

effective wearing with the nominal erase voltage is 3K. On the other hand, if the erase voltage was set to 14% less than the nominal voltage, the total sum of the effective wearing is only 1.38K because the effective wearing with r of 0.14 is 0.46. As a result, $MAX_{P/E}$ can be increased more than twice as much when the erase voltage is reduced by 14% over the nominal case. In this paper, we will use a NAND endurance model with five different erase voltage modes (as described in Section 3.5).

3.2 Dynamic Program Scaling

In order to write data to a shallowly erased NAND block, it is necessary to change program bias conditions dynamically so that narrow threshold voltage distributions can be formed after program operations. If a NAND block was erased with a lower erase voltage, a threshold voltage window for a program operation is reduced by the decrease in the erase voltage because the value of the erase voltage decides how deeply a NAND block is erased. For example, as shown in Figure 4, if a NAND block is shallowly erased with a lower erase voltage V_{ERASE}^{small} (which is lower than the nominal erase voltage $V_{ERASE}^{nominal}$), the width of a threshold voltage window is reduced by a saved threshold voltage margin ΔW_{Vth} (which is proportional to the voltage difference between $V_{ERASE}^{nominal}$ and V_{ERASE}^{small}). Since threshold voltage distributions can be formed only within the given threshold voltage window when a lower erase voltage is used, a fine-grained program control is necessary, thus increasing the program time of a shallowly erased block.

In our proposed DPES technique, we use five different erase voltage modes, EVmode$_0$, $\cdots$, EVmode$_4$. EVmode$_0$ uses the highest erase voltage V_0 while EVmode$_4$ uses the lowest erase voltage V_4. After a NAND block is erased, when the erased block is programmed again, there is a strict requirement on the minimum interval length of the program time which depends on the erase voltage mode used for the erased block. (As explained above, this minimum program time requirement is necessary to form threshold voltage distributions within the reduced threshold voltage window.) Figure 5(a) shows these minimum program times for five erase voltage modes. For example, if a NAND block were erased by EVmode$_4$, where the erase voltage is 89% of the nominal erase voltage, the erased block would need at least twice longer program time than the nominal program time. On the other hand, if a NAND block were erased by EVmode$_0$, where the erase voltage is same as the nominal erase voltage, the erased block can be programmed with the same nominal program time.



(a) An example relationship between erase voltages and the normalized minimum program times when the total sum of effective wearing is in the range of 0.0 ~ 0.5K.

(b) V_{ISPP} scaling ratios

(c) M_{Pi} scaling ratios

Figure 5: The relationship between the erase voltage and the minimum program time, and V_{ISPP} scaling and M_{Pi} scaling for dynamic program scaling.

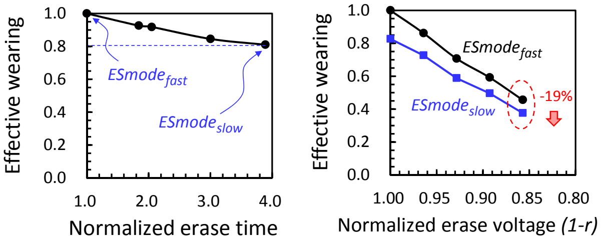
In order to satisfy the minimum program time requirements of different EVmode$_i$'s, we define five different write modes, Wmode$_0$, $\cdots$, Wmode$_4$ where Wmode$_i$ satisfies the minimum program time requirement of the blocks erased by EVmode$_i$. Since the program time of Wmode$_j$ is longer than that of Wmode$_i$ (where $j > i$), Wmode$_k$, Wmode$_{(k+1)}$, $\cdots$, Wmode$_4$ can be used when writing to the blocks erased by EVmode$_k$. Figure 5(b) shows how V_{ISPP} should be scaled for each write mode so that the minimum program time requirement can be satisfied. The program time is normalized over the nominal T_{PROG}.

In order to form threshold voltage distributions within a given threshold voltage window, a fine-grained program control is necessary by reducing M_{Pi}'s and W_{Pi}'s. As described in Section 2.2, we can reduce W_{Pi}'s by scaling V_{ISPP} based on the program time requirement. Figure 5(b) shows the tradeoff relationship between the program time and V_{ISPP} scaling ratio based on our NAND characterization study. The program time is normalized over the nominal T_{PROG}. For example, in the case of `Wmode`$_4$, when the program time is two times longer than the nominal T_{PROG}, V_{ISPP} can be maximally reduced. Dynamic program scaling can be easily integrated into an existing NAND controller with a negligible time overhead (e.g., less than 1% of T_{PROG}) and a very small space overhead (e.g., 4 bits per block). On the other hand, in conventional NAND chips, M_{Pi} is kept large enough to preserve the data retention requirement under the worst-case operating condition (e.g., 1-year data retention after 3,000 P/E cycles). However, since the data retention requirement is proportional to the total sum of the effective wearing [9], M_{Pi} can be relaxed by removing an unnecessary data retention capability. Figure 5(c) shows our M_{Pi} scaling model over different total sums of the effective wearing based on our measurement results. In order to reduce the management overhead, we change the M_{Pi} scaling ratio every 0.5-K P/E cycle interval (as shown by the dotted line in Figure 5(c)).

3.3 Erase Time Scaling and its Effect on NAND Endurance

When a NAND block is erased, a high nominal erase voltage (e.g., 14 V) is applied to NAND memory cells. In the beginning period of an erase operation, since NAND memory cells are not yet sufficiently erased, an excessive high voltage (i.e., the nominal erase voltage plus the threshold voltage in a programmed cell) is inevitably applied across the tunnel oxide. For example, if 14 V is required to erase NAND memory cells, when an erase voltage (i.e., 14 V) is applied to two programmed cells whose threshold voltages are 0 V and 4 V, the total erase voltages applied to two memory cells are 14 V and 18 V, respectively. As described in Section 3.1, since the probability of damage is proportional to the erase voltage, the memory cell with a high threshold voltage is damaged more than that with a low threshold voltage, resulting in unnecessarily degrading the memory cell with a high threshold voltage.

In order to minimize unnecessary damage in the beginning period of an erase operation, it is an effective way to start the erase voltage with a sufficiently low voltage (e.g., 10 V) and gradually increase to the nominal erase voltage [11]. For example, if we start with the erase voltage of 10 V, the memory cell whose threshold voltage is 4 V may be partially erased because the erase voltage is 14 V (i.e., 10 V plus 4 V) without excessive damage to the memory cell. As we increase the erase voltage in subsequent ISPE (incremental step pulse erasing [16]) loops, the threshold voltage in the cell is reduced by each ISPE step, thus avoiding unnecessary damage during an erase operation. In general, the lower the starting erase voltage, the less damage to the cells.



(a) Effective wearing variations over different erase times

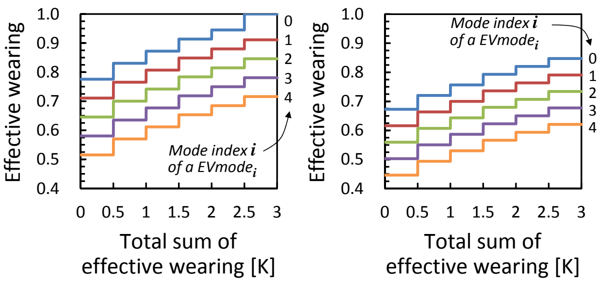
(b) Effective wearing variations over varying erase voltage scaling ratios (r's) under two different erase time settings

Figure 6: The effect of erase time scaling on the NAND endurance.

However, as an erase operation starts with a lower voltage than the nominal voltage, the erase time increases because more erase loops are necessary for completing the erase operation. Figure 6(a) shows how the effective wearing decreases, on average, as the erase time increases. The longer the erase time (i.e., the lower the starting erase voltage), the less the effective wearing (i.e., the higher NAND endurance.). We represent the fast erase mode by `ESmode`$_{fast}$ and the slow erase mode by `ESmode`$_{slow}$. Our measurement results with 20 nm-node NAND chips show that if we increase the erase time by 300% by starting with a lower erase voltage, the effective wearing is reduced, on average, by 19%. As shown in Figure 6(b), the effect of the slow erase mode on improving the NAND endurance can be exploited regardless of the erase voltage scaling ratio r. Since the erase voltage modes are continuously changed depending on the program time requirements, the endurance-enhancing erase mode (i.e., the lowest erase voltage mode) cannot be used under an intensive workload condition. On the other hand, the erase time scaling can be effective even under an intensive workload condition, if slightly longer erase times do not affect the overall write throughput.

3.4 Lazy Erase Scheme

As explained in Section 3.2, when a NAND block was erased with `EVmode`$_i$, a page in the shallowly erased block can be programmed using specific `Wmode`$_j$'s (where $j \geq i$) only because the requirement of the saved threshold voltage margin cannot be satisfied with a faster write mode `Wmode`$_k$ $(k < i)$. In order to write data with a faster write mode to the shallowly erased NAND block, the shallowly erased block should be erased further before it is written. We propose a lazy erase scheme which additionally erases the shallowly erased NAND block, when necessary, with a small extra erase time (i.e., 20% of the nominal erase time). Since the effective wearing mainly depends on the maximum erase voltage used, erasing a NAND block by a high erase voltage in a lazy



(a) The endurance model for ESmode$_{fast}$.

(b) The endurance model for ESmode$_{slow}$.

Figure 7: The proposed NAND endurance model for DPES-enabled NAND blocks.

fashion does not incur any extra damage than erasing it with the initially high erase voltage[4]. Since a lazy erase cancels an endurance benefit of a shallow erase while introducing a performance penalty, it is important to accurately estimate the write speed of future write requests so that correct erase modes can be selected when erasing NAND blocks, thus avoiding unnecessary lazy erases.

3.5 NAND Endurance Model

Combining erase voltage scaling, program time scaling and erase time scaling, we developed a novel NAND endurance model that can be used with DPES-enabled NAND chips. In order to construct a DPES-enabled NAND endurance model, we calculate saved threshold voltage margins for each combination of write modes (as shown in Figure 5(b)) and M_{Pi} scaling ratios (as shown in Figure 5(c)). Since the effective wearing has a near-linear dependence on the erase voltage and time as shown in Figures 3(b) and 6(b), respectively, the values of the effective wearing for each saved threshold voltage margin can be estimated by a linear equation as described in Section 3.1. All the data in our endurance model are based on measurement results with recent 20 nm-node NAND chips. For example, when the number of P/E cycles is less than 500, and a block is slowly erased before writing with the slowest write mode, a saved threshold voltage margin can be estimated to 1.06 V (which corresponds to the erase voltage scaling ratio r of 0.14 in Figure 6(b)). As a result, we can estimate the value of the effective wearing as 0.45 by a linear regression model for the solid line with squared symbols in Figure 6(b).

Figure 7 shows our proposed NAND endurance model with five erase voltage modes (i.e., EVmode$_0$ ∼ EVmode$_4$) and two erase speed modes (i.e., ESmode$_{slow}$ and ESmode$_{fast}$). EVmode$_0$ (which uses the largest erase voltage) supports the fastest write mode (i.e., Wmode$_0$) with no slowdown in the write speed while EVmode$_4$ (which uses the smallest erase voltage) supports only the slowest write mode (i.e., Wmode$_4$) with the largest wearing gain. Similarly, ESmode$_{fast}$ is the fast erase mode with no additional wearing gain while ESmode$_{slow}$ represents the slow erase mode with the improved wearing gain. Our proposed NAND endurance model takes account of both V_{ISPP} scaling and M_{Pi} scaling described in Figures 5(b) and 5(c).

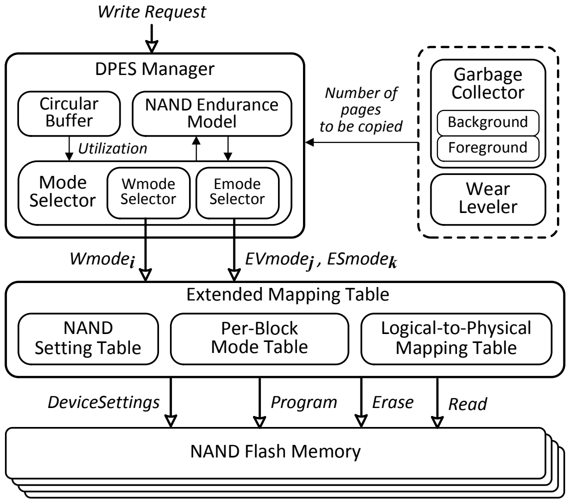


Figure 8: An organizational overview of autoFTL.

4 Design and Implementation of AutoFTL

4.1 Overview

Based on our NAND endurance model presented in Section 3.5, we have implemented autoFTL, the first DPES-aware FTL, which automatically changes write and erase modes depending on write throughput requirements. AutoFTL is based on a page-level mapping FTL with additional modules for DPES support. Figure 8 shows an organizational overview of autoFTL. The DPES manager, which is the core module of autoFTL, selects a write mode Wmode$_i$ for a write request and decides both an appropriate erase voltage mode EVmode$_j$ and erase speed mode ESmode$_k$ for each erase operation. In determining appropriate modes, the mode selector bases its decisions on the estimated write throughput requirement using a circular buffer. AutoFTL maintains per-block mode information and NAND setting information as well as logical-to-physical mapping information in the extended mapping table. The per-block mode table keeps track of the current write mode and the total sum of the effective wearing for each block. The NAND setting table is used to choose appropriate device settings

[4] Although it takes a longer erase time, the total sum of the effective wearing by lazily erasing a shallowly erased block is less than that by erasing with the initially high erase voltage. This can be explained in a similar fashion as why the erase time scaling is effective in improving the NAND endurance as discussed in the previous section. The endurance gain from using two different starting erase voltages is higher than the endurance loss from a longer erase time.

Table 1: The write-mode selection rules used by the DPES manager.

Buffer utilization u	Write mode
$u > 80\%$	$\texttt{Wmode}_0$
$60\% < u \leq 80\%$	$\texttt{Wmode}_1$
$40\% < u \leq 60\%$	$\texttt{Wmode}_2$
$20\% < u \leq 40\%$	$\texttt{Wmode}_3$
$u \leq 20\%$	$\texttt{Wmode}_4$

for the selected write and erase modes, which are sent to NAND chips via a new interface *DeviceSettings* between autoFTL and NAND chips[5]. AutoFTL also extends both the garbage collector and wear leveler to be DPES-aware.

4.2 Write Mode Selection

In selecting a write mode for a write request, the Wmode selector of the DPES manager exploits idle times between consecutive write requests so that autoFTL can increase $MAX_{P/E}$ without incurring additional decrease in the overall write throughput. In autoFTL, the Wmode selector uses a simple circular buffer for estimating the maximum available program time (i.e., the minimum required write speed) for a given write request. Table 1 summarizes the write-mode selection rules used by the Wmode selector depending on the utilization of a circular buffer. The circular buffer queues incoming write requests before they are written, and the Wmode selector adaptively decides a write mode for each write request. The current version of the Wmode selector, which is rather conservative, chooses the write mode, $\texttt{Wmode}_i$, depending on the buffer utilization u. The buffer utilization u represents how much of the circular buffer is filled by outstanding write requests. For example, if the utilization is lower than 20%, the write request in the head of the circular buffer is programmed to a NAND chip with $\texttt{Wmode}_4$.

4.3 Extended Mapping Table

Since erase operations are performed at the NAND block level, the per-block mode table maintains five linked lists of blocks which were erased using the same erase voltage mode. When the DPES manager decides a write mode for a write request, the corresponding linked list is consulted to locate a destination block for the write request. Also, the DPES manager informs a NAND chip how to configure appropriate device settings (e.g., ISPP/ISPE voltages, the erase voltage, and reference voltages for read/verify operations) for the current write mode using the per-block mode table. Once NAND chips are set to a certain mode, an additional setting is not necessary as long as the write and the erase modes are maintained. For a read request, since different write modes require different reference voltages for read operations, the per-block mode table keeps track of the current write mode for each block so that a NAND chip changes its read references before serving a read request.

4.4 Erase Voltage Mode Selection

Since the erase voltage has a significant impact on the NAND endurance as described in Section 3.1, selecting a right erase voltage is the most important step in improving the NAND endurance using the DPES technique. As explained in Section 4.2, since autoFTL decides a write mode of a given write request based on the utilization of the circular buffer of incoming write requests, when deciding the erase voltage mode of a victim block, autoFTL takes into account of the future utilization of the circular buffer. If autoFTL could accurately predict the future utilization of the circular buffer and erase the victim block with the erase voltage that can support the future write mode, the NAND endurance can be improved without a lazy erase operation. In the current version, we use the average buffer utilization of 10^5 past write requests for predicting the future utilization of the circular buffer. In order to reduce the management overhead, we divide 10^5 past write requests into 100 subgroups where each subgroup consists of 1000 write requests. For each subgroup, we compute the average utilization of 1000 write requests in the subgroup, and use the average of 100 subgroup's utilizations to calculate the estimate of the future utilization of the buffer.

When a foreground garbage collection is invoked, since the write speed of a near-future write request is already chosen based on the current buffer utilization, the victim block can be erased with the corresponding erase voltage mode. On the other hand, when a background garbage collection is invoked, it is difficult to use the current buffer utilization because the background garbage collector is activated when there are no more write requests waiting in the buffer. For this case, we use the estimated average buffer utilization of the circular buffer to predict the buffer utilization when the next phase of write requests (after the background garbage collection) fills in the circular buffer.

[5] As semiconductor technologies reach their physical limitations, it is necessary to use cross-layer optimization between system software and NAND devices. As a result, some of internal device interfaces are gradually opened to public in the form of additional user interface. For example, in order to track bit errors caused by data retention, a new device setting interface which adjusts the internal reference voltages for read operations is recently opened to public [17, 18]. There are already many set and get functions for modifying or monitoring NAND internal configurations in the up-to-date NAND specifications such as the toggle mode interface and ONFI. For the measurements presented here, we were fortunately able to work in conjunction with a flash manufacturer to adjust erase voltage as we wanted.

4.5 Erase Speed Mode Selection

In selecting an erase speed mode for a block erase operation, the DPES manager selects an erase speed mode which does not affect the write throughput. An erase speed mode for erasing a NAND block is determined by estimating the effect of a block erase time on the buffer utilization. Since write requests in the circular buffer cannot be programmed while erasing a NAND block, the buffer utilization is effectively increased by the block erase time. The effective buffer utilization u' considering the effect of the block erase time can be expressed as follows:

$$u' = u + \Delta u^{erase}, \tag{3}$$

where u is the current buffer utilization and Δu^{erase} is the increment in the buffer utilization by the block erase time. In order to estimate the effect of a block erase operation on the buffer utilization, we convert the block erase time to a multiple M of the program time of the current write mode. Δu^{erase} corresponds to the increment in the buffer utilization for these M pages. For selecting an erase speed mode of a NAND block, the mode selector checks if $\texttt{ESmode}_{slow}$ can be used. If erasing with $\texttt{ESmode}_{slow}$ does not increase u' larger than 100% (i.e., no buffer overflow), $\texttt{ESmode}_{slow}$ is selected. Otherwise, the fast erase mode $\texttt{ESmode}_{fast}$ is selected. On the other hand, when the background garbage collection is invoked, $\texttt{ESmode}_{slow}$ is always selected in erasing a victim block. Since the background garbage collection is invoked when an idle time between consecutive write requests is sufficiently long, the overall write throughput is not affected even with $\texttt{ESmode}_{slow}$.

4.6 DPES-Aware Garbage Collection

When the garbage collector is invoked, the most appropriate write mode for copying valid data to a free block is determined by using the same write-mode selection rules summarized in Table 1 with a slight modification to computing the buffer utilization u. Since the write requests in the circular buffer cannot be programmed while copying valid pages to a free block by the garbage collector, the buffer utilization is effectively increased by the number of valid pages in a victim block. By using the information from the garbage collector, the mode selector recalculates the effective buffer utilization u^* as follows:

$$u^* = u + \Delta u^{copy}, \tag{4}$$

where u is the current buffer utilization and Δu^{copy} is the increment in the buffer utilization taking the number of valid pages to be copied into account. The mode selector decides the most appropriate write mode based on the write-mode selection rules with u^* instead of u. After copying all the valid pages to a free block, a victim block is erased by the erase voltage mode (selected by the rules described in Section 4.4) with the erase speed (chosen by the rules described in Section 4.5). For example, as shown in the case 1 of Table 2, if garbage collection is invoked when u is 70%, and the number of valid pages to be copied is 30 (i.e., $\Delta u^{copy} = 30/200 = 15\%$), $\texttt{Wmode}_0$ is selected because u^* is 85% ($= 70\% + 15\%$), and $\texttt{ESmode}_{slow}$ is selected because erasing with $\texttt{ESmode}_{slow}$ does not overflow the circular buffer. (We assume that Δu^{erase} for $\texttt{ESmode}_{slow}$ and Δu^{erase} for $\texttt{ESmode}_{fast}$ are 8% and 2%, respectively.) On the other hand, as shown in the case 2 of Table 2, when the number of valid pages to be copied is 50 (i.e., $\Delta u^{copy} = 50/200 = 25\%$), $\texttt{ESmode}_{slow}$ cannot be selected because u' becomes larger than 100%. As shown in the case 1, $\texttt{ESmode}_{slow}$ can still be used even when the buffer utilization is higher than 80%. When the buffer utilization is higher than 80% (i.e., an intensive write workload condition), the erase voltage scaling is not effective because the highest erase voltage is selected. On the other hand, even when the buffer utilization is above 90%, the erase speed scaling can be still useful.

Table 2: Examples of selecting write and erase modes in the garbage collector assuming that the circular buffer has 200 pages and the current buffer utilization u is 70%.

(Case 1) The number of valid pages in a victim block is 30.

u^{copy}	u^*	Δu^{erase}		u'	Selected modes
15%	85%	Slow	8%	93%	$\texttt{EVmode}_0$ & $\texttt{ESmode}_{slow}$ $\texttt{Wmode}_0$
		Fast	2%	87%	

(Case 2) The number of valid pages in a victim block is 50.

u^{copy}	u^*	Δu^{erase}		u'	Selected modes
25%	95%	Slow	8%	103%	$\texttt{EVmode}_0$ & $\texttt{ESmode}_{fast}$ $\texttt{Wmode}_0$
		Fast	2%	97%	

4.7 DPES-Aware Wear Leveling

Since different erase voltage/time affects the NAND endurance differently as described in Section 3.1, the reliability metric (based on the number of P/E cycles) of the existing wear leveling algorithm [19] is no longer valid in a DPES-enabled NAND flash chip. In autoFTL, the DPES-aware wear leveler uses the total sum of the effective wearing instead of the number of P/E cycles as a reliability metric, and tries to evenly distribute the total sum of the effective wearing among NAND blocks.

5 Experimental Results

5.1 Experimental Settings

In order to evaluate the effectiveness of the proposed autoFTL, we used an extended version of a unified development environment, called *FlashBench* [12], for NAND flash-based storage devices. Since the efficiency of our DPES is tightly related to the temporal characteristics of write requests, we extended the existing FlashBench to be timing-accurate. Our extended FlashBench emulates the key operations of NAND flash memory in a timing-accurate fashion using high-resolution timers (or hrtimers) (which are available in a recent Linux kernel [20]). Our validation results on an 8-core Linux server system show that the extended FlashBench is very accurate. For example, variations on the program time and erase time of our DRAM-based NAND emulation models are less than 0.8% of T_{PROG} and 0.3% of T_{ERASE}, respectively.

For our evaluation, we modified a NAND flash model in FlashBench to support DPES-enabled NAND flash chips with five write modes, five erase voltage modes, and two erase speed modes as shown in Figure 7. Each NAND flash chip employed 128 blocks which were composed of 128 8-KB pages. The maximum number of P/E cycles was set to 3,000. The nominal page program time (i.e., T_{PROG}) and the nominal block erase time (i.e., T_{ERASE}) were set to 1.3 *m*s and 5.0 *m*s, respectively.

We evaluated the proposed autoFTL in two different environments, mobile and enterprise environments. Since the organizations of mobile storage systems and enterprise storage systems are quite different, we used two FlashBench configurations for different environments as summarized in Table 3. For a mobile environment, FlashBench was configured to have two channels, and each channel has a single NAND chip. Since mobile systems are generally resource-limited, the size of a circular buffer for a mobile environment was set to 80 KB only (i.e., equivalently 10 8-KB pages). For an enterprise environment, FlashBench was configured to have eight channels, each of which was composed of four NAND chips. Since enterprise systems can utilize more resources, the size of a circular buffer was set to 32 MB (which is a typical size of data buffer in HDD) for enterprise environments.

We carried out our evaluations with two different techniques: baseline and autoFTL. Baseline is an existing DPES-unaware FTL that always uses the highest erase voltage mode and the fast erase mode for erasing NAND blocks, and the fastest write mode for writing data to NAND blocks. AutoFTL is the proposed DPES-aware FTL which decides the erase voltage and the erase time depending on the characteristic of a workload and fully utilizes DPES-aware techniques, described in Sections 3 and 4, so it can maximally exploit the benefits of dynamic program and erase scaling.

Table 3: Summary of two FlashBench configurations.

Environments	Channels	Chips	Buffer
Mobile	2	2	80 KB
Enterprise	8	32	32 MB

Our evaluations were conducted with various I/O traces from mobile and enterprise environments. (For more details, please see Section 5.2). In order to replay I/O traces on top of the extended FlashBench, we developed a trace replayer. The trace replayer fetches I/O commands from I/O traces and then issues them to the extended FlashBench according to their inter-arrival times to a storage device. After running traces, we measured the maximum number of P/E cycles, $MAX_{P/E}$, which was actually conducted until flash memory became unreliable. We then compared it with that of baseline. The overall write throughput is an important metric that shows the side-effect of autoFTL on storage performance. For this reason, we also measured the overall write throughput while running each I/O trace.

5.2 Benchmarks

We used 8 different I/O traces collected from Android-based smartphones and real-world enterprise servers. The `m_down` trace was recorded while downloading a system installation file (whose size is about 700 MB) using a mobile web-browser through 3G network. The `m_p2p1` trace included I/O activities when downloading multimedia files using a mobile P2P application from a lot of rich seeders. Six enterprise traces, `hm_0`, `proj_0`, `prxy_0`, `src1_2`, `stg_0`, and `web_0`, were from the MS-Cambridge benchmarks [21]. However, since enterprise traces were collected from old HDD-based server systems, their write throughputs were too low to evaluate the performance of modern NAND flash-based storage systems. In order to partially compensate for low write throughput of old HDD-based storage traces, we accelerated all the enterprise traces by 100 times so that the peak throughput of the most intensive trace (i.e., `src1_2`) can fully consume the maximum write throughput of our NAND configuration[6]. (In our evaluations, therefore, all

[6] Since recent enterprise SSDs utilize lots of inter-chip parallelism (multiple channels) and intra-chip parallelism (multiple planes), peak throughput is significantly higher than that of conventional HDDs. We tried to find appropriate enterprise traces which satisfied our requirements to (1) have public confidence; (2) can fully consume the maximum throughput of our NAND configuration; (3) reflect real user behaviors in enterprise environments; (4) are extracted from under SSD-based storage systems. To the best of our knowledge, we could not find any workload which met all of the requirements at the same time. In particular, there are few enterprise SSD workloads which are opened to

Table 4: Normalized inter-arrival times of write requests for 8 traces used for evaluations.

Trace	Distributions of normalized inter-arrival times t over $T_{PROG}^{effective}$ [%]		
	$t \leq 1$	$1 < t \leq 2$	$t > 2$
proj_0	40.6%	47.0%	12.4%
src1_2	41.0%	55.6%	3.4%
hm_0	14.2%	72.1%	13.7%
prxy_0	8.9%	34.6%	56.5%
stg_0	7.1%	81.5%	11.4%
web_0	5.4%	36.7%	56.9%
m_down	45.9%	0.0%	54.1%
m_p2p1	49.5%	0.0%	50.5%

the enterprise traces are 100x-accelerated versions of the original traces.)

Table 4 summarizes the distributions of inter-arrival times of our I/O traces. Inter-arrival times were normalized over $T_{PROG}^{effective}$ which reflects parallel NAND operations supported by multiple channels and multiple chips per channel in the extended FlashBench. For example, for an enterprise environment, since up to 32 chips can serve write requests simultaneously, $T_{PROG}^{effective}$ is about 40 us (i.e., 1300 us of T_{PROG} is divided by 32 chips.). On the other hand, for a mobile environment, since there are only 2 chips can serve write requests at the same time, $T_{PROG}^{effective}$ is 650 us. Although the mobile traces collected from Android smartphones (i.e., m_down [22] and m_p2p1) exhibit very long inter-arrival times, normalized inter-arrival times over $T_{PROG}^{effective}$ are not much different from the enterprise traces, except that the mobile traces show distinct bimodal distributions which no write requests in $1 < t \leq 2$.

5.3 Endurance Gain Analysis

In order to understand how much $MAX_{P/E}$ is improved by DPES, each trace was repeated until the total sum of the effective wearing reached 3K. Measured $MAX_{P/E}$ values were normalized over that of baseline. Figure 9 shows normalized $MAX_{P/E}$ ratios for eight traces with two different techniques. Overall, the improvement on $MAX_{P/E}$ is proportional to inter-arrival times as summarized in Table 4; the longer inter-arrival times are, the more likely slow write modes are selected.

AutoFTL improves $MAX_{P/E}$ by 69%, on average, over baseline for the enterprise traces. For proj_0 and src1_2 traces, improvements on $MAX_{P/E}$ are less than 50% because inter-arrival times of more than 40% of write requests are shorter than $T_{PROG}^{effective}$ so that it is difficult to use the lowest erase voltage mode. For the other enterprise traces, $MAX_{P/E}$ is improved by 79%, on average, over baseline.

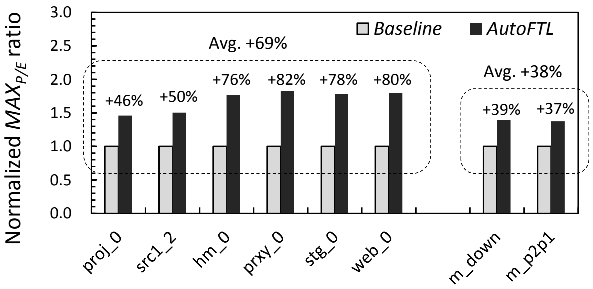


Figure 9: Comparisons of normalized $MAX_{P/E}$ ratios for eight traces.

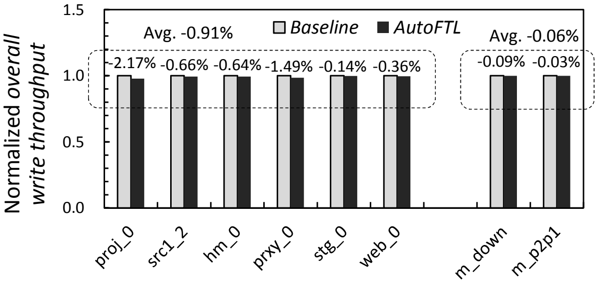


Figure 10: Comparisons of normalized overall write throughputs for eight traces.

On the other hand, for the mobile traces, AutoFTL improves $MAX_{P/E}$ by only 38%, on average, over baseline. Although more than 50% of write requests have inter-arrival times twice longer than $T_{PROG}^{effective}$, autoFTL could not improve $MAX_{P/E}$ as much as expected. This is because the size of the circular buffer is too small for buffering the increase in the buffer utilization caused by the garbage collection. For example, when a NAND block is erased by the fast speed erase mode, the buffer utilization is increased by 40% for the mobile environment while the effect of the fast erase mode on the buffer utilization is less than 0.1% for the enterprise environment. Moreover, by the same reason, the slow erase speed mode cannot be used in the mobile environment.

5.4 Overall Write Throughput Analysis

Although autoFTL uses slow write modes frequently, the decrease in the overall write throughput over baseline is less than 2.2% as shown in Figure 10. For proj_0 trace, the overall write throughput is decreased by 2.2%. This is because, in proj_0 trace, the circular buffer may become full by highly clustered write requests. When the circular buffer becomes full, if the foreground garbage collection should be invoked, the write response time of NAND chips can be directly affected. Although inter-arrival times in prxy_0 trace are relatively long over other enterprise traces, the overall write throughput is degraded more than the other enterprise traces. This is because almost all the write requests exhibit inter-arrival

public. We would welcome input from anyone who believes they have such traces to make available.

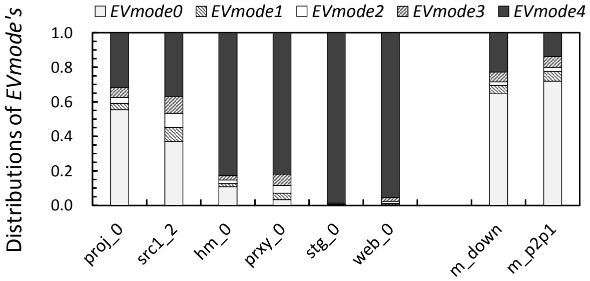
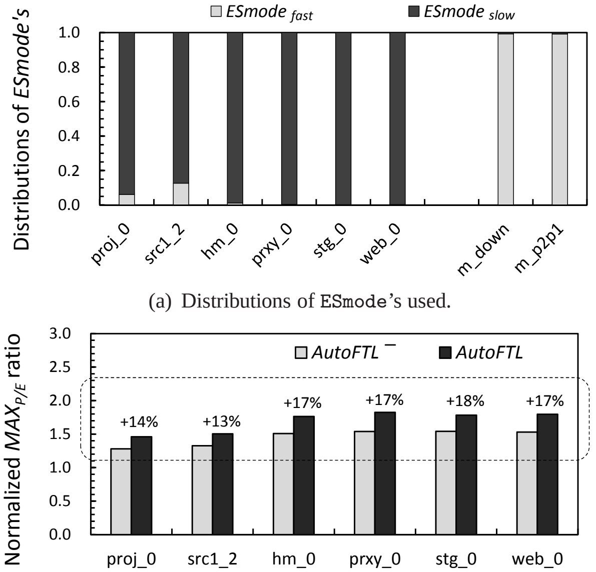


Figure 11: Distributions of EVmode's used.



(a) Distributions of ESmode's used.

(b) The effect of $\texttt{ESmode}_{slow}$ on improving $MAX_{P/E}$.

Figure 12: Distributions of ESmode's used and the effect of ESmode's on $MAX_{P/E}$.

times shorter than 10 *m*s so that the background garbage collection is not invoked at all[7]. As a result, the foreground garbage collection is more frequently invoked, thus increasing the write response time.

We also evaluated if there is an extra delay from a host in sending a write request to the circular buffer because of DPES. Although autoFTL introduced a few extra queueing delay for the host, the increase in the average queueing delay per request was negligible compared to $T_{PROG}^{effective}$. For example, for src1_2 trace, 0.4% of the total programmed pages were delayed, and the average queueing delay per request was 2.6 *u*s. For stg_0 trace, less than 0.1% of the total programmed pages were delayed, and the average queueing delay per request was 0.1 *u*s.

5.5 Detailed Analysis

We performed a detailed analysis on the relationship between the erase voltage/speed modes and the improvement of $MAX_{P/E}$. Figure 11 presents distributions of EVmode's used for eight I/O traces. Distributions of EVmode's exactly correspond to the improvements of $MAX_{P/E}$ as shown in Figure 9; the more frequently a low erase voltage mode is used, the higher the endurance gain is. In our evaluations for eight I/O traces, lazy erases are rarely used for all the traces.

Figure 12(a) shows distributions of ESmode's for eight I/O traces. Since the slow erase mode is selected by using the effective buffer utilization, there are little chances for selecting the slow erase mode for the mobile traces because the size of the circular buffer is only 80 KB. On the other hand, for the enterprise environment, there are more opportunities for selecting the slow erase mode. Even for the traces with short inter-arrival times such as proj_0 and src1_2, only 5%∼10% of block erases used the fast erase mode.

We also evaluated the effect of the slow erase mode on the improvement of $MAX_{P/E}$. For this for evaluation, we modified our autoFTL so that $\texttt{ESmode}_{fast}$ is always used when NAND blocks are erased. (We represent this technique by autoFTL⁻.) As shown in Figure 12(b), the slow erase mode can improve the NAND endurance gain up to 18%. Although the slow erase mode can increase the buffer utilization, its effect on the write throughput was almost negligible.

6 Related Work

As the endurance of recent high-density NAND flash memory is continuously reduced, several system-level techniques which exploit the physical characteristics of NAND flash memory have been proposed for improving the endurance and lifetime of flash-based storage systems [8, 7, 23, 24].

Mohan *et al.* investigated the effect of the damage recovery on the SSD lifetime for enterprise servers [8]. They showed that the overall endurance of NAND flash memory can be improved with its recovery nature. Our DPES technique does not consider the self-recovery effect, but it can be easily extended to exploit the physical characteristic of the self-recovery of flash memory cells.

Lee *et al.* proposed a novel lifetime management technique that guarantees the lifetime of storage devices by intentionally throttling write performance [7]. They also exploited the self-recovery effect of NAND devices, so as to lessen the performance penalty caused by write throttling. Unlike Lee's work (which sacrifices write performance for guaranteeing the storage lifetime), our DPES technique improves the lifetime of NAND devices without degrading the performance of NAND-based storage systems.

Wu *et al.* presented a novel endurance enhancement

[7] In our autoFTL setting, the background garbage collection is invoked when a idle time between two consecutive requests is longer than 300 *m*s.

technique that boosts recovery speed by heating a flash chip under high temperature [23]. By leveraging the temperature-accelerated recovery, it improved the endurance of SSDs up to five times. The major drawback of this approach is that it requires extra energy consumption to heat flash chips and lowers the reliability of a storage device. Our DPES technique improves the endurance of NAND devices by lowering the erase voltage and slowing down the erase speed without any serious side effect.

Jeong *et al*. proposed an earlier version of the DPES idea and demonstrated that DPES can improve the NAND endurance significantly without sacrificing the overall write throughput [24]. Unlike their work, however, our work treats the DPES approach in a more complete fashion, extensively extending the DPES approach in several dimensions such as the erase speed scaling, shallow erasing and lazy erase scheme. Furthermore, more realistic and detailed evaluations using the timing-accurate emulator are presented in this paper.

7 Conclusions

We have presented a new system-level approach for improving the lifetime of flash-based storage systems using dynamic program and erase scaling (DPES). Our DPES approach actively exploits the tradeoff relationship between the NAND endurance and the erase voltage/speed so that directly improves the NAND endurance with a minimal decrease in the write performance. Based on our novel NAND endurance model and the newly defined interface for changing the NAND behavior, we have implemented autoFTL, which changes the erase voltage and speed in an automatic fashion. Moreover, by making the key FTL modules (such as garbage collection and wear leveling) DPES-aware, autoFTL can significantly improve the NAND endurance. Our experimental results show that autoFTL can improve the maximum number of P/E cycles by 69% for enterprise traces and 38% for mobile traces, on average, over an existing DPES-unaware FTL.

The current version of autoFTL can be further improved in several ways. For example, we believe that the current mode selection rules are rather too conservative without adequately reflecting the varying characteristics of I/O workload. As an immediate future task, we plan to develop more *adaptive* mode selection rules that may adaptively adjust the buffer utilization boundaries for selecting write modes.

Acknowledgements

We would like to thank Erik Riedel, our shepherd, and anonymous referees for valuable comments that greatly improved our paper. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT and Future Planning (MSIP) (NRF-2013R1A2A2A01068260). This research was also supported by Next-Generation Information Computing Development Program through NRF funded by MSIP (No. 2010-0020724). The ICT at Seoul National University and IDEC provided research facilities for this study.

References

[1] S.-H. Shin *et al.*, “A New 3-bit Programming Algorithm Using SLC-to-TLC Migration for 8 MB/s High Performance TLC NAND Flash Memory,” in *Proc. IEEE Symp. VLSI Circuits*, 2012.

[2] J. Choi *et al.*, “3D Approaches for Non-volatile Memory,” in *Proc. IEEE Symp. VLSI Technology*, 2011.

[3] A. A. Chien *et al.*, “Moore’s Law: The First Ending and A New Beginning,” *Tech. Report, Dept. of Computer Science, the Univ. of Chicago*, TR-2012-06.

[4] J.-W. Hsieh *et al.*, “Efficient Identification of Hot Data for Flash Memory Storage Systems,” *ACM Trans. Storage*, vol. 2, no. 1, pp. 22-40, 2006.

[5] F. Chen *et al.*, “CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory Based Solid State Drives,” in *Proc. USENIX Conf. File and Storage Tech.*, 2011.

[6] S. Lee *et al.*, “Improving Performance and Lifetime of Solid-State Drives Using Hardware-Accelerated Compression,” *IEEE Trans. Consum. Electron.*, vol. 57, no. 4, pp. 1732-1739, 2011.

[7] S. Lee *et al.*, “Lifetime Management of Flash-Based SSDs Using Recovery-Aware Dynamic Throttling,” in *Proc. USENIX Conf. File and Storage Tech.*, 2012.

[8] V. Mohan *et al.*, “How I Learned to Stop Worrying and Love Flash Endurance,” in *Proc. USENIX Workshop Hot Topics in Storage and File Systems*, 2010.

[9] N. Mielke *et al.*, “Bit Error Rate in NAND Flash Memories,” in *Proc. IEEE Int. Reliability Physics Symp.*, 2008.

[10] K. F. Schuegraf *et al.*, “Effects of Temperature and Defects on Breakdown Lifetime of Thin SiO_2 at Very Low Voltages,” *IEEE Trans. Electron Devices*, vol. 41, no. 7, pp. 1227-1232, 1994.

[11] S. Cho, "Improving NAND Flash Memory Reliability with SSD Controllers," in *Proc. Flash Memory Summit*, 2013.

[12] S. Lee *et al.*, "FlashBench: A Workbench for a Rapid Development of Flash-Based Storage Devices," in *Proc. IEEE Int. Symp. Rapid System Prototyping*, 2012.

[13] R.-S. Liu *et al.*, "Optimizing NAND Flash-Based SSDs via Retention Relaxation," in *Proc. USENIX Conf. File and Storage Tech.*, 2012.

[14] K.-D. Suh *et al.*, "A 3.3 V 32 Mb NAND Flash Memory with Incremental Step Pulse Programming Scheme," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1149-1156, 1995.

[15] JEDEC Standard, "Stress-Test-Driven Qualification of Integrated Circuits," JESD47H.01, 2011.

[16] D.-W. Lee *et al.*, "The Operation Algorithm for Improving the Reliability of TLC (Triple Level Cell) NAND Flash Characteristics," *Proc. IEEE Int. Memory Workshop*, 2011.

[17] J. Yang, "High-Efficiency SSD for Reliable Data Storage Systems," *Proc. Flash Memory Summit*, 2011.

[18] R. Frickey, "Data Integrity on 20 nm NAND SSDs," *Proc. Flash Memory Summit*, 2012.

[19] L.-P. Chang, "On Efficient Wear Leveling for Large-Scale Flash-Memory Storage Systems," in *Proc. ACM Symp. Applied Computing*, 2007.

[20] IBM "Kernel APIs, Part 3: Timers and Lists in the 2.6 Kernel," http://www.ibm.com/developerworks/library/l-timers-list/.

[21] D. Narayanan *et al.*, "Write Off-Loading: Practical Power Management for Enterprise Storage," in *Proc. USENIX Conf. File and Storage Tech.*, 2008.

[22] http://www.ubuntu.com/download

[23] Q. Wu *et al.*, "Exploiting Heat-Accelerated Flash Memory Wear-Out Recovery to Enable Self-Healing SSDs," in *Proc. USENIX Workshop Hot Topics in Storage and File Systems*, 2011.

[24] J. Jeong *et al.*, "Improving NAND Endurance by Dynamic Program and Erase scaling," in *Proc. USENIX Workshop Hot Topics in Storage and File Systems*, 2013.

[25] M. Lenzlinger, "Fowler-Nordheim tunneling into thermally grown SiO2," *Journal of Applied Physics*, vol. 40, no. 1, pp. 278-283, Jan. 1969.

Appendix

A Verification Results of the NAND Endurance Model

In this paper, we use a linear approximation model which simplifies the wear-out behavior over P/E cycles. Our current linear model can overestimate the effective wearing under low erase voltage scaling ratios while it can underestimate the effective wearing under high erase voltage scaling ratios. We verified that, by the combinations of over-/under-estimations of the effective wearing in our model, the current linear model achieves a reasonable accuracy with an up to 10% overestimation while supporting a simple software implementation. We will briefly explain our verification process.

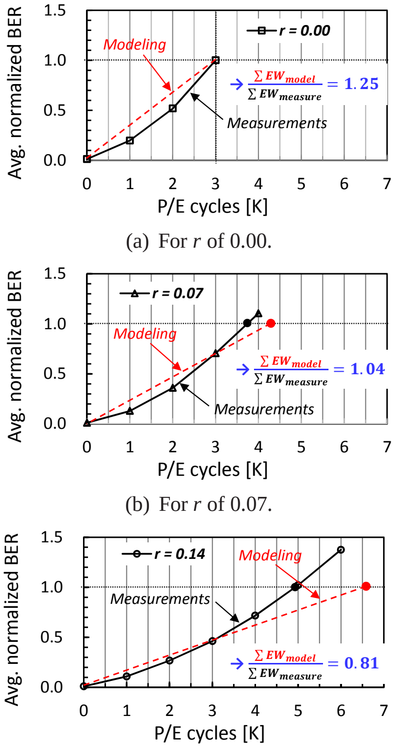
Figures 13(a), (b), and (c) are the extended versions of Figure 3(a) which shows average (retention) BER variations over P/E cycles. Dotted lines with red color are the linearly approximated model of BER in this paper and solid lines with black color are measured results (i.e., the average normalized BER over more than 80 measurements). As shown in three graphs, since the linear approximation model uses only two points of BERs at 0K and 3K P/E cycles, the effective wearing (i.e., the retention BER) is overestimated in the range of low P/E cycles (the number of P/E cycles $\leq$ 3K) while it is underestimated in the range of high P/E cycles (the number of P/E cycles $>$ 3K). As described in Sections 1 and 3.1, since the NAND endurance is represented by the number of P/E cycles where the total sum of the effective wearing is 3K, the NAND endurance by our model is underestimated in the range of low P/E cycles while it is overestimated in the range of high P/E cycles.

In order for a fair comparison between modeling and measurements, we calculate *the relative total wearing* which is the ratio of the total sum of effective wearing with modeling over that with measurements. As shown in Figure 13(a), for example, the relative total wearing is 1.25 for r of 0.00, which means that our model overestimates the total sum of the effective wearing by 25%, on average, compared to real measurements. On the other hand, for r of 0.14, our model underestimates the total sum of the effective wearing by about 19%, as shown in Figure 13(c).

As described in Section 3.1, the NAND endurance is inversely proportional to the total sum of the effective wearing. The upper bound of the estimation error caused by our model can be calculated from the relative wearing for each extreme case as follows:

(1) If we always erase a NAND block with the normalized erase voltage of 1.00, the NAND endurance is underestimated by 20% (=1/1.25 - 1).

(2) If we always use the normalized erase voltage of 0.86 (i.e., r is 0.14), the NAND endurance is overestimated by 23% (=1/0.81 - 1).



(a) For r of 0.00.

(b) For r of 0.07.

(c) For r of 0.14.

Figure 13: Comparisons of BER variations from measurements and modeling.

Figure 14 illustrates the relationship between the normalized erase voltage and the relative total wearing. However, overestimating the effective wearing is not a problem because the endurance gain is under-estimated. So, we can concentrate on checking how much wearing our model underestimates. The lower the erase voltage is used, the more endurance gain can be achieved while the probability of estimation error is also increased.

Since we use the conservative M_{Pi} scaling model as shown in Figure 5(c), the minimum available erase voltage is only 0.89 (i.e., r is 0.11). Dotted rectangle with red color in Figure 14 represents the total range of used erase voltages and the relative total wearing. Since the lower bound of the relative total wearing is 0.91, the maximum overestimation in the endurance gain is only

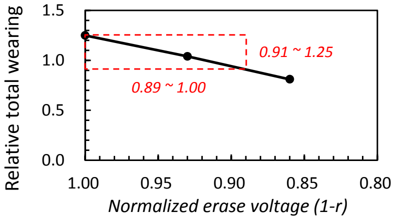


Figure 14: The relationship between the normalized erase voltage and the relative total wearing.

9.9% (=1/0.91). Therefore, we conclude that our device model does not overestimate the endurance gain more than 10%.

B How An Erase Voltage Affects Programmed Cells

We will briefly explain how an erase voltage affects threshold voltages of differently programmed cells.

(1) The effective erase voltage is automatically scaled depending on the number of electrons for each programmed cell.

(2) The number of electrons in each cell eventually converges on the same value after a sufficiently long erase time, regardless of their initial number of electrons. It is widely known that erasing NAND memory cells is based on the FN-tunneling mechanism. Since the probability of electron's tunneling has an exponential dependence on the applied voltage [25], how much electrons evicted during erasing is also proportional to the initial *built-in potential* of each programmed cell as well as the applied erase voltage. As a result, after a long erase time, the number of electrons in each cell eventually converges on the same value.

(3) The number of electrons after an erase operation is mainly dependent on the initially applied erase voltage. Since the FN-tunneling occurs only when the voltage difference across the tunnel oxide is higher than certain critical voltage [25], an erase operation will be stopped when the total erase voltage is reduced to the critical voltage. This is because the built-in potential of a memory cell is naturally reduced as electrons are evicted during the erase operation, and the FN-tunneling process will be stopped after the total erase voltage becomes effectively lower than the critical voltage. The higher the erase voltage is used, the less the number of electrons after erasing.

We will explain why this is the case using a simple example as illustrated in Figure 15. When an erase voltage (i.e., 14 V) is applied to two memory cells whose threshold voltages (i.e., the built-in potential) are 0 V and 4 V, the total erase voltages applied to each cell are initially 14 V and 18 V, respectively. Assuming that the critical voltage for the FN-tunneling is 10 V, the voltage differences across the tunnel oxide for each cell are differently reduced but eventually converge on the same voltage (i.e., 10 V) after a long erase time. Finally, a threshold voltage of each cell is reduced to -4 V. When a lower erase voltage (e.g., 13 V) is used for erasing a NAND block, a threshold voltage of each cell is reduced to only -3 V so that a threshold voltage window is reduced by 1 V compared to that by erasing with 14 V.

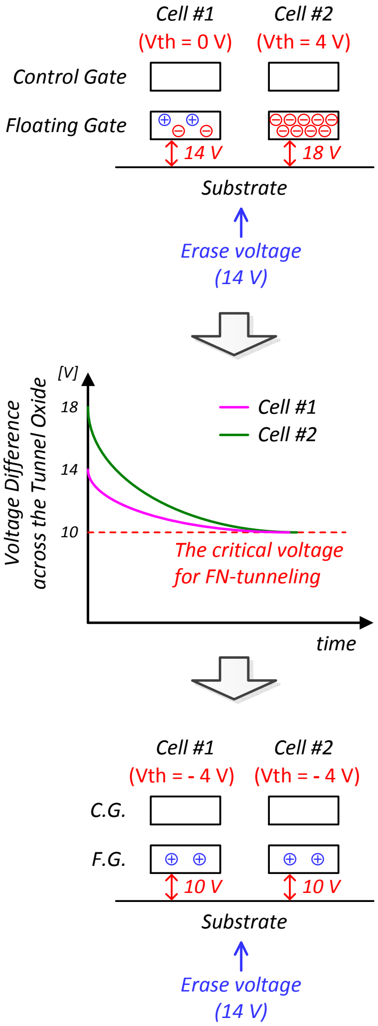


Figure 15: An example of how an applied erase voltage affects threshold voltages of differently programmed cells.