Improving NAND Endurance by Dynamic Program and Erase Scaling

Jaeyong Jeong, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim
Department of Computer Science and Engineering,
Seoul National University, Korea
$\{$*jyjeong, shanehahn, chamdoo, jihong*$\}$*@davinci.snu.ac.kr*

Abstract

We propose a new approach, called dynamic program and erase scaling (DPES), for improving the endurance of NAND flash memory. The DPES approach is based on our key finding that the NAND endurance is dependent on the erase voltage as well as the number of P/E cycles. Since the NAND endurance has a near-linear dependence on the erase voltage, lowering the erase voltage is an effective way of improving the NAND endurance. By modifying NAND chips to support multiple write modes with different erase voltages, DPES enables a flash software to exploit the new tradeoff between the NAND endurance and write speed. In this paper, we present a novel NAND endurance model which accurately captures the tradeoff relationship between the endurance and write speed under dynamic program and erase scaling. Based on our NAND endurance model, 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 45% over an existing DPES-unaware FTL with less than 0.2% 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 $M_{P/E}$ 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):

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

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 $M_{P/E}$ of P/E cycles. For example, a recent study [8] suggests $M_{P/E}$ 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 $M_{P/E}$. The key intuition of our approach, which is motivated by a NAND device physics model on the endurance degradation, is that different P/E settings can affect the NAND endurance differently. By modifying a NAND device to support multiple write modes (which have different impact on the NAND endurance) and allowing a firmware/software module to choose the most appropriate write mode (e.g., depending on a given workload), DPES can significantly increase $M_{P/E}$.

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 is proportional to the stress voltage as well as the length of the stressed time interval [10], reducing the stress voltage is an effective way of improving the NAND endurance. Our measurement results with recent 20 nm-node NAND chips show that when the stress voltage (particularly, the erase voltage) is reduced by 17% during P/E cycles, $M_{P/E}$ can increase on average by 96%. However, in order to lower the erase voltage, it is necessary to form narrow threshold voltage distributions after program operations. Since shortening the width W_{Vth} of a threshold voltage distribution requires a fine-grained control during a program operation, the program time is increased if a lower erase voltage is used.

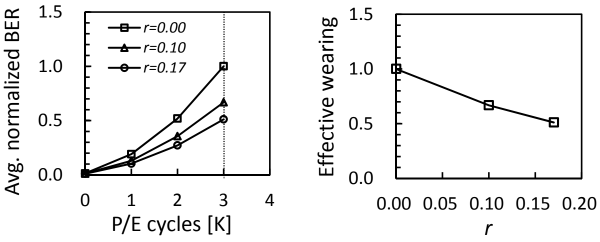
Our DPES approach exploits the tradeoff relationship between the NAND endurance and write speed at the firmware-level (or the software level in general) so that the NAND endurance is improved while the write throughput is not affected. For example, since the maximum performance of NAND flash memory is not always needed in real workloads, a DPES-based technique can exploit idle times between consecutive write requests for reducing W_{Vth}. If such idle times can be automatically estimated by a firmware/system software, the DPES-based technique can choose the most appropriate write mode for each write request. By aggressively selecting an endurance-enhancing write mode when a large idle time is available, the NAND endurance can be significantly increased because less damaging P/E's 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 write 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 program and erase voltages in an automatic fashion, thus improving the NAND endurance with a negligible degradation in the write throughput. Since no NAND chip currently allows an FTL firmware to change its program and erase voltages dynamically, we evaluated the effectiveness of autoFTL with the *FlashBench* emulation environment [11] using a DPES-enabled NAND simulation model (which supports multiple write modes). Our experimental results using various I/O traces show that autoFTL can improve $M_{P/E}$ by 45% over an existing DPES-unaware FTL with less than 0.2% decrease in the overall write throughput.

The rest of the paper is organized as follows. In Section 2, we present the proposed DPES approach in detail. Section 3 describes our DPES-aware autoFTL. Experimental results follow in Section 4. Finally, Section 5 concludes with summary and future work.

2 Dynamic Program and Erase Scaling

The DPES approach is based on our key finding that the NAND endurance is dependent on the erase voltage as well as the number of P/E cycles. In this section, we explain the effect of erase voltage scaling on improving the NAND endurance and describe the dynamic program scaling method for lowering the erase voltage. 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.



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

(b) Effective wearing over different r's

Figure 1: The effect of lowering the erase voltage on the NAND endurance.

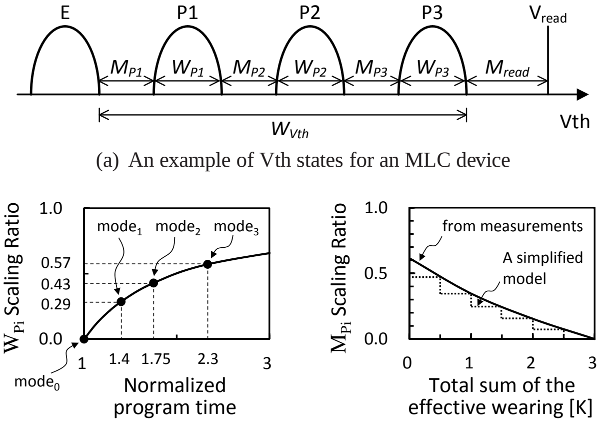
2.1 Erase Scaling and its Effect on NAND Endurance

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[1] by changing the erase voltage. In our tests, we used the NAND retention BER (i.e., a BER after 10 hours' baking at $125\,^{\circ}\mathrm{C}$[2]) as a measure for quantifying the wearing degree of a NAND chip [9]. Fig. 1(a) shows how the retention BER changes, on average, as the number of P/E cycles increases while varying erase voltage scaling ratios. (We denote an erase voltage scaling ratio by r. When r is set to x%, the erase voltage is reduced by x%.) The retention BERs were normalized over the retention BER after 3K P/E cycles when the default erase voltage was used. As shown in Fig. 1(a), the more the erase voltage is reduced (i.e., the higher r's), the less the retention BERs. For example, when the erase voltage is reduced by 17%, the normalized retention BER is reduced by 49% after 3K P/E cycles over the default erase voltage case. 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 wear-*

[1] In a NAND cycling test, program and erase operations are repeated 3,000 times (which are roughly equivalent to $M_{P/E}$ of a recent 20 nm-node NAND device [3]). Our cycling tests for each case are performed with more than 20 blocks which are randomly selected from 5 chips.

[2] This is a standard NAND retention evaluation procedure specified by JEDEC.



(a) An example of Vth states for an MLC device

(b) W_{Pi} scaling ratios

(c) M_{Pi} scaling ratios

Figure 2: W_{Pi} scaling and M_{Pi} scaling for dynamic program scaling.

ing), 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].

As shown in Fig. 1(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 effective wearing with the default erase voltage is 3K. On the other hand, if the erase voltage was set to 17% less than the default voltage, the total sum of the effective wearing is only 1.53K because the effective wearing with r of 0.17 is 0.51. As a result, $M_{P/E}$ can be increased almost twice as much when the erase voltage is reduced by 17% over the default case. In this paper, we will use a simple endurance model with four different write modes (as described in Section 2.3).

2.2 Dynamic Program Scaling

In order to use a reduced erase voltage when a NAND block is erased, it is necessary to change program bias conditions dynamically so that narrow threshold voltage distributions can be formed after program operations. Since the erase voltage is proportional to the maximum program voltage (which is dependent on the width of threshold voltage distributions), extra threshold voltage margins from narrow threshold voltage distributions can be used to lower the erase voltage. Fig. 2(a) illustrates how threshold voltage distributions for four MLC states are typically formed. Since we need to shorten the width (i.e., W_{Vth}) of threshold voltage distributions for reducing the erase voltage, we reduce M_{Pi}'s and W_{Pi}'s during program times. In conventional NAND chips, the voltage gap M_{Pi} between two adjacent states is kept large enough to preserve data integrity. On the other hand, the width W_{Pi} of a program state is mainly affected by a program-speed requirement. If a slow program is acceptable, W_{Pi} can be made short [12]. Since flash manufacturers should guarantee the reliability and performance of NAND flash memory throughout the storage lifespan, M_{Pi}'s and W_{Pi}'s are usually *fixed* under the worst-case operating conditions of a target product. Since W_{Pi}'s are fixed by a flash manufacturer, a program operation takes a constant program time, which we represent by T_{PROG}.

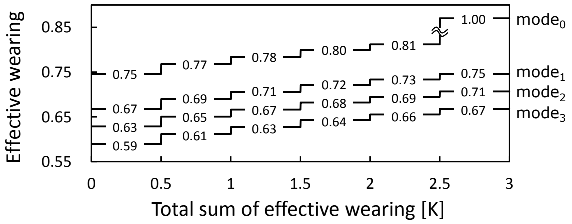


Figure 3: The proposed NAND endurance model for DPES-enabled NAND blocks.

The proposed DPES approach is unique in that it allows M_{Pi}'s and W_{Pi}'s to be changed dynamically depending on workload variations. We scale W_{Pi} based on a program time requirement. Fig. 2(b) shows the relationship between the program time and its corresponding W_{Pi} scaling ratio based on our NAND characterization study. The program time is normalized over T_{PROG}. For example, in the case of $\texttt{mode}_3$ when the program time is 2.3 times longer than T_{PROG}, W_{Pi} can be reduced by 57%. On the other hand, M_{Pi} is scaled based on the total sum of the effective wearing to guarantee the required reliability (i.e., data retention). Fig. 2(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 Fig. 2(c)). Dynamic program scaling can be easily integrated into an existing NAND controller with a negligible time overhead (e.g., less than 0.1% of T_{PROG}) and a very small space overhead (e.g., 2 bits per block).

2.3 NAND Endurance Model

Combining erase voltage scaling and program scaling, we developed a novel NAND endurance model that can be used with DPES-enabled NAND chips. Fig. 3 shows our proposed NAND endurance model with four write modes, $\texttt{mode}_0 \sim \texttt{mode}_3$. $\texttt{Mode}_0$ (which uses the largest erase voltage) is the fastest write mode with no slowdown in the write speed while $\texttt{mode}_3$ (which uses the smallest erase voltage) represents the slowest write mode with the largest wearing gain. Our proposed NAND endurance model takes account of both W_{Pi} scaling and M_{Pi} scaling described in Figs. 2(b) and 2(c).

3 AutoFTL: DPES-Aware FTL

Based on our NAND endurance model presented in Section 2.3, we have implemented autoFTL, the first DPES-aware FTL, which automatically changes program speeds depending on write throughput requirements. AutoFTL is based on a page-level mapping FTL

[3] Since the normalized retention BER is reduced by 49% when the erase voltage is reduced by 17%, the effective wearing becomes 0.51. When the default erase voltage is used, the effective wearing is 1.

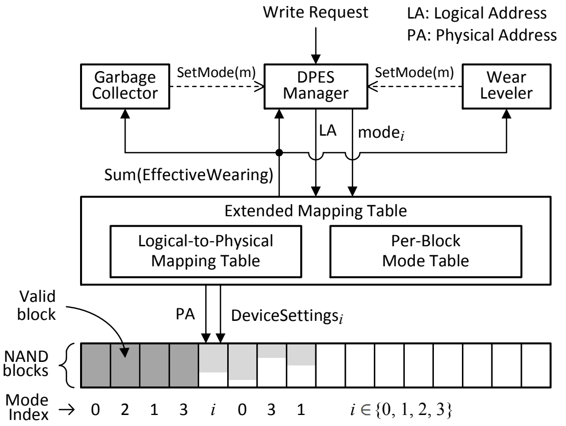


Figure 4: An organizational overview of autoFTL.

with additional modules for DPES support. Fig. 4 shows an organizational overview of autoFTL. The DPES manager is responsible for choosing the most appropriate write mode for each write operation. Unlike conventional FTLs, autoFTL maintains per-block mode 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.

In selecting a write mode for a write request, the DPES manager exploits idle times between consecutive write requests so that autoFTL can increase $M_{P/E}$ without incurring additional decrease in the overall write throughput. In autoFTL, the DPES manager uses a simple circular buffer for estimating the available program time for a given workload. Table 1 illustrates how the DPES manager decides a write mode using a circular buffer. The circular buffer queues incoming write requests before they are written, and the mode selector adaptively changes a write mode. The mode selector chooses the write mode, $\texttt{mode}_i$, depending on the buffer utilization ratio (which represents how much of the circular buffer is filled by outstanding requests). For example, if the utilization ratio is lower than 40%, the write request in the head of the circular buffer is programmed to a NAND chip with $\texttt{mode}_3$. In the current version of autoFTL, we used rather conservative write-mode selection rules so that the overall write throughput is not affected by DPES.

Since erase operations are performed at the NAND block level, the per-block mode table maintains four linked lists of blocks using the same write 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 voltage and reference voltages for read/verify operations) for the current write mode using the per-block mode table. 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.

In autoFTL, both the garbage collector and wear leveler are modified to be DPES-aware. In the garbage collector, when a background GC is invoked, the slowest write mode is used in copying valid data. On the other hand, when a foreground GC is invoked, the fastest mode is used so that valid data can be moved as soon as possible. The wear leveler tries to evenly distribute the total sum of effective wearing, not the number of P/E cycles, to NAND blocks by employing a DPES-aware data separation technique in the existing dual-pool algorithm [13].

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

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

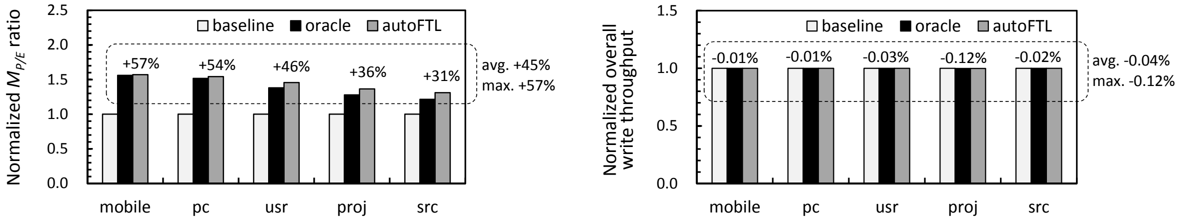
4 Experimental Results

In order to evaluate the effectiveness of the proposed autoFTL, we used a unified development environment, called *FlashBench* [11], for NAND flash-based storage devices. For our evaluation, we modified a NAND flash model in FlashBench to support DPES-enabled NAND flash chips with four write modes as shown in Fig. 3. Since the organization of mobile storage systems and enterprise storage systems are quite different, we used two FlashBench configurations in our experiments. For a mobile benchmark, FlashBench was configured to have one channel with four NAND flash chips. For PC and enterprise benchmarks, FlashBench was configured to have four channels with four chips per channel. Each NAND flash chip has 128 blocks which are composed of 128 4-KB pages. The page program time (i.e., T_{PROG}) was set to 1.3 ms. The size of the circular buffer used in autoFTL was set to 32 KB and 32 MB, respectively, for mobile and enterprise benchmarks.

We evaluated $M_{P/E}$ and the overall write throughput for three different techniques: baseline, autoFTL, and oracle. Baseline is an existing DPES-unaware FTL that always uses the fastest mode for writing data. AutoFTL is the proposed DPES-aware FTL that selects a write mode based on the utilization of a circular buffer. Oracle is a simple off-line algorithm with an oracle predictor on the arrival time of the next request. Since oracle has the full knowledge of future write requests, it can choose the most appropriate write mode for the next request without a circular buffer. (That is, oracle shows the performance upper bound when no circular buffer is used.)

We used five different I/O traces: `mobile` (collected from a P2P application on an Android smartphone), `pc` (extracted from a PC user), and `usr`, `proj`, and `src` (chosen from the MS-Cambridge benchmarks [14], which represent the enterprise storage workload.). Table 2 summarizes the distributions of inter-arrival times between two consecutive write requests for five I/O traces. Inter-arrival times were normalized over T_{PROG}.

In order to measure $M_{P/E}$, each trace was repeated until the total sum of the effective wearing reached 3K. Measured $M_{P/E}$ values were normalized over that of



(a) Normalized $M_{P/E}$ ratio

(b) Normalized overall write throughput

Figure 5: Comparisons of normalized $M_{P/E}$ ratio and overall write throughput for five traces with different techniques.

Trace	Distributions of normalized inter-arrival times t over T_{PROG} [%]			
	$t < 1.4$	$1.4 \leq t < 1.75$	$1.75 \leq t < 2.3$	$t \geq 2.3$
`mobile`	1.6	0.9	2.4	95.1
`pc`	12.0	9.0	18.0	61.0
`usr`	55.6	5.2	9.8	29.4
`proj`	71.1	0.8	2.2	25.9
`src`	90.3	0.1	0.1	9.5

Table 2: Characteristics of traces used for evaluations.

baseline. Fig. 5(a) shows normalized $M_{P/E}$ ratios for five traces with three different techniques. AutoFTL improves $M_{P/E}$ by 45%, on average, over baseline. Overall, the improvement on $M_{P/E}$ is proportional to inter-arrival times listed in Table 2; the longer inter-arrival times are, the more likely slow write modes are selected. Although autoFTL uses slow write modes frequently, the decrease in the overall write throughput over baseline is less than 0.2% as depicted in Fig. 5(b). We also evaluated if there is an extra delay in sending a write request to the circular buffer of autoFTL over baseline. The increase in the average queuing delay per request was negligible compared to T_{PROG} (i.e., 1300 μs). For `mobile`, there was a delay of 0.3 μs while, for `src`, there was a delay of 50.8 μs. One interesting observation is that autoFTL performs better than oracle, improving $M_{P/E}$ by 6%, on average, over oracle. This is because autoFTL can choose slow write modes even for highly clustered consecutive writes because write modes are determined by the circular buffer utilization which reflects longer-term write throughput fluctuations. On the other hand, oracle, which has no buffering support, must choose fast write modes for the clustered writes.

5 Conclusions

We have presented a novel approach, DPES, for improving the endurance of NAND flash memory. Unlike existing lifetime-enhancing techniques, our DPES approach is unique in that it directly improves the NAND endurance by exploiting the device physics on the NAND endurance. Based on our novel NAND endurance model which accurately describes the effect of DPES on the NAND endurance, we have implemented autoFTL, which automatically changes program speeds and the erase voltage. Our experimental results show that autoFTL can improve the maximum number of P/E cycles by 45% over an existing DPES-unaware FTL with a minimal decrease in the write throughput. Since the DPES approach adds a new dimension on the NAND lifetime optimization problem, we expect that there are ample opportunities for future research issues. Most existing solutions for NAND endurance improvements may need to be revisited from the DPES' perspective.

Acknowledgements

We would like to thank Peter Desnoyers, our shepherd, and anonymous referees for valuable suggestions that greatly improved the paper. This research was supported by WCU (World Class University) program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (MEST) (No. R33-2012-000-10095-0). This work was also supported by NRF funded by MEST (No. 2012-0006417). 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. Lee *et al.*, "FlashBench: A Workbench for a Rapid Development of Flash-Based Storage Devices," in *Proc. IEEE Int. Symp. Rapid System Prototyping*, 2012.

[12] 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.

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

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