Subpage Programming for Extending the Lifetime of NAND Flash Memory

Jung-Hoon Kim
Memory Business
Samsung Electronics Corp.
jhdev.kim@samsung.com

Sang-Hoon Kim
Department of Computer Science
KAIST
sanghoon@calab.kaist.ac.kr

Jin-Soo Kim
College of Info. & Comm. Engineering
Sungkyunkwan University
jinsookim@skku.edu

***Abstract*—During the past decade, the density of NAND flash memory has been increased in many folds. The increase has been driven by storing multiple bits in a cell and scaling down the fabrication process. Such advance in manufacturing technology, however, has been significantly impaired the reliability of flash memory so that it becomes one of the major concerns in use of flash memory. Moreover, as flash memory writes data in the unit of flash page, the trend of the increase in page size worsens the reliability by amplifying a small update to a full flash page programming.**

In this paper, we propose a new programming method to improve the flash endurance cycle, especially when a small amount of data are written repeatedly. The proposed method, so called "subpage programming", partitions a page into smaller subpages. A small amount of data can be programmed to one of the subpages while the other subpages are inhibited from the programming by leveraging the mechanisms of flash cell programming. Thus, the number of flash cells that undergo programming is minimized. We evaluated the effect of the proposed subpage programming on real NAND flash memory chips from three different manufacturers. Our evaluation results show that subpage programming improves the flash endurance cycle by up to 258%.

I. Introduction

During the past decade, the density of NAND flash memory has been increased in many folds. The use of Multi-Level Cell (MLC) and Triple-Level Cell (TLC) technology, which stores 2 bits and 3 bits per cell, respectively, doubles and triples the density, compared to the Single-Level Cell (SLC) technology that stores only 1 bit per cell. Scaling down of the fabrication process enables to pack more cells on a NAND die. Such advance in the NAND flash manufacturing technology, however, has significantly impaired the reliability of NAND flash memory.

The reliability has been measured by means of the flash endurance cycle and the flash retention time. The flash endurance cycle is defined as the maximum number of Program and Erase cycles (P/E cycles) that NAND flash memory is guaranteed to work without being worn out. The flash retention time means the maximum period of time to keep the stored information unchanged without performing flash operations. Figure 1 illustrates the inverted relationship between the density and the reliability metrics since the advent of NAND flash memory technology. While the density has been increased by 32 times, the flash endurance cycle and the flash retention time

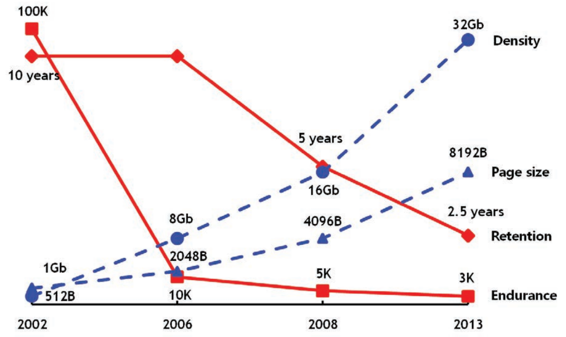


Fig. 1. The evolution of NAND flash memories

have been shorten to less than one-thirtieth and one-fourth, respectively [1]. As a result, the reliability issue becomes one of the major concerns in using flash memory.

The reliability problem is even worsened by the trend of the increase in page size. A page is the unit of read and program (i.e., write) operations in flash memory [2]. Manufacturers have been maximizing the throughput of those operations by increasing the page size and utilizing parallelism [3], [4], [5] as Figure 1 shows the trend. For MLC or TLC NAND flash memory, the page should be programmed at once all the time. This means that, even if only a small amount of data needs to be written to NAND flash memory, we should read corresponding page, update a small part of the page, and program the entire page. This unnecessarily wears out the unmodified part of the page, and thereby exacerbates the reliability issue.

In this paper, we propose a new page programming method, called *subpage programming*, to improve the flash endurance cycle, especially when a small amount of data is written repeatedly. A subpage is defined as a program unit that is smaller than the flash page size. To write a small amount of data, subpage programming only writes on a subpage and leaves the remaining part of the page unprogrammed by leveraging the mechanisms of cell programming. In this way, the number of flash cells that are programmed is minimized, and the flash endurance cycle is maximized. We have evaluated the effect of subpage programming on real NAND flash memory chips from three different manufacturers. Our evaluation results show that subpage programming can extend the flash endurance cycle by up to 258%.

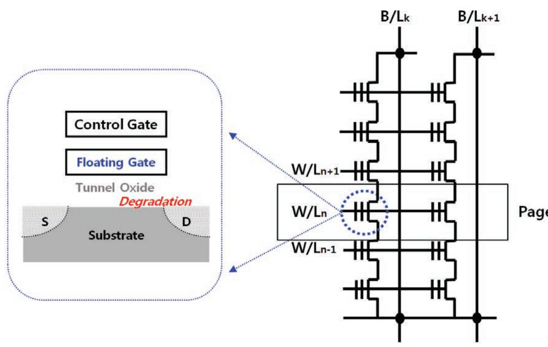


Fig. 2. The cell and array structure of NAND flash memory

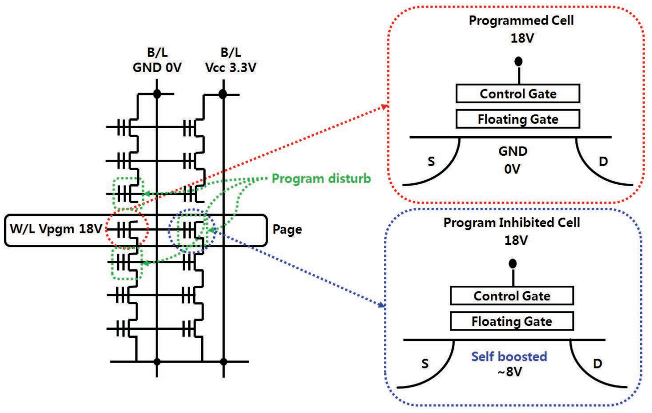


Fig. 3. Voltages across components in a programmed cell and a program inhibited cell

II. Background

A. NAND Flash Memory Cells and Oxide Degradation

NAND flash memory typically consists of transistors, called cells. Each cell is comprised of a floating gate, a control gate, and silicon substrate. Figure 2 shows the components of the flash cell. The floating gate is isolated from other components with oxide material [6] and can retain electrons inside it. The amount of electrons in the floating gate differentiates the logical value of the cell and is controlled with the Fowler-Nordheim (FN) tunneling mechanism [7]. High voltage (Vpgm) between the control gate and the substrate makes electrons tunnel through the oxide between the floating gate and the substrate. The electrons are trapped to the floating gate, programming the cell to a particular value. The electrons can be removed from the floating gate by reversing the voltage. Then, the value in the cell is erased and the cell is reverted to an unprogrammed state. However, repetitive FN tunneling can cause electrons to be trapped to the tunnel oxide [8] and degrade the integrity of the oxide [9]. The trapped electrons increase the potential barrier of the tunnel oxide and prevent other electrons from being injected to the floating gate. Also, the trapped electrons make retaining electrons in the floating gate difficult. Therefore, the cell of NAND flash memory can support only a limited number of programming/erase (P/E) cycles, which are known as the flash endurance cycle, and eventually becomes unable to operate. Note that the erase is done in a unit of erase block.

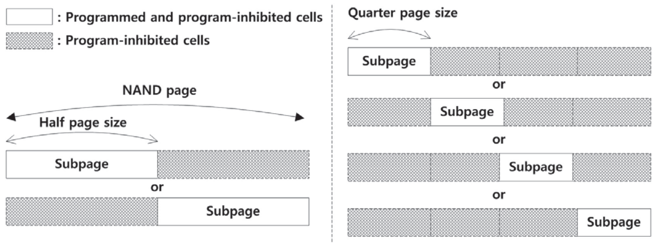


Fig. 4. Subpages in half or quarter of page size

B. Page Programming and Program Disturbance

In order to optimize the silicon surface area, the cells are arranged to a 2-D array structure, connected by Bit-Lines (B/L) and Word-Lines (W/L). B/L connects a column of the cell array in a serial manner through the source and the drain in the substrate. W/L connects a row of the cell array through control gates of cells. Those cells which share the same W/L form a page and multiple pages compose an erase block. Figure 2 sketches the cells and their array structure.

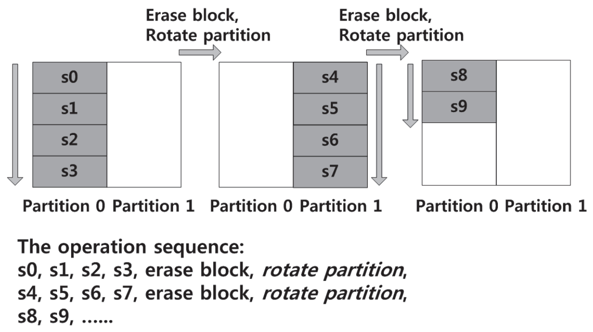
Modern NAND flash memory exploits the self-boosting mechanism [10] to program only particular cells in a page. In order to program a page, the W/L of the page is driven to Vpgm. To program a cell, the corresponding B/L is connected to ground so that the voltage gap between the control gate and the substrate becomes high enough to cause the FN tunneling. To inhibit a cell from programming, i.e., to keep the cell in an unprogrammed state, the corresponding B/L is driven to Vcc so that the source-drain channel is driven high by the self-boosting mechanism, which prevents the FN tunneling. Figure 3 shows the voltages across components for a programmed cell and a program inhibited cell. As the oxide degradation is proportional to the voltage that the tunnel oxide experiences, the program inhibited cell experiences less oxide degradation than the programmed cell.

During programming the page, the high programming voltage can affect other cells. This so called "program disturbance" phenomenon happens to the cells that are adjacent to the programmed cells. The program disturbance happens to the cells not only in the same page but also in adjacent pages. The affected cells can be unintentionally programmed as well. As the expected number of programmed cells per each page program is in proportion to the page size, the program disturbance becomes severe as the page size increases. Also, the cell with less oxide integrity is more likely to be affected by the program disturbance.

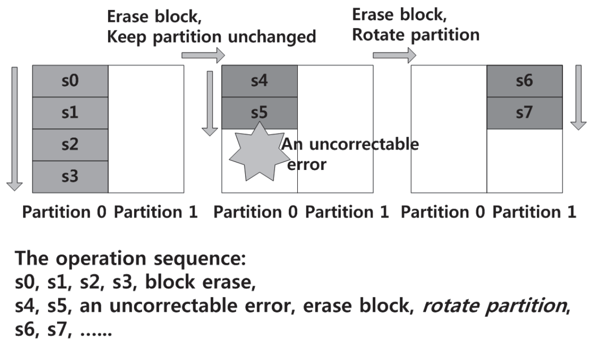
III. Subpage Programming

A. Subpage

We define a *subpage* as the program unit that is smaller than the page and in a multiple of the sector size. Figure 4 shows the cases where the subpage size is half or quarter of the page size. We also define a *partition* as a set of subpages in a block which have the same offset within the page. For example, if the subpage size is half of the page size, each block has two partitions, i.e., partition 0 and partition 1. The partition 0 consists of subpages that are on the lower part of pages, while the partition 1 on the upper part of pages. In the


Erase block,
Rotate partition
Erase block,
Rotate partition
s0
s1
s2
s3
s4
s5
s6
s7
s8
s9
Partition 0 Partition 1 Partition 0 Partition 1 Partition 0 Partition 1
The operation sequence:
s0, s1, s2, s3, erase block, rotate partition,
s4, s5, s6, s7, erase block, rotate partition,
s8, s9, ……

(a) Ping-pong


Erase block,
Keep partition unchanged
Erase block,
Rotate partition
s0
s1
s2
s3
s4
s5
An uncorrectable error
s6
s7
Partition 0 Partition 1 Partition 0 Partition 1 Partition 0 Partition 1
The operation sequence:
s0, s1, s2, s3, block erase,
s4, s5, an uncorrectable error, erase block, rotate partition,
s6, s7, ……

(b) Pong-pong

Fig. 5. Subpage programming in SP mode

same manner, we can define four partitions in a block if the subpage is quarter of the page size. The number of subpages in a partition is the same as that of pages in a block.

When programming a subpage with data, the rest of the page is filled with the value, typically 1, that corresponds to the unprogrammed cell. Thus, the cells outside of the subpage are program inhibited. As the program inhibited cells experience less oxide degradation, subpage programming inflicts less stress on cells than full page programming.

The key idea behind subpage programming is to program the subpage only when we need to write a small amount of data rather than to pad the subpage with original data and to program the entire page. This is highly useful in handling small writes that are likely to be overwritten soon, which is known to happen frequently in managing metadata.

Subpage programming is similar to partial page programming of SLC NAND flash memory [11] in the way that pages are programmed partially. The main differences come from how the unprogrammed cells are handled. In case of partial page programming, the remaining part of a page is programmed soon by successive partial program operations. In subpage programming, however, the rest of the page cannot be programmed by another subpage programming unless the block is erased.

B. Subpage Programming (SP) Mode

We call a block is in SP (Subpage Programming) mode when the block is dedicated for subpage programming. Full page programming is prohibited on the block in SP mode.

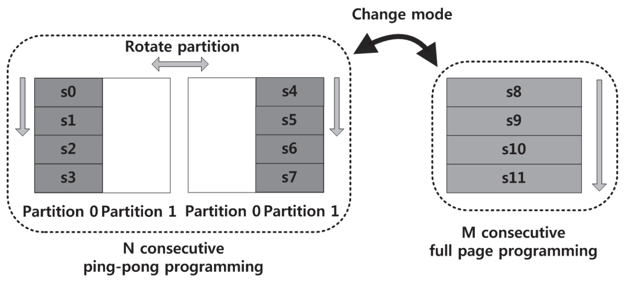


Fig. 6. Subpage programming in PSP mode

We impose two more restrictions in programming the block in SP mode. The first is that only the subpages belonging to the same partition can be used at a time. When a partition is used for a block in SP mode, the other partitions can be used only after the block is erased. The other restriction is that each subpage should be sequentially programmed from the lowest page number to the highest page number as full page programming does.

Since there are two or more partitions in SP-mode blocks, we need to consider the sequence in using the partitions. For simplicity, we consider two subpage programming orders in this paper: ping-pong and pong-pong. In the ping-pong order, the partitions are used in a round-robin fashion, rotated on every block erase. Figure 5(a) shows an example where subpages are programmed by the ping-pong order. The ping-pong order has the characteristic that all flash cells in a page get roughly the same level of stress similar to the case with full page programming. In the pong-pong order, a partition is continuously used until an uncorrectable bit error occurs from the partition. Then, the next partition is used until an uncorrectable bit error occurs again from the partition. In the pong-pong order, the flash memory cells in unused partitions can stay in the program inhibited state until they are actually used. Figure 5(b) shows an example of the pong-pong order.

C. Page and Subpage Programming (PSP) Mode

In order to evaluate the influence of subpage programming on the overall flash endurance cycle, we consider the case when a block switches the SP mode and the full page programming mode back and forth. More specifically, we call a block is in N:M PSP (Page and Subpage Programming) mode if it repeats to use the block in SP mode with the ping-pong order for N consecutive P/E cycles then in the normal mode for the following M consecutive P/E cycles. Figure 6 shows an example of programming sequence of a block in PSP mode.

IV. Evaluation Methodology

A. Flash Memory Chips

We have performed our evaluation on three MLC NAND flash memory chips manufactured by three different manufacturers. These chips are fabricated with 20nm technology and widely used in commercial products such as embedded systems and SSDs. Table I shows the features of the NAND flash memory chips.

All chips have the same page size of 8192 bytes. The size of the spare area, which is used to store ECC parity data and

TABLE I. FEATURES OF THE EVALUATED MLC NAND FLASH MEMORY CHIPS

Manu-facturer	Process (nm)	Page size (bytes)	Spare size (bytes)	Cell mode	Pages per block
A	27	8192	640	SLC-mode	64
				MLC-mode	128
B	26	8192	640	SLC-mode	128
				MLC-mode	256
C	25	8192	448	SLC-mode	128
				MLC-mode	256

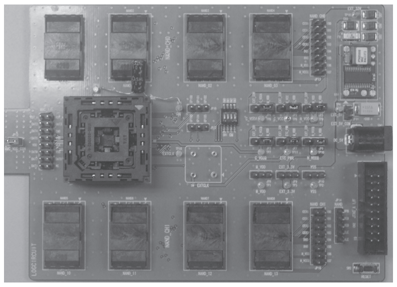


Fig. 7. The evaluation board for NAND flash memory chips

additional page information for each page, varies from 448 to 640 bytes. The cell mode represents a way of using pages in the block. In MLC-mode, we use both LSB pages and MLC pages of the MLC block. In SLC-mode, only LSB pages are used. Therefore, we can make use of only a half of pages in the SLC-mode block.

B. Flash Evaluation Board

Figure 7 shows the photograph of the flash evaluation board used in our experiments. The flash evaluation board consists of eight sockets for NAND chips and one socket for a controller chip. The controller chip has a System-on-Chip (SoC) architecture which integrates an external I/O interface, a NAND I/O interface, a data transmission buffer, a microprocessor, and memory components. The external I/O interface is used to communicate with the host system. The NAND I/O interface transfers data between the NAND flash memory chips and the data transmission buffer. The NAND I/O interface also supports 48bit BCH error correction codes [12] in 2KB granularity.

C. Endurance Test Sequence

We implemented an in-house microbenchmark that measures the flash endurance cycle, and ran the benchmark on the evaluation board at a room temperature. The benchmark iterates programming pages in a block and then erasing the block until the block wears out. Each page programming is followed by the verification of the page using the 48bit BCH ECC. If an uncorrectable bit error is detected, the benchmark terminates. After all pages in the block are programmed, each page is verified again to check silent data corruption due to the program disturbance. An uncorrectable bit error during the verification also finishes the benchmark. If no such error is occurred, the benchmark erases the block and restarts to program pages.

TABLE II. THE CONFIGURATIONS FOR EVALUATING FULL PAGE, PARTIAL PAGE, AND SUBPAGE PROGRAMMING

Configuration	Programming method	Programming unit (bytes)
$P_{\{SLC\|MLC\}}$	Full page	8192
PP_{SLC}	Partial page	4096
$SP2_{\{SLC\|MLC\}}$	Subpage (ping-pong)	4096
$SP2O_{\{SLC\|MLC\}}$	Subpage (pong-pong)	4096
$SP4_{\{SLC\|MLC\}}$	Subpage (ping-pong)	2048
$SP4O_{\{SLC\|MLC\}}$	Subpage (pong-pong)	2048
$PSP2_{MLC}$	Full page / Subpage (ping-pong)	8192 / 4096

The benchmark programs pages with specific patterns, which are devised to evenly distribute stress to the cells in the pages. During the first iteration of the test sequence, even-numbered pages in a block are programmed with a checkerboard pattern of `0x55AA` while odd-numbered pages are with a checkerboard pattern of `0xAA55`. The patterns are inverted on each block erase. Thus, on the next iteration, even-numbered pages are programmed with the pattern of `0xAA55` and odd-numbered pages with `0x55AA`.

Table II summarizes the configurations of page programming methods that we have considered. The abbreviations of P, PP, SP, and PSP represent full page programming, partial page programming, subpage programming in SP mode, and subpage programming in PSP mode, respectively. The number after SP and PSP denotes the number of subpages within a page. For subpage programming, both ping-pong ($SP2$ and $SP4$) and pong-pong ($SP2O$ and $SP4O$) orders are evaluated. When a block is used in PSP mode, the various ratios of N and M are used (cf. Section V-B). Note that the block in the PSP mode uses the ping-pong order only. Each test is performed on both SLC-mode and MLC-mode as denoted by the subscript $_{SLC}$ and $_{MLC}$ of the configuration if possible. Partial page programming (PP) is evaluated only on SLC-mode because it is not allowed in MLC-mode.

V. Evaluation Results

A. Enhancement in Flash Endurance Cycle

We collected the result of the flash endurance cycles from around 10 blocks per configuration. Throughout the measurement, we have also observed that the first uncorrectable bit error occurs only at specific page addresses as Cai et al. claimed [13].

Figure 8 shows the flash endurance cycles of SLC-mode blocks on various page programming configurations. Boxes represent the averages of the measured endurance cycles and error bars at the end of the boxes represent their standard deviations. The results from SP mode in the pong-pong order ($SP2O_{SLC}$ and $SP4O_{SLC}$) show the breakdown of the endurance cycle for each partition (P0, P1, P2, and P3). Note that the P/E cycles are normalized to that of the baseline configuration (P_{SLC}), which are roughly more than ten thousands regardless of manufacturers. All chips support the P_{SLC} configuration, i.e., operate correctly in using MLC blocks in SLC-mode. However, the chip C does not handle partial page program (PP_{SLC}) properly; subsequent partial page programming makes the previously programmed part of the page to the unprogrammed values, i.e., 0xFFFF. We find

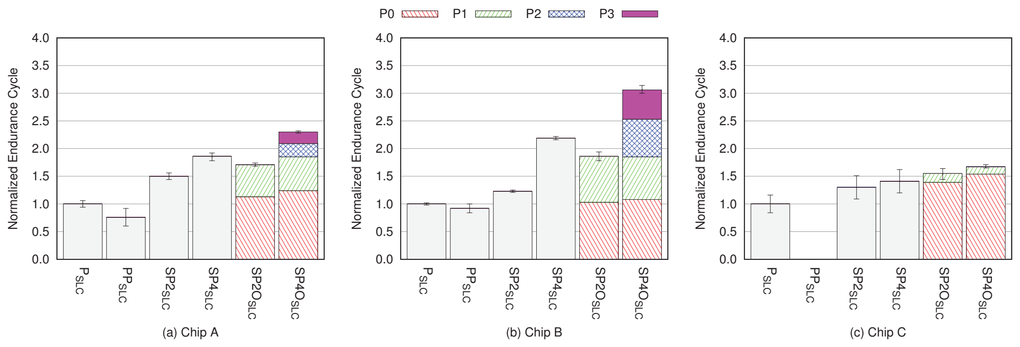


Fig. 8. Flash endurance cycles obtained from SLC-mode blocks

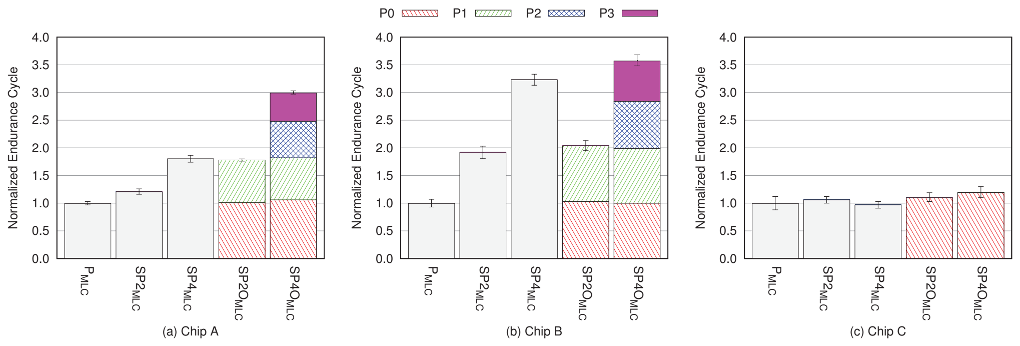


Fig. 9. Flash endurance cycles obtained from MLC-mode blocks

that PP_{SLC} shows the reduced P/E cycle than P_{SLC} by 24% and 8% for chip A and B, respectively. We believe that the decrease in the endurance cycle is originated from the increase in the intra-page program disturbance.

All SP mode configurations show better endurance cycles than the baseline P_{SLC} configuration, improved by up to 207%. This result implies that the proposed subpage programming can significantly improve the lifetime of flash memory in handling small writes. The configurations with a smaller subpage ($SP4_{SLC}$ and $SP4O_{SLC}$) always outperform the configurations with the larger subpage ($SP2_{SLC}$ and $SP2O_{SLC}$), regardless of the chip and the order in subpage programming. This is due to that a smaller subpage implies a less number of programmed cells per subpage program.

Under the same subpage size, the configurations in the pong-pong order ($SP2O_{SLC}$ and $SP4O_{SLC}$) always outperform the configurations in the ping-pong order ($SP2_{SLC}$ and $SP4_{SLC}$). In the ping-pong order, the endurance cycle is determined by the first uncorrectable bit error occurred from any partitions, even if the subpages in the other partitions can endure more P/E cycles. In contrast, the pong-pong order allows one uncorrectable bit error per each partition and utilizes such endurable subpages. The breakdown of the endurance cycle in the pong-pong order configurations supports the explanation; the lifetime of the first partition is comparable to that of P_{SLC}, but additional endurance cycles can be obtained from other partitions. In this sense, SP mode in the pong-pong order can maximize the endurance cycles.

Chip C exhibits different characteristics in comparison with other chips. Chip A and B show the improvement in the endurance cycles by up to 130% and 207%, respectively. However, chip C shows only 68% of improvement at best on $SP4O_{SLC}$ configuration. In addition, the subsequent partitions do not provide much improvement in the endurance cycle; the improvement comes mostly from the first partition. We assume the singularity is due to different cell organization or page programming mechanisms against other two chips.

Figure 9 shows the endurance cycles obtained from MLC-mode blocks. The endurance cycles are normalized to that of P_{MLC}, which are roughly more than three thousands for all chips. The results are consistent with the results collected from SLC-mode blocks. The configurations using SP mode show the extended endurance cycle, increased by up to 200% and 258% for chip A and B, respectively, on the $SP4O_{MLC}$ configuration. Chip C exhibits the unusual characteristics on the MLC-mode blocks as well, however, subpage programming does not hurt the endurance cycle of the chip.

B. *Influence of Subpage Programming*

In order to verify the effect of subpage programming on the flash endurance cycle, we measured the endurance cycle from various PSP modes. We vary the N:M combination of PSP mode so that the ratio of SP mode changes from 0% to 100%. When the block operates in SP mode, the block is partitioned into two partitions, which are used in the ping-pong order. Figure 10 shows the endurance cycle of PSP modes in various ratios of SP mode, normalized to that of full page programming, i.e., 0:1 PSP mode. Each point represents the average of the measured P/E cycle from around 10 blocks and the vertical bar represents their standard deviations.

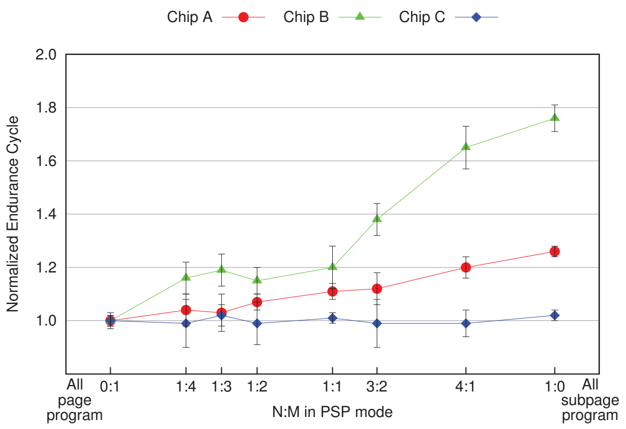


Fig. 10. Endurance cycles of MLC blocks on various $N{:}M$ PSP modes

The result shows the increase in endurance cycle is linearly proportional to the ratio of SP mode in chip A and B. The linear relationship suggests that the block's lifetime is not impaired even when the block switches full page programming and subpage programming repeatedly.

VI. Related Work

Many studies have suggested the way to enhance the lifetime of flash memory. The common idea behind these studies is to level the flash wear-out among blocks in a chip and pages in a block. Mothilal [14] suggests an algorithm to evenly level the program and erase cycles of blocks. Woo and Kim [15] propose to utilize programming latency as an additional indicator of per-block wear-out. Wear-unleveling [16] suggests to relieve error-prone pages from writing to prevent further wear-out from such critical pages. Dynamic Program and Erase Scaling (DPES) scheme [17] proposes a system-level approach to extend the lifetime at flash cell-level. DPES controls the programming and erase voltage dynamically to minimize the oxide degradation at low performance overhead.

VII. Conclusion & Future Work

The page size has been growing in order to maximize the throughput of flash memory. This trend, however, aggravates the reliability issue of flash memory by amplifying a small amount of write to an entire page programming. We propose a novel subpage programming scheme to deal with such distressing small writes. Subpage programming splits a page into subpages and programs only one of the subpages at a time. As the other subpages are inhibited from the programming, other cells in the subpages experience less oxide degradation and program disturbance. We evaluated the suggested scheme on three NAND flash memory chips from different manufacturers. Evaluation results show that subpage programming can increase the endurance cycle by up to 258%.

We leave the detailed analysis on what makes subpage programming more effective on some chips. We also plan to utilize the subpage programming scheme at the flash translation layer (FTL). We expect subpage programming can effectively deal with the small writes that are incurred in managing metadata and mapping information in FTL.

Acknowledgment

This work was supported by the IT R&D program of MKE/KEIT (No.10041244) and the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIP) (No. 2013R1A2A1A01016441).

References

[1] M. Abraham, "NAND flash architecture and specification trends," *Flash Memory Summit*, 2012.

[2] Open NAND Flash Interface, "Open NAND Flash Interface specification revision 1.0," 2006. [Online]. Available: http://www.onfi.org/specifications

[3] C. Kim, J. Ryu, T. Lee, H. Kim, J. Lim, J. Jeong, S. Seo, H. Jeon, B. Kim, I. Lee, D. Lee, P. Kwak, S. Cho, Y. Yim, C. Cho, W. Jeong, K. Park, J. Han, D. Song, K. Kyung, Y. Lim, and Y. Jun, "A 21nm high performance 64Gb MLC NAND flash memory with 400MB/s asynchronous toggle DDR interface," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 4, pp. 981–989, 2012.

[4] R. Fisher, "Optimizing NAND flash performance," *Flash Memory Summit*, 2008.

[5] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf, "Characterizing flash memory: anomalies, observations, and applications," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*, 2009, pp. 24–33.

[6] F. Masuoka, M. Momodomi, Y. Iwata, and R. Shirota, "New ultra high density EPROM and flash EEPROM with NAND structure cell," in *Proceedings of the 1987 International Electron Devices Meeting*, vol. 33, 1987, pp. 552–555.

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

[8] A. Modelli, A. Visconti, and R. Bez, "Advanced flash memory reliability," in *Proceedings of the International Conference on Integrated Circuit Design and Technology 2004. (ICICDT '04)*, 2004, pp. 211–218.

[9] Y.-B. Park and D. K. Schroder, "Degradation of thin tunnel gate oxide under constant Fowler-Nordheim current stress for a flash EEPROM," *IEEE Transactions on Electron Devices*, vol. 45, no. 6, pp. 1361–1368, 1998.

[10] K.-D. Suh, B.-H. Suh, Y.-H. Um, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, J.-H. Choi, J.-R. Kim, and H.-K. Lim, "A 3.3V 32Mb NAND flash memory with incremental step pulse programming scheme," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.

[11] Y. Li, Y. K. Fong, and T. Miwa, "Non-volatile memory and control with improved partial page program capability," US Patent 7 453 735, November, 2008.

[12] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, pp. 68–79, 1960.

[13] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE '12)*, 2012, pp. 521–526.

[14] A. R. Mothilal, "Static wear leveling," US Patent 8 065 469, November, 2011.

[15] Y. Woo and J. Kim, "Diversifying wear index for MLC NAND flash memory to extend the lifetime of SSDs," in *Proceedings of the 11th ACM International Conference on Embedded Software (EMSOFT '13)*, 2013.

[16] X. Jimenez, D. Novo, and P. Ienne, "Wear unleveling: improving NAND flash lifetime by balancing page endurance," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST '14)*, 2014, pp. 47–59.

[17] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Lifetime improvement of NAND flash-based storage systems using dynamic program and erase scaling," in *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST '14)*, 2014, pp. 61–74.