NAND Flash Memory With Multiple Page Sizes for High-Performance Storage Devices

Jin-Young Kim, Sang-Hoon Park, Hyeokjun Seo, Ki-Whan Song, Sungroh Yoon, and Eui-Young Chung

***Abstract*— In recent years, the demand for NAND flash-based storage devices has rapidly increased because of the popularization of various portable devices. NAND flash memory (NFM) offers many advantages, such as nonvolatility, high performance, the small form factor, and low-power consumption, while achieving high chip integration with a specialized architecture for bulk data access. A unit of NFM's read and program operations, the page, has continuously grown. Although increasing page size reduces costs, it adversely affects performance because of the resultant side effects, such as fragmentation and wasted space, caused by the incongruity of data and page sizes. To address this issue, we propose a multiple-page-size NFM architecture and its management. Our method dramatically improves write performance through adopting multiple page sizes without requiring additional area overhead or manufacturing processes. Based on the experimental results, the proposed NFM improves write latency and NFM lifetime by up to 65% and 62%, respectively, compared with the single-page-size NFM.**

***Index Terms*— Multiple page sizes, NAND flash memory (NFM).**

I. Introduction

With the recent popularity of various portable devices, the demand for compact and reliable NAND flash-based storage devices (NFSDs) has dramatically increased. NAND flash memory (NFM) offers many advantages, such as nonvolatility, high performance, the small form factor, and low-power consumption, while rapidly improving capacity and cost by downscaling the process technology to 21 nm [1] and providing a 3-D NFM structure.

NFM's higher degree of chip integration compared with other types such as NOR flash memory [2] is achieved by its specialized architecture for bulk data access. In this architecture, the page—the unit for read and program operations—includes numerous memory cells. The unit for erase operations, the block, is composed of tens of pages. Page size has continuously increased since early NFMs, when a single page was smaller than 1 kB.

Increasing the page size enlarges the portion of cell areas in a die and reduces the number of NFM operations that service a given amount of data. Thus, the larger page reduces the cost-per-bit of NFMs and improves the throughput of NFSDs. However, the larger page cannot guarantee better performance in all cases. In particular, the larger page causes higher fragmentation within NFMs, which causes inefficient utilization of NFM space [called false capacity (FC)] and increases the number of garbage collections (GCs) [3]. Moreover, when requests with a small amount of data are given, NFM write performance deteriorates because of frequent read-and-modify procedures. Bang *et al.* [4] analyzed these large-page problems in detail.

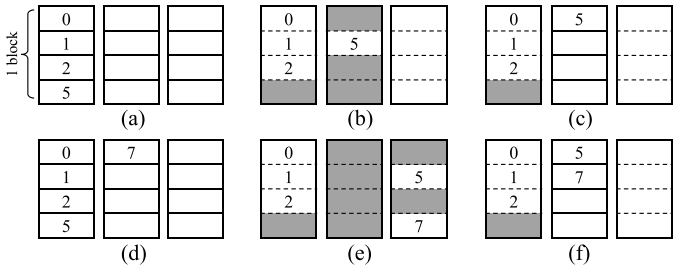

Fig. 1. Page status of (a) and (d) SPNFM with small pages, (b) and (e) SPNFM with large pages, and (c) and (f) MPNFM when four sectors are written and given two example requests are handled by NFMs, respectively.

Despite the limitations, increasing the page size is inevitable because of its cost reduction and the performance bottlenecks caused by large requests. We, therefore, propose an NFM architecture that offers the advantages of a small page without sacrificing the benefits of a large one. Although Wang *et al.* [3] discussed the necessity of such an architecture, they did not propose specific and realistic solutions for the implementation.

The contributions of this brief can be summarized as follows. First, we propose a multiple-page-size NFM (MPNFM) architecture.[1] These multiple page sizes occur within a die without affecting the area and manufacturing process. In addition, the implementation is simple, because only some logics are shifted to generate the differential page sizes.

Second, a management method for effectively utilizing NFSDs equipped with the MPNFM is proposed. The related algorithm cooperates with any existing flash translation layer (FTL), the software layer that manages NFM. The proposed method retains the sophisticated features of the existing FTLs and adds certain modules to maximize the MPNFM effects. According to our experiments, the combination of MPNFM and the management algorithm greatly improves write performance compared with single-page-size NFM (SPNFM).

The remainder of this paper is organized as follows. In Section II, we present the background and motivation. In Sections III and IV, we respectively introduce an MPNFM device and its management method. We present our experimental results in Section V and our conclusions in Section VI.

II. Background and Motivation

A. Inefficiency of Large Pages of NFM

To elucidate the advantages of MPNFM, we use a simple example with three different NFMs, as shown in Fig. 1: 1) SPNFM with small pages (SPNFM-S); 2) SPNFM with large pages (SPNFM-L); and

[1] In this brief, we consider only the MPNFM with two different page sizes; nevertheless, the architecture and management algorithm can be applied to more general cases.

Manuscript received September 28, 2014; revised January 12, 2015; accepted February 13, 2015. Date of publication March 24, 2015; date of current version January 19, 2016. This work was supported by the National Research Foundation of Korea under Grant 2013R1A1A2011208 through the Ministry of Education, and under Grant 20110009963 through the Ministry of Science, ICT and Future Planning, and by Samsung Electronics Company, Ltd., Suwon, Korea.

J.-Y. Kim is with Samsung Electronics Company, Ltd., Suwon 443-742, Korea, and also with Yonsei University, Seoul 120-749, Korea (e-mail: jy0615.kim@samsung.com).

S.-H. Park, H. Seo, and E.-Y. Chung are with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea (e-mail: soskhong@dtl.yonsei.ac.kr; jjsky7@dtl.yonsei.ac.kr; eychung@yonsei.ac.kr).

K.-W. Song is with Samsung Electronics Company, Ltd., Suwon 443-742, Korea (e-mail: kiwhan.song@samsung.com).

S. Yoon is with the Department of Electrical and Computer Engineering, Seoul National University, Seoul 151-744, Korea (e-mail: sryoon@snu.ac.kr).

Digital Object Identifier 10.1109/TVLSI.2015.2409055

TABLE I
REQUIRED NFM COMMANDS FOR REQUESTS

	Request 1 : Write, Address 7, 1 sector	**Request 2** : Read, Address 0, 3 sector
Command	{program, read}	{program, read}
SPNFM-S	{1, 0}	{0, 3}
SPNFM-L	{1, 1}	{0, 1}
MPNFM	{1, 0}	{0, 1}

3) MPNFM. The smallest represents a sector (512 B); the block size of all NFMs is 2 kB.

The numbers of pages and page sizes among the three NFMs differ. One block of SPNFM-S includes four small pages, and a page consists of one sector, whereas one block of SPNFM-L includes one large page comprised of four sectors. In Fig. 1, the gray areas represent unusable portions because of the impossibility of in-place updates. The number of gray areas is proportional to the amount of FC.

We define a request as {type, logical address, data length} and assume that two requests {write, address 0, 3 sectors} and {write, address 5, 1 sector} have been sequentially executed in the three NFMs. Fig. 1(a)–(c) shows the page statuses of each NFM after handling two requests; the number in each box denotes the logical address of the stored sector. The three NFMs will receive the two new requests; the results of these requests will demonstrate why MPNFM may be deemed superior to the other memory types.

We now assume that the two new requests are issued as shown in Table I. For Request 1, SPNFM-S and MPNFM require only one NFM program, whereas SPNFM-L requires an additional NFM read for read-and-modify operations. (When a page needs to be modified, NFM can write new data only after moving some old data in the page into a new page in order to preserve the old data not changed by the new request. This is called read-and-modify [3], [4].) This means that the NFM that utilizes smaller pages is more suitable for small writes. Cumulative read-and-modify procedures can significantly degrade write performance.

Request 2 is a sequential read request that fetches three sectors. When Request 2 is given, SPNFM-L and MPNFM, which include large pages, outperform SPNFM-S, which has only small pages, because of the higher throughput caused by the large pages. This difference is additionally demonstrated by sequential write requests. A small page typically issues more program commands than larger pages; therefore, its throughput is lower than that of larger pages.

In addition, the page size difference shows significant effects after the two requests are serviced. As described above, the gray boxes in Fig. 1 denote portions of pages that cannot be written without an erase operation (i.e., FC). Large FC, which is caused by fragmentation, increases the frequency of GC. After two requests, the FC values of SPNFM-S, SPNFM-L, and MPNFM are 0, 7, and 1 sector, respectively, as shown in Fig. 1(d)–(f). In other words, SPNFM-L is the first NFM type requiring GC because of its low efficiency in utilizing pages.

B. Observation of Workload Characteristics

The aforementioned advantages of MPNFM are effective only if the workloads given to NFMs have both sequential and random requests sufficient for contributing to NFSD performance. Using the same approach as in [3], we analyzed I/O characteristics of several workloads; Fig. 2 shows that the request size in various workloads had two different aspects.

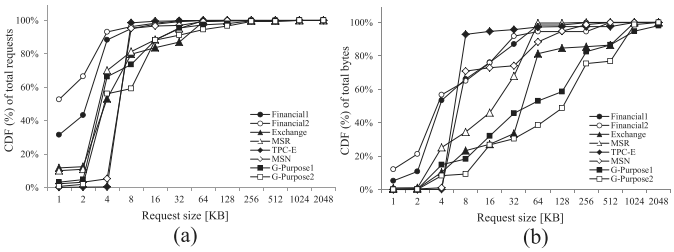

Fig. 2. Cumulative distribution functions of (a) request count and (b) I/O contribution according to request's size.

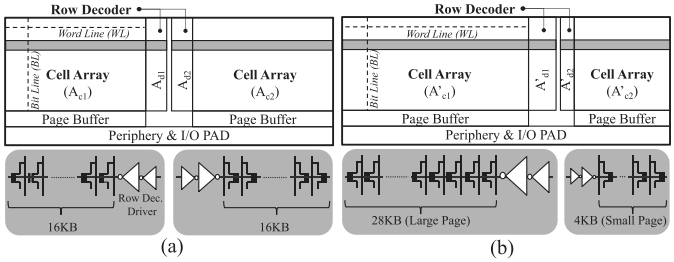

Fig. 3. Architectures of (a) SPNFM and (b) MPNFM.

Fig. 2(a) shows that 89% of requests were smaller than 8 kB. These requests were sensitive to latency of NFMs; therefore, the reduction of read-and-modify operations and fragmentation using small pages was effective. Moreover, the higher throughput of larger pages cannot be ignored because sequential requests significantly contributed to the total I/O (48% on average), as shown in Fig. 2(b). In summary, consideration of request size and MPNFM advantages over SPNFM motivate us to propose MPNFM and its management method.

III. NFM WITH MULTIPLE PAGE SIZES

A. Design Concept and Cost

An NFM consists of two parts: 1) the core and 2) the peripheral. The peripheral consists of numerous logic and analog circuits to assist the core, which includes cell arrays, a row/column decoder, and page buffers. A page is defined as the cells activated by one word line within a cell array; therefore, SPNFM, as shown in Fig. 3(a), includes identical pages across multiple symmetric cell arrays.

The proposed MPNFM as shown in Fig. 3(b), on the other hand, has asymmetric cell arrays, which result in different page sizes for different cell arrays. In fact, the difference between MPNFM and SPNFM is not significant with respect to the architecture, except for the multiple page sizes. Hence, the design and fabrication of MPNFM do not cause obvious changes in terms of cost and the manufacturing process.

For instance, assume SPNFM with a 16 kB page and MPNFM with 28-/4 kB pages. When the area of each cell array and row decoder are defined as A_{cx} and A_{dx}, respectively, the total respective areas of SPNFM and MPNFM can be computed as follows:

$$A_{c1} + A_{c2} = A'_{c1} + A'_{c2}, \quad A_{d1} + A_{d2} \approx A'_{d1} + A'_{d2}. \tag{1}$$

The total area of cell arrays for the two NFMs is identical, because the total number of cells connected to the left and right word lines and the bit line are the same; the only difference between them is the size of the row decoders. The row decoder for a 28 kB page must adopt a larger driver size, but the increased driver size can be compensated by the reduction of the driver size for the 4 kB page.

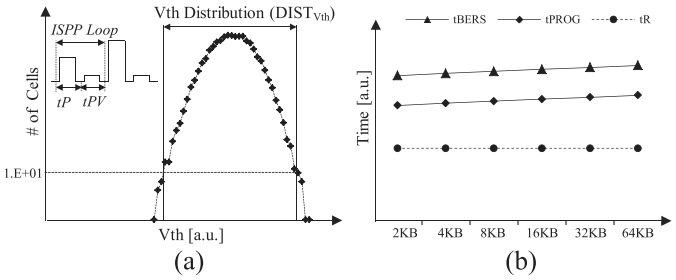

Fig. 4. (a) $\mathrm{DIST}_{V\mathrm{th}}$ measured in NFM of 21 nm CMOS technology. (b) Operation times according to page size.

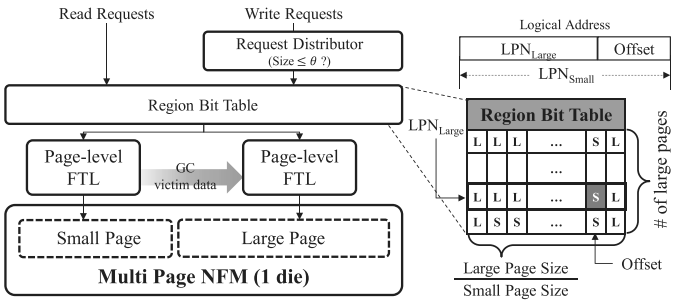

Fig. 5. Overview of FTL for MPNFM.

Thus, the total area of row decoders becomes approximately similar.

In addition, the cell structure and operation conditions of MPNFM are the same as that of SPNFM; only the number of cells connected to a word line differs. This helps MPNFM not to incur an extra burden in terms of the physical problem, such as read disturb [5] as well as the manufacturing process.

B. Operation Time in Multiple Page Sizes

The various page sizes for MPNFM affect operation time. First, time to read a page of NFM (tR) is as follows [6], [7]:

$$tR = (t\mathrm{PRE} + t\mathrm{SENS} + t\mathrm{DISCH}) \propto \mathrm{RC}_{\mathrm{BL}} \tag{2}$$

where tPRE, tSENS, and tDISCH, respectively, denote time to precharge, sense, and discharge for a read command, while $\mathrm{RC}_{\mathrm{BL}}$ represents parasitic characteristics of a bit line. The three timing parameters are not affected by differential page size, because they depend on $\mathrm{RC}_{\mathrm{BL}}$ instead of the word line, which is actually affected by differential page size.

Time to program a page (tPROG) is as follows [1], [6]:

$$t\mathrm{PROG} = (tP + tPV) \times \#\ \mathrm{ISPP\ \ loop} \propto \mathrm{DIST}_{V\mathrm{th}} \tag{3}$$

where tP and tPV are the time to program and verify cells, respectively. # ISPP loop and $\mathrm{DIST}_{V\mathrm{th}}$ are the number of program and verify operations in the incremental step pulse programming/erasing scheme (ISPP), which is typically used by most of modern NFMs [1], and the distribution of the NFM cell threshold voltage (Vth) after one program pulse is given to the cells, respectively, as shown in Fig. 4(a). In (3), increasing page size leads to larger $\mathrm{DIST}_{V\mathrm{th}}$ due to increase of the number of cells connected to word line, this is followed by the increase of # ISPP loop, and ultimately increases tPROG.

To quantify the variation of write performance according to page size, we measured $\mathrm{DIST}_{V\mathrm{th}}$ in an NFM of 21 nm CMOS technology and analyzed the correlation between $\mathrm{DIST}_{V\mathrm{th}}$ and page size. The results are shown in Fig. 4. We predicted that tPROG is increased by $\sim$10% when the page size was doubled (the NFM used for our evaluation was the same as the proposed device in [1]).

Time to erase a block (tBERS) can be computed as [6], [8]

$$t\mathrm{BERS} = (tE + tEV) \propto A_{\mathrm{well}} \tag{4}$$

where tE, tEV, and A_{well} denote the time to erase the block, the time to verify it, and the area of the matrix insulated p-well for all blocks, respectively. A_{well} is proportional to the page size; accordingly, a larger page requires a longer tBERS when the number of pages per block is identical. Considering the proportion affected by A_{well} in erase pulses [8], the amount of variation in tBERS is similar to that of tPROG, which is roughly 10% for doubled pages according to our measurement.

In conclusion, a large page in MPNFM is worse than a small page in terms of tPROG and tBERS, as shown in Fig. 4(b); we design the specification of MPNFM focusing the large page, because the portion of the large pages is dominant in overall storage space. To effectively utilize multiple page sizes, MPNFM may require additional timing control circuits. However, the area overhead incurred by circuits (e.g., some delay chains and flip-flop logics) is not obvious; therefore, a detailed analysis of them is omitted because of space constraints.

IV. Flash Translation Layer for MPNFM

A. Overview

To support differential pages within MPNFM, we exploit existing FTL with minimal modification. Fig. 5 shows the overall FTL diagram for MPNFM. It includes two independent page-level FTLs that can be employed to leverage an attractive features in modern NFSDs. Even different FTLs can be used for each page without modification, except on the GC part of the small-page FTL. Further, we add two preprocess modules—a request distributor and a region bit table—to efficiently utilize MPNFM. The request distributor classifies incoming write requests based on their lengths, whereas the region bit table is a small table that indicates requested data locations between small and large pages by one bit.

B. Address Translation

The request distributor and region bit table are modules relating to address translation. To classify write requests, the request distributor forwards write requests smaller than θ to the small-page FTL, where θ is the size of one small page. By forwarding small writes to small pages, large-page fragmentation incurred by small writes can be reduced.

By checking one bit of the region bit table, a request can find the requested data location. The region bit table stores the bit sequence; each bit indicates a large (L) or small (S) page. The table is indexed by the disassembled addresses of the given request's logical address, as shown in Fig. 5. ($\mathrm{LPN}_{\mathrm{Large}}$/$\mathrm{LPN}_{\mathrm{Small}}$) is the logical page address when a logical page includes the group of adjacent data whose size is the same as the (large page size/small page size) of MPNFM. The offset is the relative location of one small page within a large page. In summary, the given request obtains its destination page and location of the requested data if the data have been already written. Then, the FTL of the destination page handles the request and accesses NFM.

C. Garbage Collection

GC of each page type is separately performed, because there are two independent FTLs for each page type. However, owing to the limited capacity of small pages and migration overhead, it is better for data to be eventually moved to large pages; when the data are moved from large to small pages, more program operations must be executed.

TABLE II
EXPERIMENTAL WORKLOAD INFORMATION

Name	Number of Requests	Write Ratio (%)	Average W-Request Length (sectors)	Random Write Ratio (%)
Financial1	5334949	77	9.5	87
Financial2	3699194	18	7.9	88
Exchange	204571	79	15.2	67
MSR	1211035	88	25.8	72
TPC-E	1147310	10	25.6	4
MSN	1081878	30	22.3	18
G-Purpose1	725123	74	30.0	74
G-Purpose2	2309480	5	431.2	18

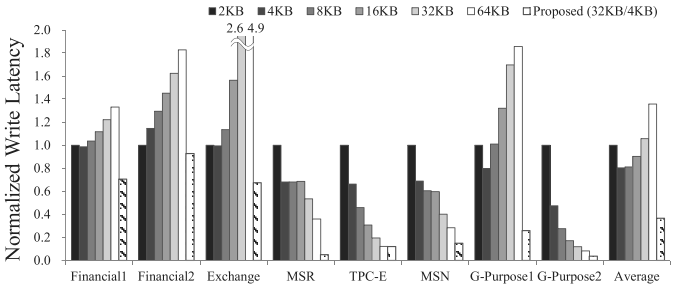

Fig. 6. Normalized average write latency.

We modify the destination of the moved data in an existing GC method and add a new table to record the data transfer. In particular, when the FTL of the small page activates GC, the valid pages of the victim blocks are written to the large pages instead of being written back to another small page, as shown in Fig. 5. Furthermore, other data with the same LPN_{Large} and stored in small pages are moved to the large page along with the data stored in the GC victim block. The GC operation is complete when all data movements have been recorded to the region bit table and the corresponding bits are flipped [15].

The GC modification is intended to maximize the effects of MPNFM. That is, the modified GC reduces fragmentation by transforming small write requests into large ones that are compatible with large pages. Moreover, the modification on the FTL does not require significant design efforts, because only reads of adjacent data are to be added and small-page programs are to be substituted for large ones.

V. EXPERIMENTS

A. Experimental Setup

To evaluate MPNFM and its management scheme, we implemented a trace-driven simulator [9], [10], which included the FTL and NFM as specified in [11]. Various workloads summarized in Table II were collected from [12] and [13], and a common PC using DiskMon [14]. The simulator includes SPNFM with page sizes ranged from 2 to 64 kB and an ordinary page mapping FTL as the control group, and MPNFM with two page sizes (32 and 4 kB) and the page mapping FTL, including the proposed management algorithm. The capacity of all NFMs was set to 4 GB for fair comparison. Finally, the NFM operation times were applied on the basis of NFM specifications from [11] and the variation rate of operation times according to the page size mentioned in Section III-B.

B. Experimental Results

1) Average Write Latency: The write latency is a critical NFSD metric that many researchers have targeted. As listed in Fig. 6, all values were normalized to the write latency of an SPNFM with 2 kB page. As reported in [4], the optimal SPNFM page size varies according to the workload. Here, workloads with many small write requests (Financial1, Financial2, Exchange, and G-Purpose1) benefited from a small-page SPNFM, while others worked more effectively with a larger page SPNFM.

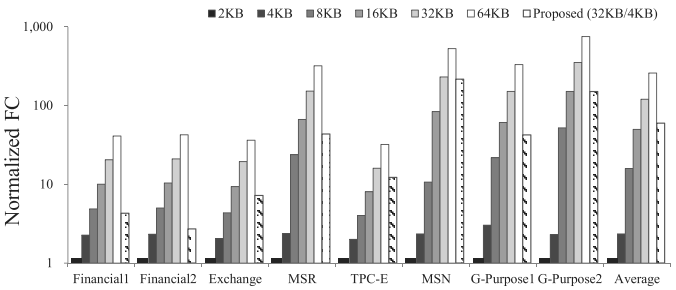

Fig. 7. Normalized FC.

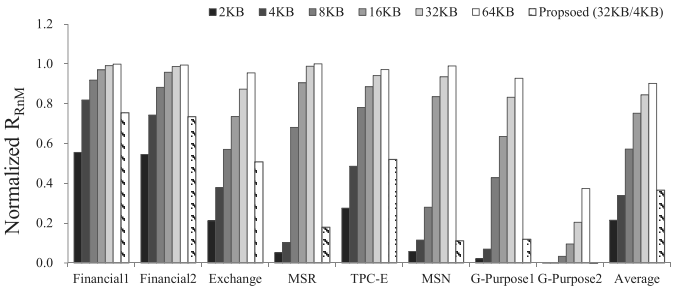

Fig. 8. Normalized read-and-modify ratio.

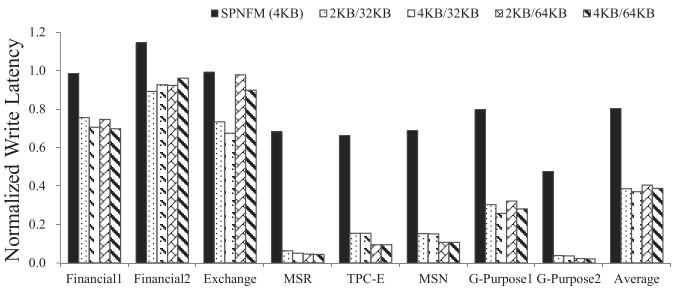

Fig. 9. Comparison of write latencies according to page configurations in the proposed method.

The proposed method showed the smallest write latency regardless of workload or optimal page size. Its write latency was reduced by 63%, 54%, and 73% compared with SPNFM with a 2-, 4-, and 64 kB page, respectively. For further analysis, we measured FC and the read-and-modify ratio (R_{RnM}), as shown in Figs. 7 and 8. They are defined as $FC = S_{\text{uns}}$ and $R_{RnM} = P_{RnM}/P_w$, where S_{uns}, P_w, and P_{RnM} refer to the numbers of total unusable sectors within a page, the total number of written pages, and the total number of read-and-modify pages, respectively [4].

The proposed method reduced the amount of FC, particularly for workloads with small writes (i.e., Financial1 and Financial2), as shown in Fig. 7. The reduced FC was the result of reduced fragmentation and leaded to fewer GCs. Moreover, the proposed method improved R_{RnM}, as shown in Fig. 8; the values were similar to that of the 4 kB SPNFM despite the limited capacity of the 4 kB pages of MPNFM. This means that data requested from the host were well matched with different page sizes of MPNFM. In fact, the proposed method was not always better than SPNFM in terms of FC and R_{RnM}; however, at least one of its two metrics

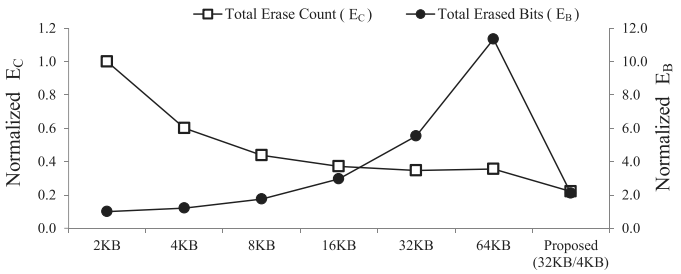

Fig. 10. Normalized total erase count (E_C) and total erased bits (E_B).

was generally lower than SPNFM's average, thereby resulting in a marked performance improvement, as shown in Fig. 6.

Results on the read latency of the proposed method were omitted from the experiment because they were similar to the read latency result of SPNFM. This is because read latency only relates to the number of required NFM read operations for the given request, unlike write requests that require read-and-modify operations and GCs [15].

2) Optimization of MPNFM: To determine the optimal combination of MPNFM page sizes under our workloads, various page combinations were conducted, as shown in Fig. 9. MPNFM outperformed SPNFM regardless of its configuration, which again proved the effectiveness of MPNFM. Moreover, MPNFMs with 4 kB page were slightly better than those with 2 kB page, because the latter required more program commands for the same amount of data than did the 4 kB page.

3) Life-Time of NFMs: In general, the total number of erased counts (E_C) in NFM is inversely proportional to the life-time. However, in our experiment, it is not enough for measuring the life-time, since the number of erased bits for each page size is quite different. For this reason, we measured the total number of erased bits (E_B) in addition to E_C. Simply speaking, the best NFM architecture in terms of life-time will show the least E_B as well as E_C. In NFMs with the same capacity, the NFMs with smaller pages show higher E_C due to smaller block size, while the NFMs with larger pages show higher E_B due to more FC. Surprisingly, it is observed that the proposed MPNFM is comparable with the best cases in both metrics—E_C and E_B. It means that our page management method allocates appropriate size of pages to the given requests, which results in the reduced E_B and E_C of Fig. 10 (more details are provided in [15]).

VI. Conclusion

In this brief, we have proposed an MPNFM and a management method for the NFM. The proposed NFM includes different sizes of pages without affecting the area and the manufacturing process, and the pages reduce fragmentation and wasted spaces occurred by the incongruity between data and page sizes. In experimental results, our method improves write latency and the life-time of NFM by up to 65% and 62%, respectively, compared with a 32 kB SPNFM.

As a future work, we will develop a wear-leveling algorithm to resolve unbalanced wear-outs between large-page and small-page size blocks. We will also focus on power-efficient MPNFM architecture for large-scale storage systems.

References

[1] Y. S. Cho *et al.*, "Adaptive multi-pulse program scheme based on tunneling speed classification for next generation multi-bit/cell NAND flash," *IEEE J. Solid-State Circuits*, vol. 48, no. 4, pp. 948–959, Apr. 2013.

[2] J. Cooke, "NAND flash 101: An introduction to NAND flash and how to design it into your next product," in *Proc. Embedded Syst. Conf.*, 2006.

[3] D. Wang, A. Sivasubramaniam, and B. Urgaonkar, "A case for heterogeneous flash in the datacenter," in *Proc. IEEE 33rd Int. Distrib. Comput. Syst. Workshops*, Jul. 2013, pp. 220–225.

[4] K. Bang, D.-G. Kim, S.-H. Park, E.-Y. Chung, and H.-J. Lee, "Application-aware design parameter exploration of NAND flash memory," *J. Semicond. Technol. Sci.*, vol. 13, no. 4, pp. 291–302, 2013.

[5] J. Cooke, "The inconvenient truths about NAND flash memory," in *Proc. Micron MEMCON*, 2007, pp. 19–23.

[6] 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, Nov. 1995.

[7] L. Crippa *et al.*, "Sensing circuits," in *Inside NAND Flash Flash Memories*. Amsterdam, the Netherlands: Springer-Verlag, 2010, pp. 197–233.

[8] R. Micheloni *et al.*, "High voltage overview," in *Inside NAND Flash Memories*. Amsterdam, the Netherlands: Springer-Verlag, 2010, pp. 329–351.

[9] S.-H. Park, S.-H. Ha, K. Bang, and E.-Y. Chung, "Design and analysis of flash translation layers for multi-channel NAND flash-based storage devices," *IEEE Trans. Consum. Electron.*, vol. 55, no. 3, pp. 1392–1400, Aug. 2009.

[10] S.-H. Park, D.-G. Kim, K. Bang, H.-J. Lee, S. Yoo, and E.-Y. Chung, "An adaptive idle-time exploiting method for low latency NAND flash-based storage devices," *IEEE Trans. Comput.*, vol. 63, no. 5, pp. 1085–1096, May 2014.

[11] *NAND Flash Memory Datasheet*, Micron Technology, Boise, ID, USA, 2009.

[12] (2007). *UMass Trace Repository*. [Online]. Available: http://traces.cs.umass.edu/

[13] (1997). *Storage Networking Industry Association*. [Online]. Available: http://iotta.snia.org/

[14] M. Russinovich. (2006). *DiskMon for Windows V2.01*. [Online]. Available: http://technet.microsoft.com/enus/sysinternals/bb896646.aspx

[15] J.-Y. Kim, S.-H. Park, H. Seo, K.-W. Song, S. Yoon and E.-Y. Chung. (2015). [Online]. Available: http://dtl.yonsei.ac.kr/down/MPNFM.pdf