3D-FlashMap: A Physical-Location-Aware Block Mapping Strategy for 3D NAND Flash Memory

Yi Wang[1], Luis Angel D. Bathen[2], Zili Shao[1], and Nikil D. Dutt[2]

[1]Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
{csywang, cszlshao}@comp.polyu.edu.hk

[2]Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697, USA
{lbathen, dutt}@uci.edu

***Abstract*—Three-dimensional (3D) flash memory is emerging to fulfil the ever-increasing demands of storage capacity. In 3D NAND flash memory, multiple layers are stacked to increase bit density and reduce bit cost of flash memory. However, the physical architecture of 3D flash memory leads to a higher probability of disturbance to adjacent physical pages and greatly increases bit error rates. This paper presents *3D-FlashMap*, a novel physical-location-aware block mapping strategy for three-dimensional NAND flash memory. 3D-FlashMap permutes the physical mapping of blocks and maximizes the distance between consecutively logical blocks, which can significantly reduce the disturbance to adjacent physical pages and effectively enhance the reliability. We apply 3D-FlashMap to a representative flash storage system. Experimental results show that the proposed scheme can reduce uncorrectable page errors by 85% with less than 2% space overhead in comparison with the baseline scheme.**

I. Introduction

NAND flash memory is facing fundamental scaling limitations beyond 20nm, and conventional cost-reduction approaches will be unable to maintain the current trend of increasing bit density and reducing flash memory bit cost [1]. To maintain the current pace of cost reduction, major NAND flash chip manufacturers are investigating three-dimensional flash memory to stack memory cells on a single wafer [2]. However, the increasing density of 3D flash causes severe reliability problems [3]. Previous studies have shown that most multi-level-cell (MLC) flash chips experience high bit error rates due to program disturb and read disturb [4]. A program or read operation can cause the disturbance of adjacent pages and change the state of the cell. This reliability problem is expected to worsen for three-dimensional flash memory.

Figure 1 illustrates Bit Cost Scalable (BiCS), a typical three-dimensional NAND flash architecture proposed by TOSHIBA [2]. In BiCS, vertical NAND strings punch through multiple stacks (plates) of the gate electrode. A stair-like interconnection, control gates, and bit lines form three-dimensional flash memory to select NAND cells. For the sake of illustration, in Figure 1(b), we extract three adjacent stacks of BiCS, and each stack consists of three adjacent physical pages. A program operation to a physical page (e.g., Page C) will not only influence adjacent pages in the same physical block (Page A and Page E), but also influence adjacent pages within the same control gate stack (Page B and Page D). This phenomena poses a threat to the integrity of data stored in 3D flash memory. To avoid and reduce the effect of disturbance, we propose a physical-location-aware strategy to provide a more reliable NAND flash memory storage system.

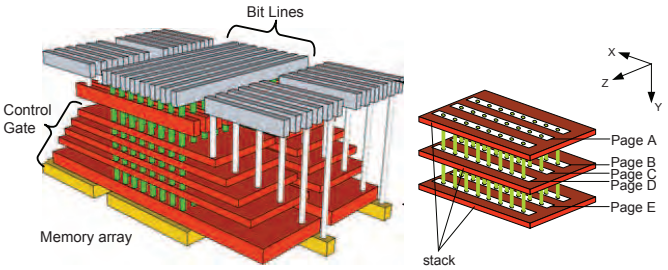

Fig. 1: (a) The architecture of BiCS [2] three-dimensional NAND flash memory. (b) The program disturb will not only influence the adjacent physical pages in the same physical block, but also influence the adjacent physical pages within the same control gate stack.

There have been approaches to provide reliable two-dimensional flash memory storage systems. Reliability can be enhanced through the modification of different components in NAND flash architecture, i.e., file system [5], [6], hardware implementation of NAND flash memory chip, or an intermediate software module called flash translation layer (FTL) [7], [8]. These approaches can provide good solutions to enhance the reliability of flash memory. Nevertheless, they are based on two-dimensional NAND flash memory, which could not be directly applied to three-dimensional flash memory. Our technique is a good supplement for these approaches by helping them effectively reduce bit errors caused by program disturb and read disturb to further improve the reliability of three-dimensional flash memory.

Recently, several 3D architectures for flash memory have been proposed, including BiCS [2], P-BiCS (Pipe-shaped Bit Cost Scalable) [9], TCAT (Terabit Cell Array Transistor) [10], VSAT (Vertical Stacked Array Transistor) [11], and VG (Vertical Gate) [12]. These 3D NAND flash memory architectures can provide the scalable bit cost by fabricating more and more vertical strings in the Z-axis direction (as shown in Figure 1). Therefore, they can provide ultra low cost flash chips compared with conventional designs for two-dimensional NAND flash. Although recent literature has focused primarily on the system structure and hardware implementation of three-dimensional flash, they make no specific attempt to cope with the reliability problem caused by the disturbance to adjacent

978-3-9810801-8-6/DATE12/©2012 EDAA

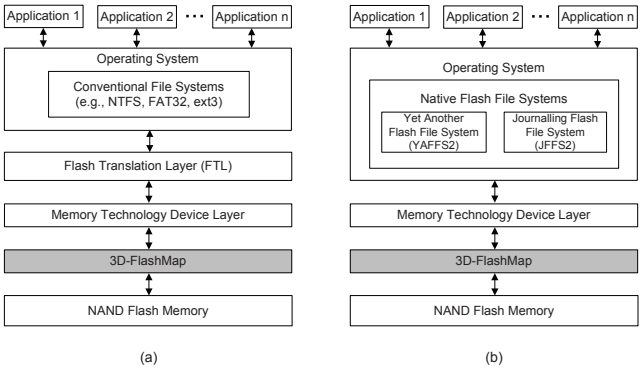

Fig. 2: (a) FTL-based NAND flash memory storage system. (b) Flash-file-system-based NAND flash memory storage system.

physical pages in three-dimensional flash.

This paper presents *3D-FlashMap*, a physical-location-aware block allocation strategy for three-dimensional NAND flash memory. 3D-FlashMap creates a physical block map to minimize the effect of program disturb and read disturb to adjacent physical locations. 3D-FlashMap transparently handles the requests from the Memory Technology Device (MTD) layer, and it is a general strategy that can be applied to both FTL-based (Figure 2(a)) and flash-file-system-based (Figure 2(b)) flash memory storage system to enhance the reliability of three-dimensional flash. *To the best of our knowledge, this is the first work that adopts a physical-location-aware strategy to enhance the reliability of three-dimensional flash memory.*

We implement 3D-FlashMap in the Linux kernel and evaluate 3D-FlashMap using a variety of I/O traces. We apply 3D-FlashMap to a representative FTL design MNFTL [13] for MLC flash memory. We use the number of uncorrectable page errors as a performance metric to evaluate the reliability of 3D-FlashMap. Experimental results show that our approach can reduce the number of uncorrectable page errors by 85% compared with the baseline scheme. In terms of the space overhead for the total number of block erase counts, our approach introduces less than 2% space overhead.

The rest of this paper is organized as follows. Section II discusses the background and presents the motivation of the paper. Section III presents our proposed 3D-FlashMap in detail. Section IV presents experimental results on the reliability enhancement and space overheads. Finally, in Section V, we conclude the paper and describe future work.

II. Background and Motivation

Unlike DRAM or SRAM, where every memory cell is addressed individually through intersecting row and column select lines (word lines and bit lines, respectively), NAND flash memory uses a "daisy chain" organization (Figure 3(a)), where 64 memory cells are connected to a source line and a bit line via select transistors on each end. To access a memory cell in the chain, all 64 cells in the chain have to be turned "On", resulting in the current flowing through all 64 cells into the bit line.

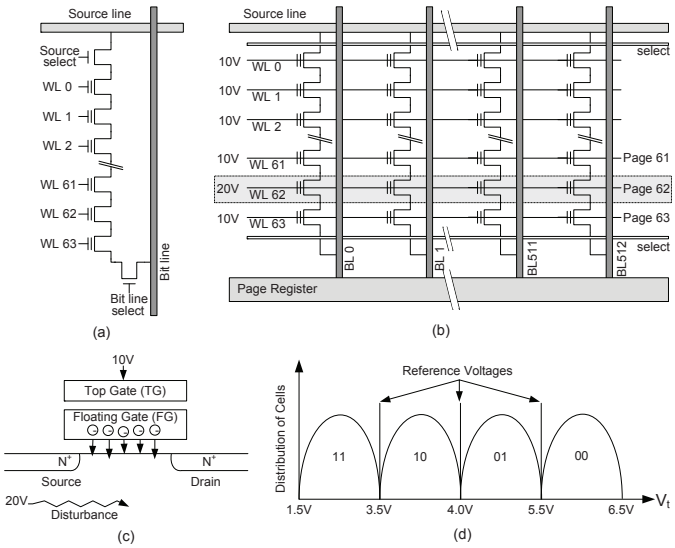

Fig. 3: (a) The "daisy chain" organization of NAND flash memory cells. (b) The program operation of a page (Page 62) and the program disturb to adjacent pages. (c) The disturbance of adjacent memory cells to let the loss of charge of the floating gate. (d) MLC flash uses reference points to determine the state of the cell.

A NAND flash memory chip consists of multiple blocks, and each block is composed of a fixed number of pages. A block is the basic unit for erase operations, while a page is the minimum unit for read/program operations. The memory cells connected to the same word line (WL) form a page or subpage. A physical block is shown in Figure 3(b), and this block contains 64 pages. The number of bit lines connected to a word line is 512 or 1024. In Figure 3(b), 512 bit lines are connected to a word line.

Both program operation and read operation to a physical page of flash memory will cause the disturbance to adjacent pages. Figure 3(b) illustrates the program operation of a physical page (Page 62). The program operation will pull up electrons from the gates through tunnel oxide into the floating gate, thereby altering the threshold voltage. This is achieved by applying a high voltage (20V) to the selected word line (WL 62). In order to allow for current flow from the source line to the bit line, all other word lines within the block need to be applied a bias voltage (10V).

Due to different voltage levels applied to adjacent pages (Page 61 and Page 63), the program operation to Page 62 can cause disturbance or the effect called Stress-Induced Leakage Current (SILC). Figure 3(c) illustrates a memory cell in adjacent pages (Page 61 and Page 63). The top gate of the memory cell is applied 10V, which makes a voltage difference between Page 62 and the adjacent pages. As shown in Figure 3(c), the disturbance will incur the loss of charge of the floating gate through the oxide layer to the substrate. Since the electrical charge of the floating gate defines the threshold voltage, any loss of charge can cause bit errors.

Current MLC flash stores two bits of information per cell. The values can be interpreted as four distinct states: 00, 01, 10, or 11. As shown in Figure 3(d), these four states are determined by reference voltages. The voltage can be

manipulated by the amount of charge put on the floating gate of the flash cell. Since the delta between each state has decreased, more rigidly controlled programming is needed to manipulate a more precise amount of charge stored on the floating gate. Program disturb will significantly influence the number of electrons stored in the floating gate and negatively impact the data integrity of flash.

The reliability problem caused by program disturb and read disturb becomes even worse for three-dimensional flash memory [3]. Three-dimensional NAND flash memory vertically stacks several conventional two-dimensional flash memory blocks to achieve higher density. *The program disturb and read disturb will not only influence the adjacent physical pages within the same physical block, but also influence the adjacent physical pages that have the same control gate stack.*

Flash file systems usually append updated data in continuous physical space of flash memory. The conventional physical block mapping strategy will consecutively allocate physical blocks, thereby causing the disturbance to previous physical blocks. Physical blocks can be classified as odd blocks and even blocks based on the physical block number. If write requests are first written to even blocks and skipping odd blocks, the program disturb will not disturb even blocks that contain the data. When all even blocks become used, the write requests then could be written to odd blocks. Since three-dimensional flash memory can provide ultra high capacity, it may not cause extra page errors. That is because, the adjacent physical blocks of each odd block may become invalid. These observations motivate us to propose a physical-location-aware physical block mapping strategy to enhance the reliability of three-dimensional NAND flash memory.

III. 3D-FlashMap: Physical-Location-Aware Mapping

A. Structure of 3D-FlashMap

This section presents *3D-FlashMap*, a novel physical-location-aware block mapping to reduce the effect of disturbance in three-dimensional flash memory. In conventional flash memory storage systems, spatial locality is normally adopted, and consecutive logical blocks (e.g., a file) will be allocated to consecutive physical blocks. Then, the logically neighboring blocks are also physically neighboring blocks. However, when disturbance effects are considered, the spatial locality results in severe disturbance to adjacent physical blocks in three-dimensional flash memory. 3D-FlashMap creates a physical distance between neighboring logical blocks in order to minimize program and read disturbances.

In 3D-FlashMap, several features are adopted to enhance the reliability of three-dimensional flash memory. First, 3D-FlashMap maintains a physical block mapping to track the physical location of each block. Second, 3D-FlashMap handles each read or write request from the MTD layer and reports the physical location of the corresponding physical block to the MTD layer. Third, 3D-FlashMap provides two reliability enhancement strategies with different levels of granularity to protect the integrity of data.

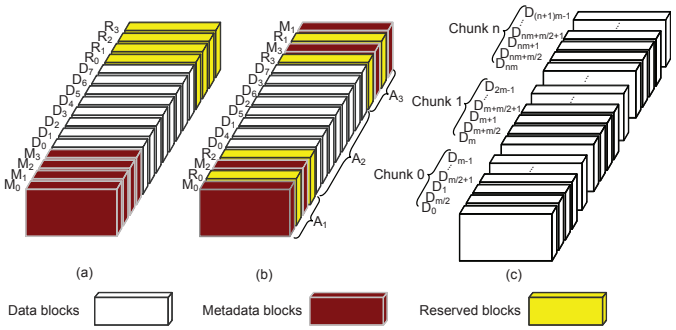

Fig. 4: (a) Physical block allocation for conventional scheme. (b) Physical block allocation for 3D-FlashMap. (c) Even-odd block differentiation strategy for data blocks.

In a flash memory storage system, physical blocks are normally classified into data blocks, metadata blocks, and reserved blocks. Data blocks usually contain the normal data. Metadata blocks are used to store the metadata (e.g., block mapping table, file system metadata, and bad block information). Reserved blocks are used to replace bad blocks due to wear-out. For a flash memory chip with N_{blk} physical blocks, assuming that there are N_{meta} metadata blocks, $N_{reserved}$ reserved blocks, and N_{data} data blocks. The number of metadata blocks N_{meta} is normally less than or equal to the number of reserved blocks $N_{reserved}$, and data blocks take up the majority of physical blocks. Traditionally, these blocks are arranged sequentially (M_0-M_3, D_0-D_7, and R_0-R_3 as shown in Figure 4(a)). The sequential allocation for physical blocks may work properly for traditional two-dimensional flash memory. However, for three-dimensional flash memory, the sequential allocation for physical blocks may cause more disturbance to adjacent physical pages.

B. Reliability Strategies of 3D-FlashMap

To reduce the effect of disturbance and ensure the data integrity for three-dimensional flash memory, 3D-FlashMap adopts two reliability enhancement strategies, *even-odd block differentiation* and *dynamic page skipping*. Even-odd block differentiation is a coarse-grained strategy that determines the physical location of each physical block. Dynamic page skipping is a fine-grained strategy that handles the next available physical page for each metadata block.

1) Even-Odd Block Differentiation: In 3D-FlashMap, physical blocks are divided into three physical areas based on the location: area one A_1 (physical blocks 0 to $\frac{1}{2} \times (N_{meta} + N_{reserved}) - 1$); area two A_2 (physical blocks $\frac{1}{2} \times (N_{meta} + N_{reserved})$ to $N_{blk} - \frac{1}{2} \times (N_{meta} + N_{reserved}) - 1$); and area three A_3 (physical blocks $N_{blk} - \frac{1}{2} \times (N_{meta} + N_{reserved})$ to $N_{blk} - 1$). Metadata blocks and reserved blocks are allocated at the first physical area A_1 and the third physical area A_3 of physical blocks, while the second physical area A_2 is used for data blocks.

3D-FlashMap utilizes the even-odd block differentiation strategy and creates a physical location map for physical blocks. Physical blocks in the first and the third physical areas (A_1 and A_3) alternate between metadata blocks and reserved

blocks. The physical location for each metadata block M_i and that for each reserved block R_i can be obtained from Equation 1 and Equation 2, respectively.

Figure 4(b) illustrates the block mapping for metadata blocks and reserved blocks. For the sake of illustration, there are 4 metadata blocks (M_0 to M_3) and 4 reserved blocks (R_0 to R_3). Based on Equations 1 and 2, in the first physical area A_1, the physical location of blocks is (M_0, R_0, M_2, R_2). There exists one reserved block (e.g., R_0) between every two metadata blocks (M_0 and M_2). Similarly, the physical block allocation in the third physical area A_3 can be obtained from Equations 1 and 2.

$$M_i = \begin{cases} i, & i = 2k, \\ N_{blk} - i, & i = 2k+1, \end{cases} \quad \forall k \in \mathbb{Z}_0^+, k \leq \frac{1}{2}(N_{meta} - 1) \tag{1}$$

$$R_i = \begin{cases} i+1, & i = 2k; i \leq N_{meta}, \\ N_{blk} - i - 1, & i = 2k+1; i \leq N_{meta} - 1, \\ \frac{1}{2}(i + N_{meta}), & i = 2k; i > N_{meta}, \\ N_{blk} - \frac{1}{2}(i + N_{meta} + 1), & i = 2k+1; i > N_{meta} - 1, \\ & \forall k \in \mathbb{Z}_0^+, k \leq \frac{1}{2}(N_{reserved} - 1) \end{cases} \tag{2}$$

Data blocks also adopt the even-odd block differentiation strategy. Figure 4(c) illustrates the physical block allocation for data blocks. Data blocks are logically partitioned into a set of data chunks, and each data chunk consists of several data blocks. Assuming that there are N_{chunk} data chunks and each data chunk contains m data blocks. Then $N_{data} = m \times N_{chunk}$. Blocks in a data chunk are further categorized into even blocks and odd blocks based on the actual physical block number. The physical block allocation of a data block D_i can be generated from Equation 3.

$$D_i = \begin{cases} \frac{N_{meta}+N_{reserved}}{2} - \lfloor \frac{i}{m} \rfloor \cdot m + 2i, & i - \lfloor \frac{i}{m} \rfloor \cdot m < \frac{m}{2}, \\ \frac{N_{meta}+N_{reserved}}{2} - (\lfloor \frac{i}{m} \rfloor + 1) \cdot m + 2i + 1, & i - \lfloor \frac{i}{m} \rfloor \cdot m \geq \frac{m}{2}, \\ & \forall i \in \mathbb{Z}_0^+, i \leq N_{data} - 1 \end{cases} \tag{3}$$

The number of physical blocks in a data chunk, m, is a configurable parameter. For example, assuming that a sequential write operation will consume N data blocks, then the number of physical blocks in a data chunk can be configured to contain $2N$ data blocks. Therefore, the original data will be written to even blocks, and it will not disturb physical blocks that contain valid data. The updated data will be written to odd blocks, which will only disturb physical blocks (even blocks) with invalid data. Chip designers can specify the parameter m according to the specific application. System users can also utilize this parameter and combine it with other reliability enhancement schemes to further improve the reliability of 3D flash memory.

2) Dynamic Page Skipping: Dynamic page skipping is the second strategy adopted in 3D-FlashMap, and it is applied to metadata blocks. Metadata blocks store file system metadata and other critical data that will directly influence the functionality of flash. If a flash memory page contains metadata, the data corruption of the page is very serious, as it may cause an unintended change in functionality of the entire flash. Therefore, to guarantee the functionality of the flash system, it is imperative to protect metadata and its storage blocks.

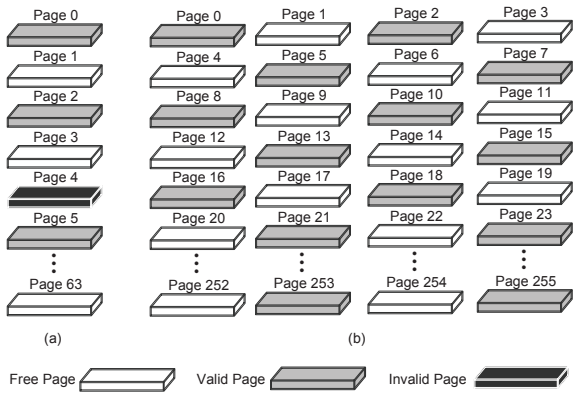

Fig. 5: Dynamic page skipping strategy for metadata blocks. (a) A metadata block with 64 pages, and each word line connects to one physical page. (b) A metadata block with 256 pages, and each word line connects to four physical pages.

Dynamic page skipping is a fine-grained strategy that provides page-level protection. Assuming that there are N_{page} pages in a physical block. Before issuing a program operation to a physical page P_i, $1 \leq i < N_{page} - 1$, it will check the status of the previous physical page P_{i-1}. If the physical page P_{i-1} is an invalid page or a free page, the dynamic page skipping strategy will directly write to page P_i. Otherwise, it will skip page P_i and write to page P_{i+1}.

Figure 5(a) illustrates the dynamic page skipping strategy for a metadata block with 64 pages. In this metadata block, each word line connects to one physical page. The dynamic page skipping strategy will first issue program operation to Page 0 and then write to Page 2. As a result, between two valid pages (Page 0 and Page 2), it will "skip" one free page (Page 1) to avoid program disturb. When Page 4 becomes invalid, the next program request will directly write the data to Page 5.

The dynamic page skipping strategy can also be applied to the metadata block, in which each word line connects to multiple physical pages. Figure 5(b) shows an example. In Figure 5(b), each location for valid page (e.g., Page 5) is surrounded with free pages (Page 1, Page 4, Page 6, and Page 9). The program operation to Page 5 will not disturb any valid pages. Therefore, the dynamic page skipping strategy can effectively reduce the disturbance to adjacent pages.

C. Overhead Analysis

3D-FlashMap adopts two reliability strategies, even-odd block differentiation and dynamic page skipping, to enhance the reliability of three-dimensional flash memory. The even-odd block differentiation strategy provides the physical mapping for each request, and it aims to permute the physical distance between the logically consecutive blocks to minimize the effect of disturbance. Since this strategy is at the block-level, it does not incur extra space overhead. The dynamic page skipping strategy skips one free page between two valid pages. Therefore, the dynamic page skipping strategy necessarily allocates more storage space to protect the integrity of metadata pages. Since metadata comprises a very small percentage of the total file system capacity, this strategy for metadata pages

will cause very low overhead. The experimental results also demonstrate that 3D-FlashMap can significantly improve the reliability of three-dimensional flash memory with less than 2% space overhead.

Furthermore, 3D-FlashMap can be implemented either as a hardware component or as a software component in a flash memory storage system. It can be integrated into the flash memory chip as a part of control logic circuit. 3D-FlashMap can also be incorporated into flash file system or the intermediate module called FTL to become a new level of mapping, which may provide more flexible configurations for specific applications. 3D-FlashMap provides the physical block map using linear functions to track physical locations, while requiring zero modifications to the hardware of flash chip or file system. Therefore, 3D-FlashMap can be implemented with low overhead.

IV. Evaluation

A. Experimental Setup

The proposed 3D-FlashMap was implemented on Fedora 7 (Linux kernel 2.6.21). The primitive read, write, and erase operations over raw flash memory are provided by Memory Technology Device (MTD) [14]. We modify the built-in NAND flash simulator (NANDsim) in Linux kernel. A representative scheme MNFTL [13] is selected as the baseline FTL scheme because of its relatively good performance in terms of wear-leveling and endurance for MLC flash memory.

The trace of data request was collected from desktop running DiskMon [15]. To eliminate the effects of operating system's internal operations, a 1TB external hard drive enclosure is used to collect disk access characteristics. The I/O traces reflect the real workload of the system in accessing the hard disk with daily-used applications. Table I lists the characteristics of traces.

TABLE I: Characteristics of traces.

Trace	# of write operations	# of read operations	% of write	% of read
chatOnline	559,085	472,907	54.18	45.82
p2p	3,695,873	2,101,474	63.75	36.25
replaceFiles	3,145,994	2,128	99.93	0.07
zipFiles	2,588,585	2,614,055	49.76	50.24

In our experiments, a 64GB NAND flash memory is configured based on the specifications of a typical 3D flash memory structure BiCS [2] from TOSHIBA . The page size, the number of pages in a block, the block size, the size of spare area of a page, the number of metadata blocks, the number of reserved blocks, and the number of data blocks in a chunk are set as 4KBytes, 64, 256KBytes, 12Bytes, 512, 512, and 256, respectively. Each word line is configured to connect with one physical page. Each page can provide 8-bit error correction. We simulate a set of errors caused by program disturb and read disturb. The distribution and the probability of errors are obtained from Grupp et al. [4].

B. Results and Discussion

1) Uncorrectable Page Errors: Program disturb and read disturb will lead to the corruption of data that could not be corrected by ECC (Error Correcting Codes). To be specific, we define uncorrectable page errors as follows: when data d_1 is written in a flash memory address (i.e., a page) and then data d_2 is read from that address, there is an uncorrectable page error if $d_1 \neq d_2$.

Figure 6 presents the number of uncorrectable page errors. Not surprisingly, 3D-FlashMap can significantly reduce the number of errors. That is because, an uncorrectable page error can be caused as program disturb or read disturb to adjacent pages in data blocks, or can be as the result of disturbance to adjacent pages in metadata blocks. Metadata blocks store the metadata (e.g., mapping table) that will be frequently accessed, so they are prone to have more page errors compared with data blocks. 3D-FlashMap utilizes the dynamic page skipping strategy to enhance the reliability of metadata blocks. Therefore, it can significantly reduce the page errors due to the disturbance to metadata blocks. As shown in Figure 6, 3D-FlashMap can achieve an 85% reduction on average compared with the baseline scheme.

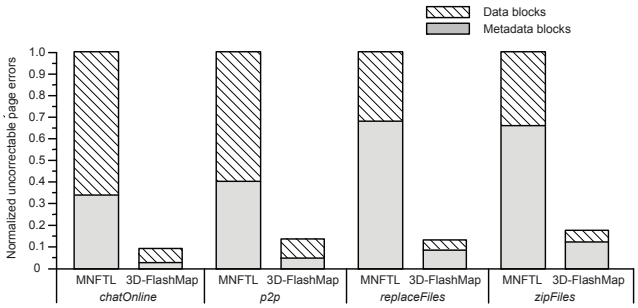

Fig. 6: The number of uncorrectable page errors for the baseline scheme and 3D-FlashMap.

2) Block Erase Counts: 3D-FlashMap adopts the dynamic page skipping strategy for metadata blocks, which will skip several free pages and cause more erase operations. This overhead is quantified, and the results are shown in Figure 7. We observe that 3D-FlashMap causes very low extra erase operations (up to 1.78%) in comparison with the baseline scheme. For traces *chatOnline* and *p2p*, they contain many update operations with randomly accessed logical addresses. The extra block erase counts are due to the adoption of the dynamic page skipping strategy, which may use several extra metadata blocks to save metadata (e.g., mapping information). For traces *replaceFiles* and *zipFiles*, the requests are allocated to sequential logical addresses, and these requests do not contain many update operations. For these traces, they may slightly increase the block erase counts for data blocks. This is because, program disturb and read disturb may cause the data corruption along the path of address mappings from a logical page number to a physical page number. Then several data blocks will become untrackable blocks due to data corruption of metadata blocks. The updated operation for these kinds of untrackable blocks will incur an uncorrectable page error, and

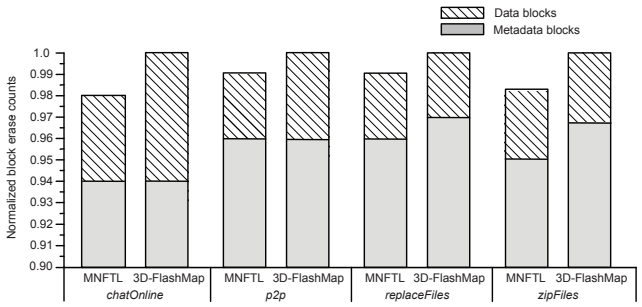

Fig. 7: The total number of block erase counts for the baseline scheme and 3D-FlashMap.

the baseline scheme will not issue erase operations as it is supposed to do.

3) Endurance: The endurance of NAND flash is one of the most important factors in analyzing the reliability of flash memory. The endurance of NAND flash is mainly affected by the worst case erase count of a physical block in the flash. Figure 8 presents the results. We observe that 3D-FlashMap can achieve similar or even better results in comparison with the baseline scheme. Since 3D-FlashMap adopts the block-level strategy with even-odd block differentiation to enhance the reliability of three-dimensional flash, it will specify the physical location of each physical block. 3D-FlashMap will not change the logical address to physical address mapping information, which is handled by file system or the intermediate module called flash translation layer. Therefore, 3D-FlashMap will not influence the endurance and the wear-leveling of flash.

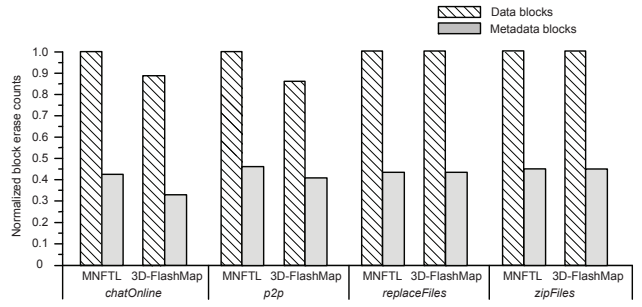

Fig. 8: The maximum number of block erase count for the baseline scheme and 3D-FlashMap.

For traces *chatOnline* and *p2p*, we observe that 3D-FlashMap can achieve better performance compared with the baseline scheme in terms of the number of the worst case erase count. That is because, untrackable blocks caused by the data corruption of metadata blocks may contain valid pages. These untrackable blocks may not be erased and reclaimed for further usage until there are no free blocks left in flash. These kinds of blocks will occupy flash memory space, which will reduce the available physical spaces in flash memory and negatively impact the endurance. Although 3D-FlashMap may introduce low overhead in terms of the total number of block erase counts, it can achieve similar or even better worst case erase count by distributing erase operations across different blocks.

V. Conclusion

In this paper, we presented 3D-FlashMap, the first reliability enhancement scheme for three-dimensional NAND flash memory. By providing physical distance between the logically consecutive blocks in flash memory, 3D-FlashMap permutes the physical distance between two consecutively accessed blocks, thereby reducing the effect of read disturb and program disturb. 3D-FlashMap also introduces a fine-grained strategy to ensure the data integrity for metadata blocks. Experimental results show that our 3D-FlashMap scheme reduces the number of uncorrectable errors by 85%, achieves the similar or better endurance of flash in comparison with the baseline scheme, while incurring up to 1.78% space overhead. In the future, we plan to investigate the reliability issue of three-dimensional flash memory with the support of the file system.

Acknowledgment

The work described in this paper is partially supported by the grants from the Innovation and Technology Support Programme of Innovation and Technology Fund of the Hong Kong Special Administrative Region, China (ITS/082/10).

References

[1] E.-S. Choi, H.-S. Yoo, H.-S. Joo, G.-S. Cho, S.-K. Park, and S.-K. Lee, "A novel 3D cell array architecture for terra-bit NAND flash memory," in *3rd IEEE International Memory Workshop*, 2011, pp. 1 –4.

[2] H. Tanaka, M. Kido, K. Yahashi, M. Oomura, R. Katsumata, M. Kito, Y. Fukuzumi, M. Sato, Y. Nagata, Y. Matsuoka, Y. Iwata, H. Aochi, and A. Nitayama, "Bit cost scalable technology with punch and plug process for ultra high density flash memory," in *VLSIT '07*, 2007, pp. 14 –15.

[3] J. Kim, A. Hong, S. Kim, K.-S. Shin, E. Song, Y. Hwang, F. Xiu, K. Galatsis, C. Chui, R. Candler, S. Choi, J.-T. Moon, and K. Wang, "A stacked memory device on logic 3D technology for ultra-high-density data storage," *Nanotechnology*, vol. 22, no. 25, pp. 332–343, 2011.

[4] 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 *MICRO '09*, 2009, pp. 24–33.

[5] Y. Kang and E. L. Miller, "Adding aggressive error correction to a high-performance compressing flash file system," in *EMSOFT '09*, 2009, pp. 305–314.

[6] P.-H. Hsu, Y.-H. Chang, P.-C. Huang, T.-W. Kuo, and D. H.-C. Du, "A version-based strategy for reliability enhancement of flash file systems," in *DAC '11*, 2011, pp. 29–34.

[7] T.-W. Kuo, Y.-H. Chang, P.-C. Huang, and C.-W. Chang, "Special issues in flash," in *ICCAD '08*, 2008, pp. 821–826.

[8] Y. Wang, D. Liu, Z. Qin, and Z. Shao, "An endurance-enhanced flash translation layer via reuse for NAND flash memory storage systems," in *DATE '11*, 2011, pp. 14–20.

[9] R. Katsumata, M. Kito, Y. Fukuzumi, M. Kido, H. Tanaka, Y. Komori, M. Ishiduki, J. Matsunami, T. Fujiwara, Y. Nagata, L. Zhang, Y. Iwata, R. Kirisawa, H. Aochi, and A. Nitayama, "Pipe-shaped BiCS flash memory with 16 stacked layers and multi-level-cell operation for ultra high density storage devices," in *VLSIT '09*, 2009, pp. 136 –137.

[10] J. Jang, H.-S. Kim, W. Cho, H. Cho, J. Kim, S. I. Shim, Y. Jang, J.-H. Jeong, B.-K. Son, D. W. Kim, Kihyun, J.-J. Shim, J. S. Lim, K.-H. Kim, S. Y. Yi, J.-Y. Lim, D. Chung, H.-C. Moon, S. Hwang, J.-W. Lee, Y.-H. Son, U.-I. Chung, and W.-S. Lee, "Vertical cell array using TCAT (Terabit Cell Array Transistor) technology for ultra high density NAND flash memory," in *VLSIT '09*, 2009, pp. 192 –193.

[11] J. Kim, A. Hong, S. M. Kim, E. Song, J. H. Park, J. Han, S. Choi, D. Jang, J. T. Moon, and K. Wang, "Novel Vertical-Stacked-Array-Transistor (VSAT) for ultra-high-density and cost-effective NAND flash memory devices and SSD (Solid State Drive)," in *VLSIT '09*, 2009, pp. 186 –187.

[12] W. Kim, S. Choi, J. Sung, T. Lee, C. Park, H. Ko, J. Jung, I. Yoo, and Y. Park, "Multi-layered vertical gate NAND flash overcoming stacking limit for terabit density storage," in *VLSIT '09*, 2009, pp. 188 –189.

[13] Z. Qin, Y. Wang, D. Liu, Z. Shao, and Y. Guan, "MNFTL: An efficient flash translation layer for MLC NAND flash memory storage systems," in *DAC '11*, 2011, pp. 17 –22.

[14] "Memory Technology Device (MTD) Subsystem for Linux," *http://www.linux-mtd.infradead.org/*, 2011.

[15] "DiskMon for Windows," *http://technet.microsoft.com/en-us/sysinternals/ bb896646.aspx*.