Adaptive Paired Page Prebackup Scheme for MLC NAND Flash Memory

Jaeil Lee and Dongkun Shin, *Member, IEEE*

***Abstract*—Multilevel cell (MLC) NAND flash memory is more cost effective compared with single-level cell NAND flash memory as it can store two or more bits in a memory cell. However, in MLC flash memory, a programming operation can corrupt the paired page under abnormal termination. In order to solve the paired page problem, a backup scheme is generally used, which inevitably causes performance degradation and shortens the lifespan of flash memory. In this paper, we propose a more efficient paired page prebackup scheme for MLC flash memory. It adaptively exploits interleaving, copyback operations, and parity data to reduce the prebackup overhead. In experiments, the proposed scheme reduced the backup overhead by up to 78%.**

***Index Terms*—Adaptive LSB prebackup, flash translation layer, multilevel cell (MLC), NAND flash memory, storage.**

I. Introduction

NAND flash memory has several advantages, such as non-volatility, shock resistance, and low power consumption. Thus, it is widely used for mobile devices, such as digital cameras, tablet PCs, and smartphones. However, it has several features that must be carefully handled. First, the I/O unit of a read or write operation is a page, which is typically 4 KB or 8 KB; an erase operation is performed by a block that consists of several pages. Second, a page cannot be overwritten before the corresponding block is erased. This characteristic is called the erase-before-write constraint. Therefore, flash memory does not permit in-place update and requires a logical-to-physical address mapping scheme. Third, there is a limit on the maximum number of program/erase (P/E) cycles. If a block is programmed and erased more than the specified maximum number of P/E cycles, the block becomes worn-out and unreliable. To handle the idiosyncrasies of flash memory, special software, called a flash translation layer (FTL) [3], is embedded within the NAND flash memory-based systems such as the embedded multimedia card (eMMC) and solid state disk (SSD). Generally, an FTL provides several functions such as logical-to-physical address mapping, garbage collection, and wear leveling.

NAND flash memory can be classified into two types: single-level cell (SLC) and multilevel cell (MLC) NAND flash memory. SLC flash memory can store one bit per memory cell, whereas MLC flash memory can store two or more bits. In 2-bit MLC flash memory, the cells of one wordline can store two paired pages and can be programmed twice for the two paired pages, called the least significant bit (LSB) page and the most significant bit (MSB) page. The LSB page should be programmed first and then the MSB page can be programmed. The program time for the MSB page is longer than that of the LSB page. On a same-sized die, MLC flash memory provides higher density than SLC flash memory does, and thus most of the recent flash memory-based systems adopt MLC flash memory. Throughout this paper, MLC flash memory represents 2-bit MLC as 3-bit MLC flash memory is generally called triple-level cell (TLC) memory.

Manuscript received October 9, 2013; revised January 7, 2014; accepted February 19, 2014. Date of current version June 16, 2014. This work was supported by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education under Grant 2013R1A1A2A10013598. This paper was recommended by Associate Editor J. Henkel.

J. Lee is with Samsung Electronics, Hwasung 445-330, Korea (e-mail: ji007.lee@samsung.com).

D. Shin (corresponding author) is with the Department of Computer Science and Engineering, Sungkyunkwan University, Suwon 440-746, Korea (e-mail: dongkun@skku.edu).

Digital Object Identifier 10.1109/TCAD.2014.2309857

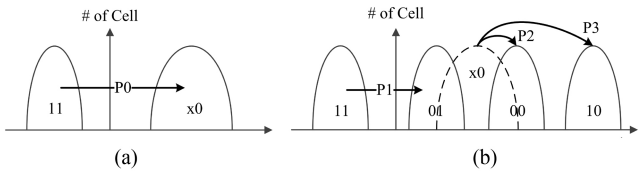

Fig. 1. Changes in threshold voltage (V_{th}) distribution during the page programming of MLC NAND flash memory. (a) LSB program. (b) MSB program.

Although MLC flash memory affords higher capacity, SLC memory achieves higher program speed and endurance. Moreover, MLC flash memory suffers from the paired page interference problem. As the paired pages share the same memory cells, if a program operation for the MSB page is abnormally aborted by power failure, reset, or program failure, the paired LSB page can be damaged as well as the MSB page [6]. Therefore, the FTL for MLC NAND flash memory should provide data recovery methods for preventing data corruption by the paired page interference problem. One simple technique is to back up the LSB page before the write operation of the MSB page. Considering that NAND flash memory is usually used in mobile devices, which are exposed to sudden power-off, the data recovery method is crucial for system safety. However, the backup-based power crash recovery schemes inevitably cause write performance degradation.

In this paper, we propose an efficient prebackup scheme to minimize the LSB page backup overhead. The proposed scheme exploits the features of flash memory-based systems, such as interleaving and copyback operations. In addition, it also utilizes the parity data to reduce the backup overhead.

II. Backgrounds

A. MLC NAND Flash Memory

Fig. 1 shows the process of a paired page program in MLC NAND flash memory. The memory cell is initially in the erased state with a 2-bit value of 11. From the erased state, the LSB of the memory cell is programmed first, followed by programming of the MSB of the memory cell. In the LSB program, if the LSB to be programmed is the logical value 0, the cell state moves from the erased state (11) to the temporary state (x0, P0 transition in Fig. 1), which will be further programmed by the MSB program step [5]. Otherwise, the memory cell remains in the 11 state. The x0 state will output the logical value 0 for an LSB page read operation.

In the MSB program, if the MSB to be programmed is the logical value 0, the cell state is changed from the 11 state to the 01 state by the P1 transition, or, depending on the previous state of the memory cell, the cell state is changed from the temporary state (x0) to the 00 state by the P2 transition. If the MSB to be programmed in the memory cell is the logical

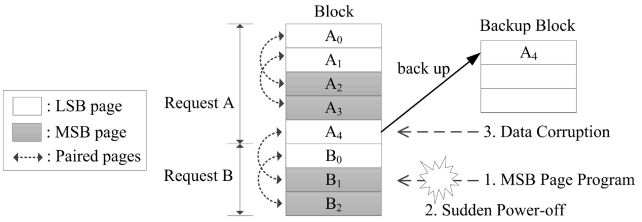

Fig. 2. Paired page problem in MLC NAND flash ($\pi = 2$).

value 1, the cell state is changed from the temporary state (x0) to the 10 state by the P3 transition, or, depending on the previous state of the memory cell, remains in the 11 state. Eventually, the memory cell has one of four distinct threshold voltage distributions: the states of 11, 01, 00, and 10. The MSB pages are more than three times slower than the LSB pages in program time [2]. As the MSB programming modifies the state of the memory cells, the LSB can be corrupted if the MSB programming is abnormally terminated, and thus, the memory cells are not programmed into one of four distinct states [5]. This phenomenon is called retroactive data corruption in [6].

In this paper, we designate the paired page interval by π, which means the nth page and $(n + \pi)$th page in a block are associated as paired pages. The paired page intervals of flash chips vary depending on the manufacturing technology.

B. LSB Backup in MLC Flash Memory

To prevent data corruption of the paired LSB page during the programming of an MSB page, current MLC flash devices use the LSB page backup scheme [4]. Before the MSB page programming, the paired LSB page is copied to a backup block. The LSB page can then be recovered even when it is corrupted during the MSB page programming. Generally, the backup MLC block is programmed by the SLC mode, where only the LSB pages are used to minimize the backup overhead as the MSB page program time is significantly longer than the LSB page program time.

Fig. 2 shows the pattern of paired pages in MLC flash blocks and the LSB page backup scheme assuming that the paired page interval is 2. In this example, a host system sends two requests, A and B. The MLC flash device writes the five pages, A_0 to A_4, of request A, and then it writes the three pages, B_0 to B_2, of request B. Before writing page B_1, page A_4 must be backed up to prevent data corruption during the programming of B_1, as A_4 and B_1 are located at the paired pages.

Note that the paired page problem can occur only between distinct write requests. For the other paired pages, (A_0, A_2), (A_1, A_3), and (B_0, B_2), there is no need to perform the LSB backup operation as the paired pages are included within the same requests. If there is a sudden power-off or program failure during the handling of a write request, then the incomplete write request is terminated, and all the written data of the request will be handled as invalid by the FTL. Therefore, the LSB backup overhead decreases proportional to the increasing size of write request. The maximum number of LSB pages to be backed up during the handling of a write request is π.

Another case in which the LSB backup is not required is when the LSB page is invalidated before the write operation on the paired MSB page. Considering such a case, current MLC devices use the post-backup scheme, which copies the paired LSB page to the backup block just before writing the corresponding MSB page. Therefore, it can avoid the backup operation if the paired LSB page is invalidated before the write request on the corresponding MSB page. However, it degrades the latency of the write operation, as the MSB page write operation must wait until the completion of the paired LSB page backup operation.

This paper proposes the prebackup scheme, which performs the LSB page backup at the same time or just after the LSB page is written in the data block. Therefore, the prebackup scheme does not increase the write latency of the MSB page. The prebackup also needs to copy at most the last π pages of a write request. As the program latency of an MSB page is significantly larger than that of an LSB page, the prebackup is a better solution for reducing the maximum write latency.

There are few solutions on the paired page problem in MLC NAND flash memory. Lee *et al.* [4] proposed a block allocation algorithm to avoid paired LSB page backup in MLC NAND flash-based database systems. The algorithm allocates a physical block to a transaction only when the physical block has no valid transaction data. Therefore, there are no LSB pages to be backed up. In addition, within a transaction, the LSB page backup is not required as a transaction can be recovered by the DBMS. This algorithm, however, can only be applied to database systems.

III. Adaptive Paired Page Prebackup

Our scheme uses the prebackup technique to reduce the backup overhead and the maximum write latency. We propose an adaptive paired page prebackup scheme, which selects one of three prebackup techniques depending on the data size: interleaving prebackup, copyback prebackup, and parity page prebackup.

A. Prebackup Schemes

1) *Interleaving Prebackup:* Recent flash memory-based storage systems, such as eMMC and SSD, contain multiple NAND flash chips to increase the I/O bandwidth by accessing them in parallel. For this purpose, multichannel and multiway architecture are used. The multiple I/O channels can independently issue read or write operations and transfer data. By writing to multiple NAND flash chips simultaneously via concurrent channels, the write performance can be improved. In the multiway architecture, multiple chips can share a single channel in an interleaved manner, wherein the program operations at different chips sharing one channel can be overlapped although the flash chips cannot use the channel simultaneously. The total number of the pages concurrently programmable via the multiple channels and ways is denoted by λ in this paper. We utilize the parallel I/O architecture to hide the LSB page backup latency.

If $\omega \leq \lambda/2$, where ω represents the page size of write request, and there are idle channels and ways for writing 2ω pages, the interleaving prebackup scheme writes both the original data and backup data for the LSB page simultaneously by using different channels and ways. The interleaving prebackup invokes no backup overhead as the backup operation is overlapped with the normal data write operation. However, the interleaving scheme can increase the channel utilization degrading the performance, and it can be used only when

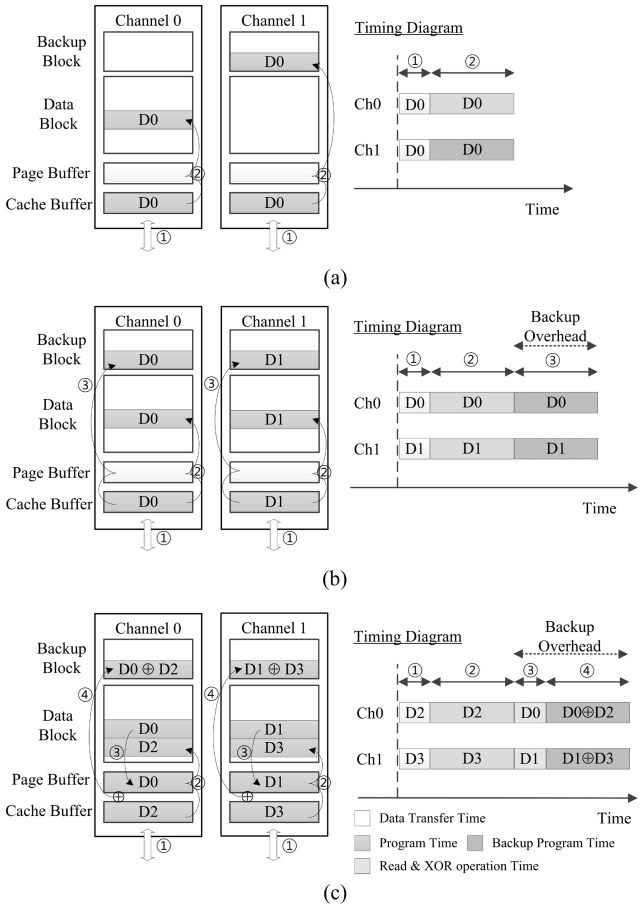

Fig. 3. Proposed prebackup schemes. (a) Interleaving prebackup scheme. (b) Copy-back prebackup scheme. (c) Parity page prebackup scheme.

there are idle channels and ways not occupied by previous requests. After completing the MSB page program in the data block, the data of the paired LSB page in the backup block are immediately invalidated.

Fig. 3(a) shows the interleaving prebackup scheme. The original data and the backup data are programmed simultaneously via channel 0 and channel 1, respectively. As shown in Fig. 3, recent NAND flash memory chips have two internal data buffers, known as the cache buffer and page buffer, to enable pipelined operations. The data are first written in the cache buffer by the FTL, and then it is programmed into the flash block via the page buffer. During the program operation for the data in the page buffer, the FTL can send the next page to the cache buffer. Therefore, with the pipeline scheme, the data transfer time and the data program time of the two different pages can be overlapped.

2) *Copyback Prebackup:* If $\lambda/2 < \omega \leq \lambda$ or there are no idle channels and ways for the interleaving backup even when $\omega \leq \lambda/2$, the copyback prebackup is used, which copies the original data of the LSB page into a backup block just after the original data is written in the data block. Therefore, it cannot hide the backup operation latency unlike the interleaving scheme. However, the copyback prebackup operation can quickly copy the original data without reloading data to the page buffer. While the post-backup requires one read operation and one write operation for the LSB page backup, the copyback prebackup requires only one write operation. Fig. 3(b) presents the copyback prebackup scheme. The data, D_0 and D_1, are programmed into the data block first. After writing the data, they still remain in the internal page buffers of flash memory chips. Therefore, the backup operation can reuse these data. They can be programmed into the backup block without the page read time. Although the copyback scheme can reduce the backup overhead when the size of write request is large, it can reduce only one data loading time; therefore, the performance improvement is insignificant.

3) *Parity Page Prebackup:* If $\omega > \lambda$, the copyback scheme cannot be used as the internal buffer is overwritten during the write operations. In such a case, the parity page prebackup scheme can be used, which writes only one backup page for each two LSB pages by utilizing the parity data generated from them. Therefore, the number of backup pages in each chip is $\lceil \pi/2 \rceil$ at most. We call the two LSB pages associated via the parity parity-paired LSB pages. Fig. 3(c) shows the parity page prebackup scheme. The host sends four pages; D_0, D_1, D_2, and D_3. The FTL first writes D_0 and D_1, and then it writes D_2 and D_3, in the LSB pages via two channels. After programming D_2 and D_3, the data still remain in both the page and the cache buffers. To get the parity data, the FTL reads D_0 and D_1 from the data block into the page buffer. The cache buffers then have D_2 and D_3, and the page buffers have D_0 and D_1. The device calculates the parity data, $D_0 \oplus D_2$ and $D_1 \oplus D_3$, with the XOR circuit embedded in the NAND flash memory. Most of the current flash memory chips have an XOR circuit to generate the signature data. The FTL writes the generated parity data into the backup block.

In MLC NAND flash memory, the latency of write operation is more than ten times longer than that of read operation. As the parity page prebackup scheme replaces one write operation with one read operation, the LSB page backup overhead is significantly reduced. In addition, the parity page prebackup scheme can reduce the number of programs in the backup blocks, and thus the lifespan of the backup blocks is improved. In this scheme, when a power failure occurs during the MSB page write operation, only one additional read operation is required to recover the paired LSB page data, as will be explained in Section III-B. However, as the read response time is short and power failures are rare, the recovery cost is negligible.

4) *Worst-Case Overhead of Prebackup Schemes:* When no LSB backup scheme is used, the total write latency for an MLC flash memory block can be modeled as follows:

$$T_{nobackup} = N_{\text{page}} \times (T_{xfr} + 0.5 \times (T_{w,lsb} + T_{w,msb})) \quad (1)$$

where N_{page} is the number of pages in a block, and T_{xfr} is the data transfer time between the host and flash memory chip. The variables $T_{w,lsb}$ and $T_{w,msb}$ are the write latencies of an LSB page and an MSB page, respectively. We should use the average value of them as there is the same number of LSB and MSB pages in an MLC block.

For the post-backup scheme, we can formalize the worst-case LSB page backup overhead as follows:

$$T^{\text{post}}_{\text{overhead}} = 0.5 \times N_{\text{page}} \times (T_r + T_{w,lsb}) \times (1 - P_{\text{invalid}}) \quad (2)$$

where T_r is the page read latency and P_{invalid} is the probability that the target LSB will be invalidated before the paired MSB page is programmed. The worst case is when the LSB backup

is required for every MSB page programming. In the worst case, the post-backup scheme requires one read operation and one LSB page write operation for each MSB write operation, except when the LSB page is invalidated before the write operation.

For the copyback prebackup scheme, we can formalize the worst-case LSB page backup overhead as follows:

$$T^{cb}_{overhead} = 0.5 \times N_{page} \times T_{w,lsb}. \quad (3)$$

If we assume that N_{page}, T_r, $T_{w,lsb}$, $T_{w,msb}$, and T_{xfr} are 128, 60 μs, 600 μs, 2 ms, and 30 μs, respectively, T_{nobackup} is 170.24 ms. The overhead ratio, $T^{\text{post}}_{\text{overhead}}/T_{\text{nobackup}}$, is $(1 - P_{\text{invalid}}) \times 42.24\,\text{ms}/170.24\,\text{ms} = (1 - P_{\text{invalid}}) \times 25\%$, and $T^{cb}_{\text{overhead}}/T_{\text{nobackup}}$ is 23% (= 38.4 ms/170.24 ms). Therefore, the copyback prebackup is better than the post-backup when $P_{\text{invalid}} < 8\%$ (= 1−23/25).

For the maximum write latency of a page, the post-backup requires 2.69 ms (= $T_{xfr} + T_{w,msb} + T_r + T_{w,lsb}$) when an MSB page is written with a backup operation, while the prebackup requires 2.03 ms (= $T_{xfr} + T_{w,msb}$) since no backup operation is required for an MSB page programming. The write latency of an LSB page in the prebackup scheme is 1.23 ms (= $T_{xfr} + 2 \times T_{w,lsb}$).

Under the parity page prebackup scheme, the LSB backup overhead is as follows:

$$T^{parity}_{overhead} = 0.25 \times N_{page} \times (T_r + T_{w,lsb}). \quad (4)$$

As the scheme writes only one page in the backup block for every two LSB pages, we need only one-fourth of the N_{page} backup operations. The backup overhead ratio is only 12% (= 21.12 ms/170.24 ms) for the real latency values.

B. Paired Page Data Recovery

When the LSB page is corrupted during the paired MSB page write operation, our recovery scheme finds the original data from the backup block. A page in the backup block stores the logical page number (LPN) and physical page number (PPN) of the original data, and the parity flag (PF) in the spare area, which is a region reserved for the FTL metadata and error correcting code (ECC). The LPN is the logical address given by the host, and the PPN is the physical address used in NAND flash memory. Generally, the FTL maintains the LPN-to-PPN (L2P) mapping table to support address translation. In our recovery scheme, the LPN and PPN in the backup block's spare area are used to find the corresponding backup data for the corrupted data. The PF is used to distinguish the parity page from the normal backup data.

For a sudden power-off, the FTL performs the crash recovery during the system initialization phase. The FTL scans all of the pages in the data blocks and finds corrupted LSB pages by checking the ECC data in the spare area of each page. Fig. 4 shows the recovery process. For example, the data **A** for LPN 1 is written at PPN 11 as shown in the L2P mapping table. If the page with PPN 11 is corrupted, the FTL first finds the PPN from the page mapping table. If the PPN cannot be found, the corrupted page has invalid data, and thus the recovery process is not required. Otherwise, the FTL scans the backup blocks to find the corresponding backup page which has the same PPN and LPN in the spare area (PPN 30 in Fig. 4). The found backup page is copied into a free page of the data block. Finally, the FTL updates the mapping table.

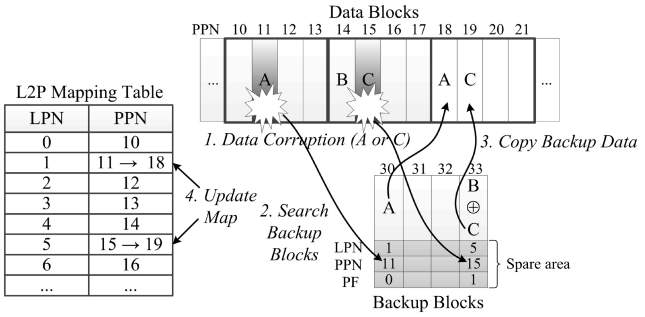

Fig. 4. Paired page data recovery.

If the PF field of the found backup page is set (PPN 33 in Fig. 4), the recovery process reads the parity-paired LSB page data (B at PPN 14) from the data block, and calculates the parity value to get the original data of the corrupted page. Since only one LSB page can be corrupted by abnormal MSB page programming, we can always get the noncorrupted parity-paired LSB page for corrupted data.

IV. Experiments

In order to evaluate the performance of the proposed adaptive paired page prebackup scheme, we implemented a flash memory-based storage system simulator and a page-level FTL algorithm similar to the demand-based map loading FTL (DFTL) [3]. We also added the proposed LSB page backup schemes to the FTL algorithm. In order to target smartphone workloads, the flash memory simulator is configured to simulate the current eMMC devices, which are used as the internal storage device of smartphones. We assumed that the target eMMC is equipped with four 8-GB, 2-bit MLC NAND flash memories. We used the timing parameters for the MLC NAND flash memory in [1], where the paired page interval, π, is 3. The target eMMC uses 2-way×2-channel architecture, where four 32-KB pages can be read/written simultaneously. In the simulation, we reserved two backup blocks in each flash chip; one is used for the interleaving backup and the other is for the copyback/parity backup.

We compared the performances of four different backup schemes: 1) no-backup where the backup operation is not performed; 2) post-backup; 3) prebackup; and 4) adaptive prebackup where all the proposed schemes are applied depending on the request size. The prebackup scheme simply copies the target LSB page to the backup block without using any proposed optimization technique. We first used synthetic workloads to observe the performance under different I/O sizes. We collected six Iozone traces with different I/O sizes in an Android-based smartphone: Iozone_4, Iozone_8, Iozone_32, Iozone_64, Iozone_128, and Iozone_512. The Iozone_n trace is generated by running the Iozone program with the option "-s 128m -r nk -i 2," which performs random write operations with n KB of record size for an 128-MB file.

We compared the average write latencies of four different backup schemes. Fig. 5(a) shows the average write latencies of the post-backup, prebackup, and adaptive prebackup schemes

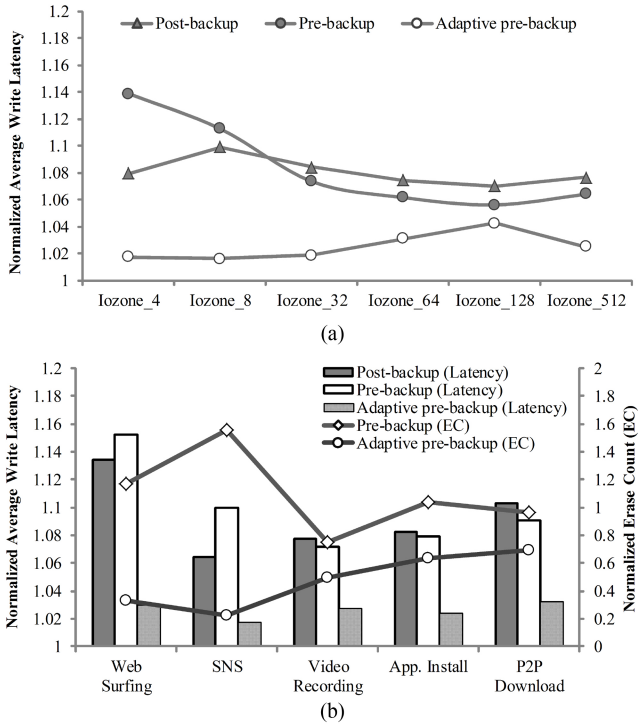

Fig. 5. Comparisons of average write latency normalized by no-backup scheme. (a) Iozone. (b) Real workload.

normalized by the no-backup scheme. post-backup and prebackup, at a maximum, represent 10% and 13.8% of the backup overhead, respectively. These values are smaller than the worst-case overhead ratios calculated in Section III-A4. As (1) does not consider the garbage collection overhead, the average write latencies in the experiments are larger.

In the post-backup and prebackup schemes, as the I/O size increases, the backup overhead generally decreases. This is because the total number of required backup operations decreases in large write requests. Compared with post-backup, prebackup reduces the backup overhead for the I/O sizes larger than 8 KB, as prebackup avoids one read operation for the LSB page to be backed up. However, when the I/O size is 4 KB or 8 KB, post-backup outperforms prebackup. For the I/O sizes of 4 KB and 8 KB, P_{invalid}, the probability of invalid paired LSB pages, is significantly large. As $P_{\text{invalid}} > 8\%$, there are many cases when post-backup can skip the backup operation, as commented on in Section III-A4.

The proposed adaptive prebackup scheme dramatically improves the write performance in all of the write patterns. This is especially the case when the I/O size is not larger than 32 KB; the backup overhead is less than 2% due to the effect of the interleaving prebackup scheme. As the target eMMC device can write 128 KB at once with the two-channel and two-way architecture, the original data and the backup data can be programmed simultaneously when the I/O size is smaller than 64 KB. Therefore, most of backup operation time was hidden by the interleaving prebackup scheme. (When the I/O size is 64 KB, the interleaving backup cannot be fully used as some requests are not aligned to 64 KB.) For an I/O size of 128 KB, adaptive prebackup shows the maximum backup overhead as the copyback prebackup scheme is mainly applied. The copyback prebackup scheme cannot significantly reduce the backup overhead. For an I/O size of 512 KB, the parity page prebackup scheme may be frequently applied. Therefore, the backup overhead in Iozone_512 is less than that of Iozone_128. Nevertheless, adaptive prebackup outperforms both post-backup and prebackup for all I/O sizes, as the interleaving, copyback, and parity page backup schemes are adaptively applied according to the size of the write request.

We also collected real-world workloads from a smartphone while executing different applications: web surfing, social networking service (SNS), video recording, application install, and P2P download. While the write patterns for web surfing and SNS workloads are small and random, the write patterns for the remaining workloads are large and sequential.

Fig. 5(b) compares the backup overhead for different backup schemes under each smartphone workload. Adaptive prebackup significantly outperforms other schemes for all workloads. In the random I/O dominant workloads, post-backup is better than prebackup. Adaptive prebackup reduces the backup overhead by up to 78% compared with post-backup, and by up to 82% compared with prebackup.

The adaptive prebackup scheme can also improve the reliability of MLC NAND flash memory. Fig. 5(b) also compares the erase counts of the backup blocks under prebackup and adaptive prebackup schemes. The values are normalized by those of the post-backup scheme. The erase counts are reduced by 31–78% in the proposed scheme. As the parity page prebackup scheme can reduce the number of writes in the backup blocks, adaptive prebackup reduces the erase count of the backup blocks and thus improves the lifespan of the backup blocks.

V. Conclusion

The paired page backup operation is the main reason for the slow performance of MLC flash memory. In this paper, we proposed an adaptive paired page prebackup scheme for MLC flash memory, which adaptively uses the interleaving, copyback operation, and parity page to reduce the backup overhead. The proposed prebackup schemes can also reduce the maximum write latency and can improve the lifespan of MLC flash memory by reducing the number of write operations on the backup blocks. Experimental results show that the adaptive prebackup scheme outperforms the existing backup schemes, imposing less than 4% of the backup overhead.

References

[1] Samsung Electronics Company, Ltd., 32 Gb MLC NAND Flash Memory, Tech. Rep. K9GBG08U0A, 2009.
[2] L. M. Grupp *et al.*, "Characterizing Flash memory: Anomalies, observations, and applications," in *Proc. MICRO'09*, pp. 24–33.
[3] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: A Flash translation layer employing demand-based selective caching of page-level address mappings," in *Proc. ASPLOS'09*, pp. 229–240.
[4] K. Y. Lee *et al.*, "Design and implementation of MLC NAND flash-based DBMS for mobile devices," *J. Syst. Software*, vol. 82, no. 9, pp. 1447–1458, 2009.
[5] K.-T. Park *et al.*, "A zeroing cell-to-cell interference page architecture with temporary LSB storing and parallel MSB program scheme for MLC NAND flash memories," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 919–928, Apr. 2008.
[6] H.-W. Tseng, L. Grupp, and S. Swanson, "Understanding the impact of power loss on flash memory," in *Proc. DAC'11*, pp. 35–40.