An Integrated Approach for Managing Read Disturbs in High-Density NAND Flash Memory

Keonsoo Ha, Jaeyong Jeong, and Jihong Kim, *Member, IEEE*

***Abstract*—The read-disturb problem is emerging as one of the main reliability issues in high-density NAND flash memory. A read-disturb error, which causes data loss, occurs to a page when a large number of reads are performed to its neighboring pages. In this paper, we propose a novel integrated approach for managing the read-disturb problem. Our approach is based on our key observations from the NAND physics that the read disturbance to neighboring pages is a function of the read voltage and the read time. Since the read disturbance has an exponential dependence on the read voltage, lowering the read voltage can improve the read-disturb resistance of a NAND block. By modifying NAND chips to support multiple read modes with different read voltages, our approach allows a flash translation layer module to exploit the tradeoff between the read disturbance and write speed. Since the read disturbance is also proportional to the read time, our approach exploits the difference in the read time among different NAND pages so that frequently read pages can be less intensively read-disturbed using fast page reads. By intelligently relocating read-intensive data to read-disturb resistant blocks and pages, our approach can reduce a large portion of the time overhead from managing read-disturb errors. We also propose a proactive data migration technique which is effective in reducing large variations in I/O response times of the existing on-demand read reclaim (RR) technique. Our experimental results show that our proposed techniques can reduce the execution time overhead by 73% over the existing read-disturb management technique while reducing I/O response time fluctuations during RR activations.**

***Index Terms*—Data storage systems, flash translation layer, NAND flash memory, read disturb.**

I. Introduction

As NAND flash memory technology scales down to 20-nm and below, data reliability becomes a major design concern for NAND flash-based storage systems. Among various reliability issues such as limited endurance and short data retention, the read-disturb problem is expected to emerge as a major reliability concern for future high-density NAND flash memory. When a page P in a block B is read, a read disturb occurs to P's neighboring pages in the block B. Since cells are serially connected in a string structure in NAND flash memory, the cells in the same string can be unintentionally programmed by a certain level of read voltage (e.g., 7 V [1]) applied whenever their neighboring cells in the same block are read. If neighboring cells are repeatedly disturbed by many reads to the same block, some of the affected cells may lose their data from frequent unintentional programming. If the number of changed bits by read disturbs exceeds the number of recoverable bits by a recovery method such as an error correcting code (ECC), a read-disturb error occurs.

To avoid data corruption by read disturbs, an extra anticipatory defense procedure, commonly called as read reclaim (RR) [2], [3], is required in partially read-disturbed blocks. When an RR technique is used, some predictive measures on the seriousness of read disturbance are applied to predict impending read-disturb errors. Since disturbed blocks can return to their initial undisturbed status when they are erased, the disturbed blocks are erased whenever they experience serious read disturbs. Similar to garbage collection (GC), if a disturbed block contains valid data, the valid data in the disturbed block are copied to a new free block before the disturbed block is erased. Once the disturbed block is erased, the block is fully recovered from the read-disturb problem, completing the RR procedure.

In future high-density NAND flash memory, RRs are expected to occur more frequently. First, as the density of NAND flash memory is increased by advanced process shrink and multileveling techniques, read-disturb resistance has been considerably weakened. Since this decreasing trend in the read-disturb resistance is anticipated to get accelerated in future NAND blocks, more frequent RRs cannot be avoided. Second, more sophisticated reliability improvement techniques for future NAND chips, such as low density parity check (LDPC) codes with the read retry function, introduce a large number of extra reads, thus causing more frequent RRs. When read retrials are necessary, a single host-level read can be amplified to multiple device-level reads [4]. For example, in recent $1\times$ nm-node multi-level cell (MLC) NAND chips which require LDPC codes with the read retry function, a single read from a host system may result in up to seven reads at the NAND device level [5]. Furthermore, emerging read-intensive

Manuscript received February 27, 2015; revised May 25, 2015 and September 10, 2015; accepted October 10, 2015. Date of publication November 30, 2015; date of current version June 16, 2016. This research was supported by Basic Science Research Program and Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2013R1A2A2A01068260 and NRF-2015M3C4A7065645). The ICT at Seoul National University and IDEC provided research facilities for this study. This paper was recommended by Associate Editor L. P. Carloni. *(Corresponding author: Jihong Kim)*

The authors are with the Department of Computer Science and Engineering, Seoul National University, Seoul 151-744, Korea (e-mail: air21c@davinci.snu.ac.kr; jyjeong@davinci.snu.ac.kr; jihong@davinci.snu.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2504868

0278-0070 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

workload will significantly increase the frequency of RRs. For example, in read-intensive solid-state drives (SSDs) with high IOPS [6]–[8] (which were specially designed to support read-dominant workload such as front-end Web servers, video on demand service, and memcached [9]), an efficient management of RRs will be an important SSD design requirement.

Frequent RRs, however, can negatively affect the performance of NAND flash memory-based storage systems due to extra operations performed during RR executions. In particular, the I/O response time may fluctuate significantly depending on whether an RR procedure is activated or not. If a normal I/O request conflicts with an RR procedure, there can be a significant delay in processing the normal I/O request [10]. Such a response time delay can seriously degrade the quality of the service of I/O intensive business applications such as electronic commerce, where consistent response times are regarded as one of the most important storage design constraints [11].

Although read disturbs can cause a serious negative impact on the performance of NAND flash memory, the read-disturb problem has not been extensively investigated compared with other NAND flash reliability issues. Most existing read-disturb management techniques employ a simple reactive solution based on some predictive measures on the severity of read disturbance without considering performance penalty for activating an RR procedure [12]. A straightforward technique uses the number of performed read operations of each block to predict the read-disturbance status of a block. When a block undergoes more read operations than a preset upper bound on the maximum read count of a block, an RR procedure is triggered [13] (we denote this method as `baseline` because it is used as a baseline read-disturb management technique in this paper). In our evaluations of the `baseline` technique, a flash translation layer (FTL) based on the `baseline` technique causes a large number of data migrations, increasing the total execution times for extra operations during RR. Moreover, the reactive data migration policy of the `baseline` technique incurs large variations on I/O response times, because once the degree of read disturbance of a disturbed block is high, the `baseline` technique moves all the valid pages in the disturbed block together, thus significantly increasing the I/O response time.

In order to efficiently manage the read-disturb problem, we propose a novel integrated approach for the read-disturb problem. Our approach is based on our key observation on read-disturb errors from the NAND device physics that the degree of the read disturbance to neighboring pages is a function of the read voltage and the read operation time. Since the degree of the read disturbance of a block is exponentially dependent on the read voltage applied to unselected word lines when a page is read, our proposed technique controls the read voltage to increase read-disturb resistance of NAND block. Another key tradeoff is that the read disturbance has a linear dependence on the read operation time. For the same number of page reads, we can effectively increase the read-disturb resistance of a block by increasing the ratio of fast page reads [e.g., the least significant bit (LSB) pages in a triple-level cell (TLC) NAND chip] to slow page reads [e.g., the most significant bit (MSB) pages in a TLC NAND chip]

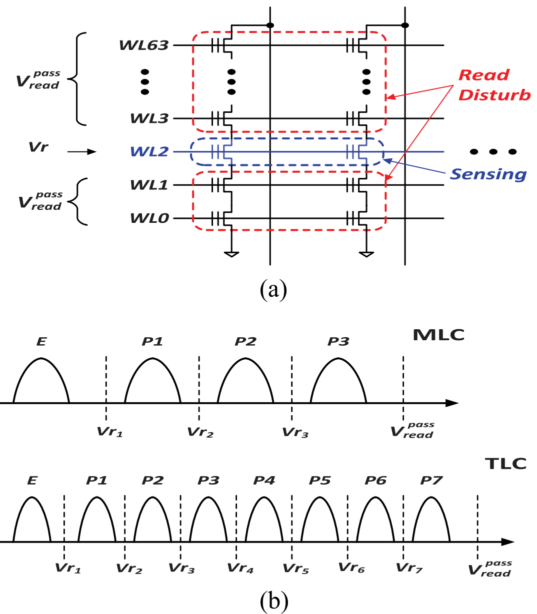

Fig. 1. Relationship among the read reference voltages and read voltage affecting the read disturbance of NAND cells. (a) Voltage settings during a read operation in a NAND flash string. (b) Reference voltages V_{r_i}'s and the read voltages $V_{\text{read}}^{\text{pass}}$'s of MLC and TLC NANDs.

in the block. Finally, in order to reduce large variations in I/O response times of the `baseline` technique, we propose a proactive background data migration technique which moves frequently-read pages in a partially read-disturbed block in advance before the block is seriously read-disturbed.

Based on our proposed read-disturb management techniques, we have developed a read-disturb aware FTL, called `redFTL`, which is a page-level FTL with a new read-disturb manager module. We evaluated the effectiveness of `redFTL` with an extended FlashBench emulation environment [14] which supports multiple read voltage modes from our proposed techniques. The experimental results using six read-intensive benchmark traces show that `redFTL` can reduce the total execution times for extra operations during RR, on average, by 73%. Furthermore, our proactive approach of `redFTL` significantly mitigates the response time fluctuations during RR activations.

The rest of this paper is organized as follows. In Section II, we describe the key NAND device physics behind the read-disturb problem, and evaluate the performance impact of managing read disturbs in NAND-based storage systems. In Section III, our integrated read-disturb management techniques are explained in detail. We describe `redFTL` employing our read-disturb management module in Section IV, and report performance evaluation results using read-intensive benchmark traces in Section V. Section VI summarizes the related work on the read-disturb problem. Finally, Section VII concludes with a summary and future extensions of our techniques.

II. Performance Implications of Read Disturbs

A. Read-Disturb Errors in NAND Flash Memory

In order to understand why read-disturb errors occur in NAND flash memory, we briefly explain how a NAND cell is

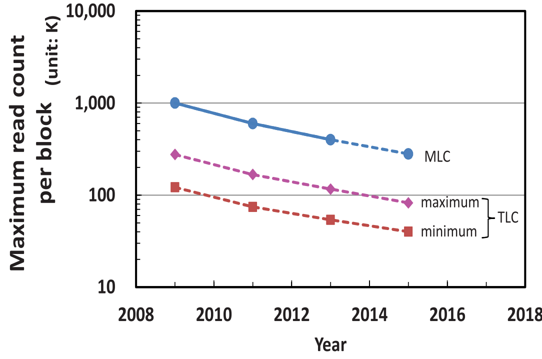

Fig. 2. Expected read-disturb trend for MLC and TLC devices.

read. As shown in Fig. 1(a), NAND memory cells are serially connected to form a string unit which is a basic operational unit. During a read operation, a preset reference voltage V_{r_i} is applied to a selected word line [e.g., WL2 in Fig. 1(a)] for determining the cell state of the selected word line while a read voltage $V_{\text{read}}^{\text{pass}}$ is applied to unselected word lines [e.g., WL0 and WL1, and WL3–WL63 in Fig. 1(a)] to fully turn on the unselected cells in the string (so that the selected cell state can be transferred to a sensing circuit without any interference). A read voltage should be high enough to make the unselected cells transparent during a read operation. Otherwise, the selected cell state cannot be precisely determined because the unselected cells will distort the current read from the selected cell. Fig. 1(b) illustrates how reference voltages and read voltages are positioned at MLC and TLC NANDs, respectively. Since a relatively high read voltage is applied to the unselected cells of the same string during a read operation, the unselected cells may be unintentionally and softly programmed by the high read voltage, resulting in the read-disturb error. Although the effect of a soft program by a single read operation is not big enough to affect the neighboring cell data, if its effect is continuously accumulated by repetitive read operations, the stored contents in memory cells will be eventually altered [1].

Based on the widely-known Fowler–Nordheim (FN)-tunneling equation [15], read-disturb errors can be quantified by the total number N_e of electrons unintentionally injected into a memory cell during repetitive read operations as follows:

$$N_e \propto J_{\text{FN}} \cdot T_S \propto \left\{ \left(V_{\text{read}}^{\text{pass}}\right)^2 \cdot \exp\left[\frac{-1}{V_{\text{read}}^{\text{pass}}}\right] \right\} \cdot \{T_{\text{read}} \cdot N_{\text{read}}\} \quad (1)$$

where J_{FN} represents an FN-tunneling current for a unit time and T_S is the entire stress-time interval, and T_{read} and N_{read} are a read operation time and the total number of read operations, respectively. Since T_S can be modeled as $T_{\text{read}} \times N_{\text{read}}$ and J_{FN} is exponentially proportional to $V_{\text{read}}^{\text{pass}}$, the read-disturb problem gets more severe with a higher read voltage or a longer read operation time.

Since both a read operation time and a read voltage get, respectively, longer and higher as the density of NAND flash memory is increased [16], read-disturb resistance has been significantly weakened. In particular, the read-disturb problem is expected to be more prominent in TLC NAND memory chips due to its high read voltage and slow read operation. Fig. 2 summarizes a future trend in the read-disturb problem. Our estimations are based on a simple approximation using the FN-tunneling equation (in a similar fashion used for forecasting the read disturbance of MLC NAND blocks in [17]). As shown in a dashed line with diamond symbols (i.e., the maximum case) of Fig. 2, the maximum read count of a TLC NAND chip is estimated to be about 28% on average of that of an MLC device (in this paper, we represent the read-disturb resistance of a NAND block by the maximum read count of the NAND block). Since a TLC NAND chip has a higher (e.g., ~5%) read voltage and a longer (e.g., ~60%) read operation time compared to an MLC device, TLC NAND cells are likely to be more read disturbed. Moreover, since the read disturbance of a NAND cell is exponentially intensified with the applied read voltage, the maximum read count of a TLC device is drastically weakened with a higher read voltage. For example, a 5% increase in a read voltage results in a decrease of as much as about 49% in the maximum read count, as shown in a dashed line with square symbols (i.e., the minimum case) of Fig. 2. It means that the maximum read count decreases by at least an order of magnitude over a typical MLC NAND in 2009. Therefore, an RR procedure is likely to be required more frequently in future high-density TLC NAND flash memory.

B. *Effect of Frequent Read Reclaims*

Since most NAND flash manufacturing consider the maximum read count of a NAND block as highly confidential data, it is not easy to how often RRs will occur in real cases. However, we can indirectly estimate the frequency of RRs using NAND technology roadmap data with commercial SSD product specifications. When a large number of highly-skewed read requests come in burst for modern high-IOPS SSDs, we believe that a significant number of RRs can occur. For example, as shown in Fig. 2, the maximum read count of MLC in 2013 was estimated to be around 400 000 [17]. In the same year, an MLC-based SSD provided read-IOPS of up to 1 300 000 [18]. If several thousands of read-dominant clients access such an SSD within a short time period in a skewed fashion to small NAND blocks in the SSD,[1] the MLC blocks in the SSD may suffer from frequent RR activations. For example, if the tpc-e trace (that was used in our experiments) was configured to mimic the read characteristics of modern storage systems, on average, 2316 RRs can be observed within 1 h. Since read-IOPS of server systems and read-disturb resistance of NAND blocks have been increased and decreased, respectively, RRs are likely to be occurred more frequently in the future.

In order to understand how the existing read-disturb management technique works when a large number of RRs are triggered, we evaluated the `baseline` technique [12] using a trace-driven simulator with six read-intensive benchmark traces (for a more detailed explanation of the experimental setup, refer to Section V). Experimental results show that frequent occurrences of RR introduce considerable extra NAND

[1]For example, a significant number of users concurrently read day's top news on a Web page.

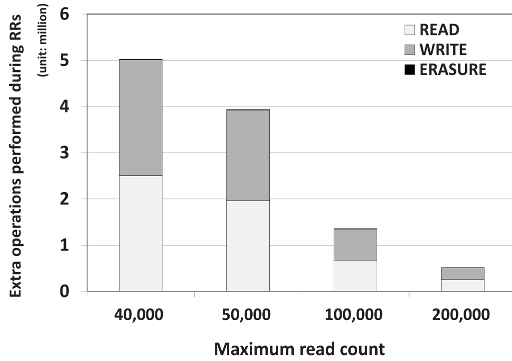

Fig. 3. Breakdown of read, write, and erasure operations from RRs over varying maximum read counts.

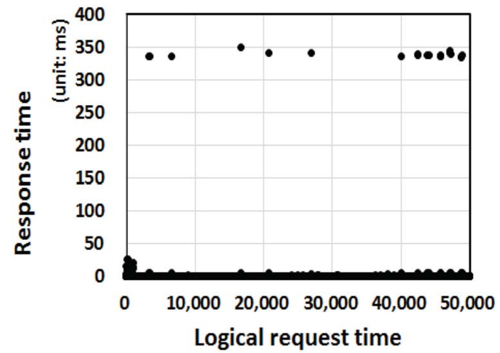

Fig. 4. Snapshot of response time variations when the ads benchmark trace is executed.

operations for data migrations and block erasures. Fig. 3 shows how extra operations during RRs change under different maximum read counts. The *x*-axis denotes various maximum read counts, and the *y*-axis represents the total number of extra operations performed. As shown in Fig. 3, the case when the maximum read count is 40 000, frequent RR activations resulted in about ten times increase in the total extra operations over when the maximum read count is 200 000. In particular, read and write operations for data migrations during RR occupied the majority of the total extra operations. This result indicates that a large number of valid pages exist in disturbed blocks, and the time overhead of migrating these valid pages during RR activations can be considerable if they are simultaneously moved. This long data migration time is the main source of I/O response time fluctuations.

Our evaluation results show that the performance degradation from using the existing simple read-disturb management technique such as `baseline` is so severe that they are not appropriate for high-density NAND flash memory with a small maximum read count. Since the maximum read count will be getting smaller because of the weakened read-disturb resistance of high-density NAND flash memory, in order for high-density NAND flash memory to be widely adopted in various storage products, it is critical to devise a more efficient read-disturb management technique.

C. Effect of Read Reclaims on Response Time Fluctuations

An RR incurs a significant fluctuation of I/O response times because a large number of page mitigations are necessary during an RR procedure. For example, Fig. 4 shows a snapshot of response time variations for the ads [19] benchmark trace when data in disturbed blocks migrate to healthier blocks. The *x*-axis and the *y*-axis represent the logical request time and the read response time, respectively. The logical request time increases by one whenever a read request is performed in NAND flash memory. In Fig. 4, there are several high peaks of the response time which correspond to RRs. Since all the valid pages should be moved to a healthier block simultaneously in order to avoid data corruption by read-disturb errors when an RR procedure is triggered, the time overhead of an RR procedure is directly proportional to the number of valid pages in the disturbed block. The more valid pages exist in a disturbed block, the larger required time to migrate them. Unfortunately, most of blocks involved in an RR procedure are filled with many valid data. For example, in our experiments, we observed that, on average, about 96% of pages in a block are moved during RR procedures when RR threshold is 40 000, thus taking a long time to complete a single RR activation. If an RR activation must move a block with a large number of valid pages, response times can increase up to about 326 ms, about 3260 times increase over the average read operation time of 100 us. If page migrations from an RR procedure can be distributed over a longer period, there could be less response time fluctuations.

TABLE I
READ HIT RATIOS OF VARIOUS SSD READ BUFFERS

Buffer size	Benchmark					
	ads	websearch	tpc-h	multi	tpc-e	tatp
64 MB	1%	0%	27%	36%	9%	10%
128 MB	2%	1%	29%	36%	17%	16%
256 MB	3%	2%	30%	96%	58%	22%

D. Effect of SSD Read Buffer on Read Reclaims

When the read-disturb problem is discussed, one straightforward solution seems to be a large SSD read buffer. If the SSD read buffer is large enough to cover most read requests (which were not served by a page cache in an operating system), it is clear that no read-disturb error is likely to occur. In order to understand the effect of SSD read buffer on RRs, we performed several experiments using an FTL simulator with six read-dominant I/O traces which have different trace times ranging from one hour to four days (for a detailed description of the experimental setup, refer to Section V). Since these I/O traces were collected at the block device level, all I/O requests in the traces are actual I/O requests sent to SSDs from a page cache of a kernel. Table I summarizes read hit ratios under different read buffer sizes (these read buffers are all assumed to be managed by the least recently used scheme). On average, about 78% of read requests are served by NAND flash chips (since the read buffer is likely to be shared by multiple programs at the same time, the read hit ratio may be even lower than our measurement). This low read hit ratio of a large SSD read buffer comes from large working sets of read-intensive traces. Our evaluation using read-intensive workloads

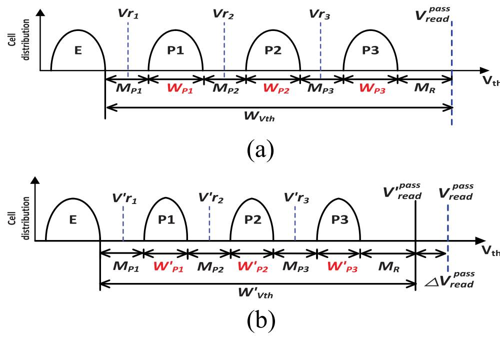

Fig. 5. Example of a read voltage shifting by narrowing the width of a threshold voltage distribution. Threshold voltage distributions when a (a) normal program method is used and (b) modified read-resistant program method is used.

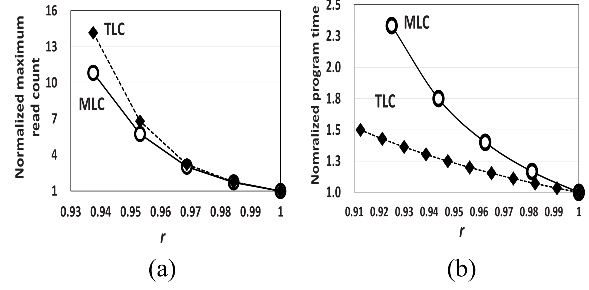

Fig. 6. Effect of lowering the read voltage on the read resistance and program time. (a) Maximum read count variations under different read voltages in MLC and TLC blocks. (b) Program time increases over different read voltages in MLC and TLC blocks.

demonstrates that a large SSD read buffer alone cannot solve the read-disturb problem, thus requiring a different solution.

III. Read Disturb Management Techniques

We propose an integrated read-disturb management approach which is based on tradeoff relationships related to the NAND read disturbance. In this section, we explain the effect of the read voltage scaling on improving the NAND read-disturb resistance in detail and describe how to control the applied read voltage in our proposed technique. We also explain the relationship between the read operation time and the read-disturb problem, and we show the effect of the read operation time scaling on mitigating the read-disturb problem. Finally, we present a NAND read-disturbance model which quantitatively indicates the degree of read disturbance of a block by a read operation under different read voltages and different read times.

A. Mitigation of Read Reclaims by Read Voltage Scaling

Reducing a read voltage is the most effective way of mitigating the effect of the read disturbance because the FN-tunneling effect is exponentially proportional to the applied read voltage. In order to use a lower read voltage when a NAND page is read, it is necessary to form narrow threshold voltage distributions when a program operation is performed. Fig. 5 illustrates how read voltage scaling affects threshold voltage distributions for four-state MLC devices. As shown in Fig. 5, the read voltage level is affected by the width of two voltage margins (the voltage margin Mp_i between two adjacent program states and the voltage margin M_R between the highest program state and the read voltage $V_{\text{read}}^{\text{pass}}$) and the width Wp_i of each program state. Therefore, a lower $V_{\text{read}}^{\text{pass}}$ can be used if we can reduce Mp_i's, M_R, or Wp_i's.

In our proposed read voltage scaling technique, we only control Wp_i's, because changing Mp_i's and M_R may significantly affect the reliability of NAND flash memory. For example, when Mp_i's are determined by a NAND flash manufacturer during the device design time, their exact values are decided so that no data retention error can occur within the data retention requirement. If we reduced Mp_i's without sufficient experimental evaluations of their impact on the data retention time requirement, such reduction of Mp_i's may result in critical data loss. Since such an experimental validation is almost impossible without close collaboration with a NAND flash manufacturer, we decided not to control Wp_i's in our proposed scaling technique. In a similar reason, our proposed technique does not control M_R as well because reducing M_R may cause some pages to be incorrectly read.

Unlike the voltage margins such as Mp_i and M_{Ri} which inflict on the reliability of NAND flash memory, narrowing the width Wp_i of a program state does not negatively affect the reliability. However, narrowing Wp_i degrades the write performance. Therefore, the read voltage must be carefully scaled down so that a performance degradation from narrowing the width Wp_i is acceptable. Since there is a tradeoff relationship between the program time and the width of a threshold voltage distribution [20], we can apply the read-resistant program method when a longer program time is not a major performance problem. In Fig. 5, we can reduce $V_{\text{read}}^{\text{pass}}$ by a total amount of $\Delta V_{\text{read}}^{\text{pass}}$ by reducing each Wp_i by $(\Delta V_{\text{read}}^{\text{pass}}/3)$.

In order to observe the relationship between the read-disturb resistance and the read voltage, we measured the read-disturb resistance of MLC and TLC NAND blocks under different read voltages. In our experiments, we measured the maximum read count of NAND flash memory with recent 20 nm-node NAND chips for quantifying the degree of read disturbance. The maximum read count of each block is determined with the number of read cycles of a block whose number of bits errors exceeds the recoverable bits by a 40-bit ECC. Fig. 6(a) shows how many read requests can be serviced when a read voltage is decreased. In Fig. 6(a), the x-axis denotes the read voltage scaling ratio r, and the y-axis represents the normalized maximum read count which is normalized with those of normal MLC and TLC blocks, respectively. When the read voltage scaling ratio r is set to x, the read voltage is reduced to x% of the normal read voltage. As shown in Fig. 6(a), in both MLC and TLC blocks, the maximum read counts are dramatically increased with lower read voltages. For example, when a read voltage of a TLC chip is decreased by about 4% (denoted as 0.96 in the x-axis), the maximum read count increases by about five times. Our measurement result confirms that read voltage scaling is an effective way of solving the read-disturb problem.

TABLE II
PROPOSED READ MODES OF AN RRB WITH DIFFERENT READ VOLTAGES

read mode	r	increase in mrc	$\Delta\mathbf{T_{prog}}$	write mode used
R_{mode_0}	1.00	× 1	0%	W_{mode_0}
R_{mode_1}	0.98	× 2	8%	W_{mode_1}
R_{mode_2}	0.96	× 5	19%	W_{mode_2}

r: read voltage scaling ratio **mrc**: maximum read count
$\Delta\mathbf{T_{prog}}$: program time increase

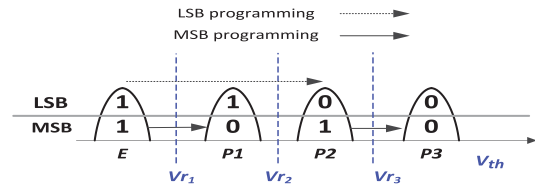

Fig. 7. Bit patterns of program states in an MLC NAND cell.

In addition to the tradeoff between the read voltage and the read resistance, Fig. 6(b) shows the relationship between the program speed and the applied read voltage in MLC and TLC blocks. The *x*-axis denotes the read voltage scaling ratio *r*, and the *y*-axis represents the normalized program time (where the program times of normal MLC and TLC blocks are used as baseline cases). As shown in Fig. 6(b), the program time is increased as the read voltage is reduced. Since the width of threshold voltage distributions of all program states are narrowed by a fine-grained program method, the amount of read voltage reduction is proportional to the number of states of NAND cells. In other words, in order to reduce the read voltage by the same $\Delta V_{\text{read}}^{\text{pass}}$, an MLC chip with three program states needs to shorten Wp_i's more than a TLC chip with seven program states, thus increasing the program time of MLC chips more sharply as *r* gets smaller.

In order to efficiently exploit read voltage scaling for reducing the RR overhead, we introduce a special NAND block, called a read-resistant block (RRB). Unlike a normal block, a proposed RRB can support multiple read modes using different read voltages. An RRB using a lower read voltage is created by forming narrow threshold voltage distributions when a program operation is performed using a fine-grained program control [20]. Table II summarizes key tradeoff relationships among three read modes. R_{mode_0} uses the nominal (i.e., the highest) read voltage while R_{mode_2} uses the lowest read voltage (which is 96% of the nominal read voltage). If an RRB *B* were to be read by R_{mode_2}, it must be first programmed by W_{mode_2} whose program time is 19% longer than a normal block.[2] However, using R_{mode_2} increases the maximum read count of *B* by five times over using R_{mode_0} (since R_{mode_i} can be used only for a block written in W_{mode_i}, where no confusion arises,[3] we use the terms R_{mode_i} and W_{mode_i} interchangeably with mode_i). Although fine-grained programs take longer (over the conventional programs) because of narrow threshold voltage distributions, in our approach, using an RRB does not degrade I/O performance because an RRB is used for only read-intensive workload (in which frequent programs are very unlikely).

[2]We can use a more read-resistant mode (than R_{mode2}) by lowering the read voltage scaling ratio *r* more than 0.96. However, as NAND flash memory technology scales down, it may not be possible to use *r* less than 0.96. For example, as the density of NAND flash memory is increased, the width of threshold voltage distributions are also getting bigger by side effects of scaling down such as cell-to-cell interference [21] and random telegraph noise [22], [23]. Considering the wider width of threshold voltage distributions, only a small voltage range is available for read voltage reduction, thus limiting the maximum allowed read voltage scaling ratio. In this paper, we limit *r* by 0.96.

[3]If pages in different blocks are programmed using different write modes, proper read voltages must be used when pages are read. If sensing operations are performed with inadequate reference voltages, data may not be read at all or a large number of read retries may be necessary, thus degrading both read reliability and performance. Therefore, blocks must be read and written by the same modes.

B. Mitigation of Read Reclaims by Read Operation Time Scaling

The read operation time is another important factor affecting read-disturb errors because the FN-tunneling effect is linearly proportional to the read operation time. Since a read operation time is fixed during the device design time, however, we cannot easily change the read operation time at the S/W level. Alternatively, we can exploit different read times among different page types. In an MLC NAND flash block, two types of pages share memory cells in the same word line. The two pages are called the LSB and the MSB pages, respectively. [In case of a TLC NAND, there is the central significant bit (CSB) page between LSB and MSB pages.] As these names indicate, each page only uses its own bit position of a bit pattern stored in each cell. For example, as shown in Fig. 7, a memory cell in an erased state is interpreted as a logical "11." When the MSB position of the cell is programmed to a logical "0," the bit pattern in the cell will be changed to "01."

In order to classify the cell state in an MLC NAND cell, different number of voltage sensing operations are required in accordance with the page types to be read. For example, in Fig. 7, in order to read an LSB page, only one reference voltage (i.e., V_{r_2}) is used while two reference voltages (i.e., V_{r_1} and V_{r_3}) are necessary for reading an MSB page. Since a read operation time is linearly proportional to the number of voltage sensing operations, the read operation time for an MSB page is twice longer than that for reading an LSB page. As a result, the degree of the read disturbance from reading an MSB page is about twice higher than that from reading an LSB page. For a TLC chip, there is a similar imbalance of read operation times depending on its page types such as LSB, CSB, and MSB [24]–[26]. Such an imbalance of read operation times among different page types suggests that the total read operation time can be effectively reduced by intelligently exploiting the operation time differences among different page types.

In order to observe the relationship between the read-disturb resistance and the read operation time, we measured the read-disturb resistance of reading different page types in an MLC block and a TLC block using 20 nm-node NAND chips. Fig. 8 shows how different types of page reads can affect the maximum read count. The *y*-axis denotes the normalized maximum read count under different page access behaviors. Since the

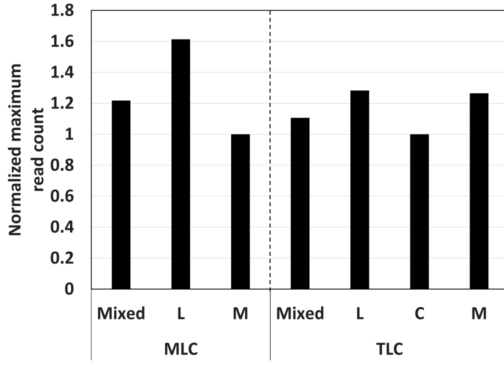

Fig. 8. Maximum read counts under different page types of MLC and TLC blocks.

maximum read count of a block is determined as the worst case where the slowest operations are always performed until the block causes read-disturb errors, the maximum read count is normalized with the maximum read count of the slowest page type. The *x*-axis represents different read workloads for MLC and TLC NAND chips. For example, L/C/M in the *x*-axis indicates the case where only LSB/CSB/MSB pages in blocks were read, respectively. On the other hand, Mixed represents the case where different page types were evenly read.

For MLC chips, L increases the maximum read count by 61% over M. Although the number of sensing operations for reading an LSB page is half of that for reading an MSB page, L cannot enhance read-disturb resistance twice over M because extra steps for reading a cell such as setting reference and read voltages, and decoding target address are performed together in a page operation. For TLC chips we tested, reading a CSB page took about 50% longer over reading an LSB page or MSB page. Our evaluation result roughly agrees with the difference in the read time, increasing the maximum read count of L and M by 28% and 26%, respectively, over C.[4] For different TLC chips where reading an MSB page takes longer than a CSB page or an LSB page [25], we expect a similar difference in the maximum read count of a block depending on read workloads.

In this paper, when a page supports a fast read over other pages, we call such pages read-resistant pages (RRPs), because a block can serve more read requests when such pages are more frequently read than slow pages. In order to reduce the read disturbance of a block, we can effectively reduce the performed average read time by inducing more frequently-read data (i.e., read-hot) to be located in RRPs.

[4]The performance imbalance among page types in a TLC NAND block is different according to TLC chip types. The speed of reading each page type is determined by the number of sensing operations necessary for each page type. The number of sensing operations for each page type depends on the method of setting a bit pattern of each program state. In a typical TLC chip, the gray code sequence has been used for mapping program states to bit patterns [27]. For example, “111,” “011,” “001,” “101,” “100,” “000,” “010,” and “110” are mapped from the erased state to the seventh program state, respectively. For this mapping, one, two, and four sensing operations are necessary in order to read LSB, CSB, and MSB pages, respectively. In the TLC chips used for our measurement study, a different bit pattern sequence was used, requiring two, three, and two sensing operations for reading LSB, CSB, and MSB pages, respectively.

TABLE III
SUMMARY OF THE KEY PARAMETERS OF OUR PROPOSED NAND READ-DISTURBANCE MODELS

model	β_{MSB}	β_{CSB}	β_{LSB}	α_0	α_1	α_2
TLC_{base}	1	0.5	0.25	1	2	5
TLC_{opt}	0.78	1	0.79			

C. NAND *Read-Disturbance Model*

Combining the effect of read voltage scaling and read operation time scaling on the read disturbance of NAND blocks, we developed a novel read-disturbance model that can be used with our proposed RRBs and RRPs. Since the read disturbance of a block from a single read varies depending on the applied read voltage and the type of a page read, we introduce a new read disturbance metric, called effective read disturbance per read (in short, effective read disturbance), which indicates the effective degree of the read disturbance to the NAND block after a specific read. When a type *j* page is read by using R_{mode_i}, the effective read disturbance is given as (β_j/α_i), where α_i $(0 \le \text{i} \le 2)$ denotes a normalized maximum read count (over the maximum read count of a normal block) when R_{mode_i} is used and β_j indicates the effective read disturbance to a block when the type *j* page is read. For example, in case of the TLC chips we tested in Section III-B, when an LSB page of a TLC block *B* is read by R_{mode_2}, the block *B* can service five times more read requests because of the low read voltage of R_{mode_2}. Furthermore, the block *B* can be read more by 28% [i.e., $(1/0.78)$] due to the fast LSB page. In other words, the block *B* is effectively read-disturbed by 0.156 compared with a normal read operation to a CSB page. In other words, reading an LSB page of an RRB results in about eight times increase in the available read operations over when a CSB of a normal block is read. When all the read operations (using different combinations of a read voltage and a read page type) to a block *B* are considered, we can represent the total sum of effective read disturbance as follows:

$$\sum_{i \in \text{RMODE}} \sum_{j \in \text{PTYPE}} \frac{\beta_j}{\alpha_i} \cdot n_{(i,j)} \tag{2}$$

where RMODE $= \{0, 1, 2\}$ represents a set of supported read modes, PTYPE $= \{\text{LSB}, \text{MSB}, \text{CSB}\}$ indicates a set of supported page types, and $n_{(i,j)}$ denotes the total number of the type *j* page reads with R_{mode_i}.

In this paper, we developed two NAND read-disturbance models, TLC_{base} and TLC_{opt}, for two different NAND TLC chips[4]. The TLC_{base} model represents a typical TLC design [26] while the TLC_{opt} model is based on the TLC chips used in our measurement study. TLC chips used in building the TLC_{opt} model are believed to have improved a typical TLC design in several directions including a better handling of various reliability issues [25]. Table III summarizes the key parameters of TLC_{base} and TLC_{opt} which are necessary in computing our proposed effective read disturbance. Because of the different bit pattern sequences between TLC_{base} and TLC_{opt}, they have different β_j's. TLC_{opt} has small differences among its β_i's while TLC_{base} has a larger difference

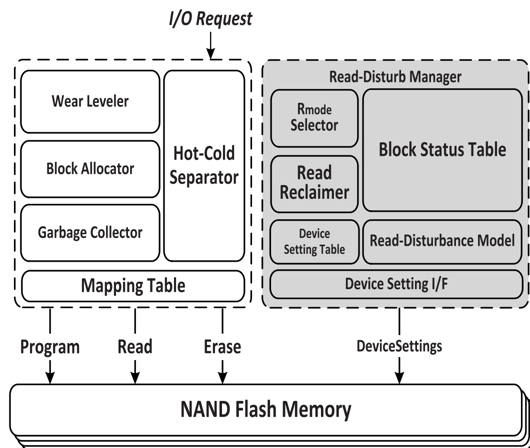

Fig. 9. Organizational overview of `redFTL` with a read-disturb manager.

among its β_i's. Using TLC_{base} and TLC_{opt}, we can estimate the effective impact of each read on the NAND read disturbance.

IV. Design and Implementation of RedFTL

A. Overview

Based on our read-disturb management techniques described in Sections III-A and III-B, and our new NAND read-disturbance model presented in Section III-C, we have implemented `redFTL`. Fig. 9 shows an organizational overview of `redFTL`. The read-disturb manager, which is a key module of `redFTL`, implements our proposed read-disturb management techniques based on our NAND read-disturbance model. For an incoming read request to a block B_{src}, the sum of effective read disturbance is calculated based on our proposed read-disturbance model with total read counts of each page types, and the read reclaimer checks whether the block B_{src} requires an RR or not based on the sum of effective read disturbance of the block B_{src}. If the sum of effective read disturbance of the block B_{src} reaches the preset proactive RR threshold, the read reclaimer finds a target free block B_{tgt} from the block allocator. The read reclaimer copies the valid pages in B_{src} to B_{tgt} by a background thread, thus mitigating fluctuations of I/O response times. Before copying the first valid data in B_{src}, a read mode R_{mode_i} for the block B_{tgt} is determined by the read mode selector submodule. Since a block written by a write mode W_{mode_i} can be read only by the read mode R_{mode_i}, the selected mode information for B_{tgt} is maintained in the block status table. This table is also consulted to decide the mode_i of a block when the block is read or written. Once the write mode is decided, `redFTL` configures a NAND chip to be accessed using the selected mode_i through a new interface DeviceSettings between `redFTL` and NAND chips. `RedFTL` sends proper NAND chip configuration parameters (such as the reference voltages V_{r_i}'s and read voltage $V_{\text{read}}^{\text{pass}}$ for mode_i) through the DeviceSettings interface to NAND chips.

In order to support our proposed multiple read and write modes, the existing NAND chip architecture should be modified so that different read voltages can be applied to wordlines of a NAND flash block. Fig. 10 gives a high-level overview of our proposed new NAND chip architecture. Before a read voltage is applied to the wordlines in the NAND chip to perform read operation, the default read voltage is determined by an internal setting register and a high voltage generator. The internal setting register stores predetermined digital code values, and the high voltage generator tunes a read voltage according to the digital code values by converting them to analog voltages. In order to change the read voltage, a new read voltage is passed to an external setting register using appropriate digital code values.

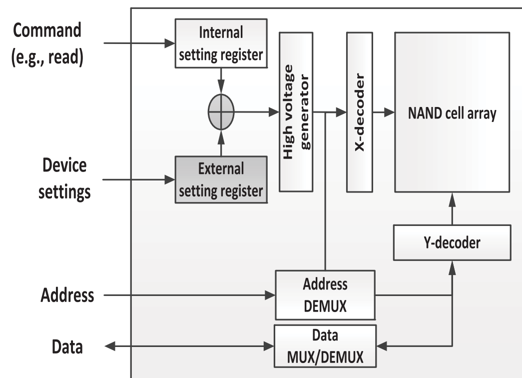

Fig. 10. Overall block diagram of our proposed new NAND chip architecture.

B. Dynamic Mode Selection

`RedFTL` dynamically uses different write modes by changing the block types between a normal block and an RRB according to workload characteristics. Before the first RR is activated for a given block, all write requests to the block are written using the normal write mode W_{mode_0}. When the first RR is activated by intensive read requests to a normal block, the valid data of the disturbed normal block are copied to an RRB using W_{mode_1}. Although different W_{mode_i}'s can be used for copying the valid data to the RRB, our current heuristic chooses W_{mode_1} for the first RR activation because using W_{mode_1} can increase the maximum read count of a block by about 100% with the smallest increase in the program time. For subsequent RR activations from the RRB, we gradually employ the more read-resistant write mode, W_{mode_2}, so that future RR activations can be avoided in a more aggressive fashion.

Valid pages stored in an RRB are moved back to a normal block during GC. When an RRB is selected as a GC victim, the valid data of the RRB are copied using W_{mode_0} (that is, written back to a normal block). Since most GC techniques tend to choose a block with a large number of invalid pages as a GC victim block, if the RRB is selected as a GC victim, it is very likely that many pages in the RRB have been already invalidated. Therefore, when we copy the valid data of the RRB during GC, W_{mode_0} is a logical choice because it is less likely that a small number of valid pages (moved to a free block after GC) will cause another RR.

C. Distributed Migration to RRBs

In addition to the adoption of an RRB, `redFTL` reduces read disturbance of NAND flash blocks by distributing read-hot

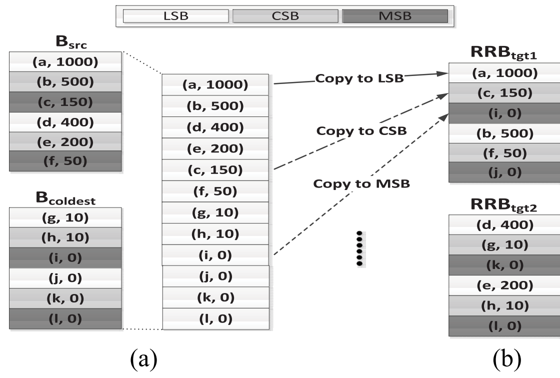

Fig. 11. Example of data sorting and migration in `redFTL`. (a) Data are sorted based on the their access counts in memory. (b) More frequently read data are copied to more faster pages.

data in the partially disturbed block B_{src} to RRPs in two target RRBs. If all the data in the block B_{src} are simply moved together to another block B_{tgt}, the moved data may cause additional RRs from the block B_{tgt} because the same read access patterns may be observed from B_{tgt} as B_{src}. In order to avoid such successive RR activations, `redFTL` distributes the read-hot data of B_{src} to two target RRBs, thus effectively mixing the block B_{src} with other read-cold data during an RR procedure. When the disturbed block B_{src} in the NAND chip C_n incurs an RR, `redFTL` mixes the read-hot data of B_{src} with the data in the least disturbed block $B_{coldest}$ of the NAND chip C_n. Since $B_{coldest}$ is the block with the minimum read count in C_n, data in $B_{coldest}$ are the read-coldest data in C_n. Data in the blocks B_{src} and $B_{coldest}$ are copied to two healthy free blocks RRB_{tgt1} and RRB_{tgt2} in a mixed fashion.

In order to exploit RRPs more effectively, `redFTL` further distinguishes the read-hot data of B_{src} so that hotter data can be moved to more RRPs of a target block. When a block B becomes a candidate block B_{src} of an RR procedure, `RedFTL` sorts pages in the blocks B_{src} and $B_{coldest}$ into a descending order based on their read access counts, and the page order is maintained in memory. According to the page order, `redFTL` copies the sorted pages to target blocks in the LSB–CSB–MSB order (for TLC NAND chips) or the LSB–MSB order (for MLC NAND chips). Fig. 11 shows how `redFTL` copies data in the blocks B_{src} and $B_{coldest}$ to the target blocks RRB_{tgt1} and RRB_{tgt2}. In Fig. 11, a block consists of six pages, and the tuple (d_B, t_B) in the rectangle represents that data d in a block B have been read t times since the block B has been erased. As shown in Fig. 11(a), data in each page in the blocks B_{src} and $B_{coldest}$ are reordered based on the read count of each page. Once the pages are sorted, the sorted pages are copied to target blocks in the LSB–CSB–MSB order during an RR procedure. For example, the frequently accessed read-hot data a, b, d, and e are copied to the fast LSB pages in target blocks RRB_{tgt1} and RRB_{tgt2}, while the read-cold data i, j, k, and l are moved to the slow MSB pages in target blocks.

D. Read-Hotness Detection

When an RR procedure is activated, `redFTL` determines the read hotness of data in a disturbed block based on their read access history. In devising our read-hotness detection techniques, we have exploited a strong read locality of read-dominant workload. As shown in Fig. 12, there exists a small set of dominant logical block addresses (LBAs) that are read very frequently. In Fig. 12, the x-axis represents the logical request time which increments by one whenever a read request is performed, and the y-axis denotes LBAs read at that time. As shown in Fig. 12, most read requests are repeatedly performed on similar LBAs. Based on the strong read locality of read-dominant workloads, `redFTL` can decide the hotness of each data by comparing the total number of reads to each data.

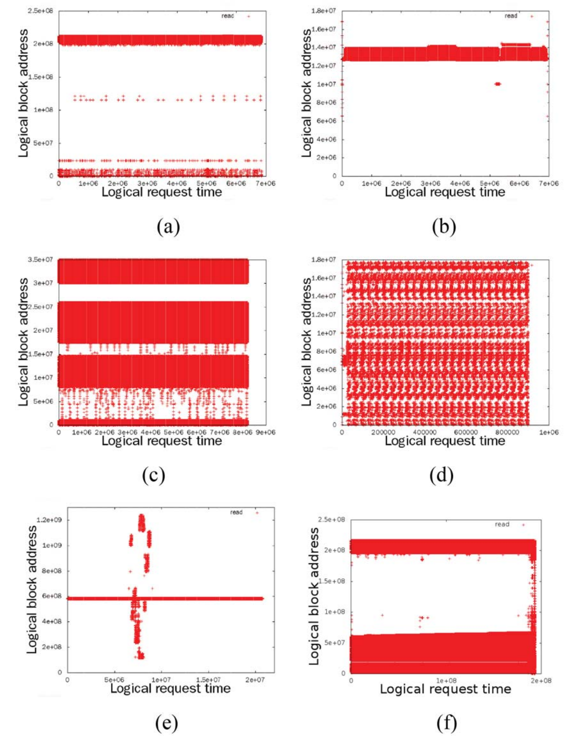

Fig. 12. Distributions of read requests in some benchmark programs. (a) ads. (b) tpc-h. (c) Websearch. (d) Multi. (e) tpc-e. (f) tatp.

However, if `redFTL` keeps track of exact read counts for the entire LBA range, a memory overhead of storing read counts can be very large (close to the same order of magnitude of implementing the page-level mapping table). In order to reduce the memory requirement, `redFTL` employs a two-level counter organization in maintaining read counts of LBAs, which requires one extra byte for each LBA in our current implementation. As explained in Fig. 12, since read-hot LBAs of read-dominant workload have a strong read locality, we can easily distinguish read-hot LBAs from read-cold LBAs by partially counting the number of read accesses to LBAs. For an efficient partial counting, we use a 4-bit saturating counter as the first-level counter. A 4-bit saturating counter increments by one each time its LBA is read, but its counter value saturates at 15. Whenever the total read count of a block B reaches $i \times$ (maximum read count/15) $(1 \leq i \leq 15)$, `redFTL` checks if the first-level counter for a valid page in the block B was saturated or not. If the first-level counter was saturated to

its upper-bound value, the second-level 4-bit counter is incremented by one. Once this sampling is completed, the first-level counters are all reset to zero. When an RR procedure is triggered for the block B, redFTL can classify the hotness of each page based on the second-level counter values. In our current implementation, redFTL sorts pages in the blocks B and $B_{coldest}$ into a descending order based on the second-level counter values. Pages with larger second-level counter values are copied to more RRPs (e.g., the LSB page type in TLC chips) of target blocks. Once an RR procedure is completed and the block B is erased, the two-level counters in the block B are reset to zero.

E. Overhead Analysis

1) Data Migration: During an RR execution in a NAND chip, redFTL selects the most disturbed block and the least disturbed block in the chip as the source blocks of an RR execution by comparing read hotness information of blocks in the chip. The time complexity for finding the source blocks is $O(N)$, where N is the number of clean blocks in the chip. In redFTL, however, the performance overhead of finding the source blocks is negligible because it is performed by a background thread during an idle time (before an RR is invoked). After deciding the source blocks for an RR, redFTL chooses some free blocks as the target blocks of the RR execution. Like a legacy FTL which maintains a free block list for GC and wear leveling (WL), redFTL keeps a similar list of free blocks for GC, WL, and RR. Since redFTL selects a free block from the free block list, the target free blocks of an RR (e.g., RRB_{tgt1} and RRB_{tgt2} in Fig. 11) can be obtained with $O(1)$ time complexity.

2) Page Rearrangement: As we explained in Section IV-C, redFTL rearranges the pages in the source blocks of an RR execution based on their read hotness information, and copies them to the target blocks. Since sorting read-hotness data of the pages in the source blocks is done in memory and it is executed only once when an RR is triggered in a block, the performance and memory overhead of the sorting read-hotness data is negligible. In order to maintain the read-hotness information of each page, redFTL maintains one two-level counter per each page as explained in Section IV-D, in the current redFTL, each two-level counter requires one byte for storing the page access history. Since the two-level counter is simply increased when a read operation is performed, the computation overhead for maintaining the read access information is negligible.

3) Write Mode Management: Since blocks can have different write modes in redFTL, redFTL needs to maintain the write mode of each block. In our current design, the write mode of each block is stored in the block status table (as shown in Fig. 9). Since redFTL supports three write modes, we use two bits for storing the write mode of a block. In order for redFTL to access a page in a block B, a NAND controller needs to know the right write mode for accessing the block B. As shown in Fig. 10, redFTL sends the write mode information to the NAND controller by setting the bits of the external setting register. The high voltage generator changes its output voltage by combining values of both internal and external setting registers. Since setting a register value and combining two register values can be done very quickly with a negligible resource overhead, our proposed NAND chip architecture can provide multiple read and write modes without significant time and area overheads.

4) Write Performance: As we described in Section IV-B, slow write modes are used in redFTL only when valid data in a disturbed block are copied to other healthier blocks during an RR procedure. If a write request arrives from the host system while an RR is activated, redFTL writes the requested data to one of the current active blocks (which are reserved for storing incoming writes). Since the write to the selected active block is programmed using a normal write method, there is no write performance penalty when write-intensive (or mixed) workloads are serviced.

TABLE IV
SUMMARY OF BENCHMARK TRACES

Benchmark	Description	Read (%)	Trace interval	RAF
ads [19]	A display ads platform	96	1 day	100
tpc-h [28]	Accesses to a database	92	10 hours	400
websearch [29]	A search engine	100	4 days	100
multi [28]	Cscope and gcc	98	40 mins	50
tpc-e [30]	Accesses to a database	95	10 hours	30
tatp [31]	Telecom application process	83	21 hours	7

V. EXPERIMENTAL RESULTS

To evaluate the effectiveness of redFTL, we have used a co-simulation environment, FlashBench++, for flash-based storage devices. FlashBench++, which is based on the existing unified development environment, FlashBench [14], was designed to seamlessly switch between two simulation modes, the speed mode and the accuracy mode, depending on an evaluation goal. In the speed mode, FlashBench++ runs fast on top of a functional NAND simulation model whose timing behavior is not accurate. When a high precision of timing accuracy is necessary, FlashBench++ runs in the accuracy mode on top of a timing accurate NAND emulation model although it runs slow. Since it takes a long time to activate RRs in the accuracy mode, we accelerate our evaluation by running FlashBench++ in the speed mode until a large number of reads are performed. By using an execution snapshot function of FlashBench++, we then switch to the accuracy mode so that we can collect a detailed timing accurate evaluation data.

In our evaluation, each block was assumed to consist of 192 pages and the size of a page is set to 8 kB. The total number of blocks in NAND flash memory was set to seven times of the working set size of each benchmark trace. The latencies of read, program, and erase operations for a normal block were set to 100 μs, 1600 μs, and 5 ms, respectively, reflecting recent TLC chips [24], [26]. We used TLC_{base} model as NAND read-disturbance model, which is most commonly used type of TLC. The maximum read count of a normal block is set to 40 000 based on the estimated value shown in Fig. 2, and the proactive RR threshold value is set to 38 000, which is 95% of 40 000.

We used highly read-dominant benchmark traces whose characteristics are summarized in Table IV. The trace interval in Table IV means the length of the time interval during which a corresponding trace was collected. In order to evaluate `redFTL` under realistic conditions, we synthetically generated new traces which better reflect real-world read-disturb cases using traces in Table IV. Since the original benchmark traces listed in Table IV were collected from HDD-based storage systems, they cannot accurately reflect the characteristics of recent SSD-based storage systems. For example, recent SSDs support at least several thousand times higher IOPS over HDDs [32]. Furthermore, thanks to high-speed network connections, such SSDs can support much higher number of concurrent clients.

To reflect these changes in storage systems, we increase the number of read requests in a new trace significantly. In order to concisely represent an increase in the number of reads in SSD-based storage systems over that of HDD-based storage systems, we defined a read amplification factor (RAF) for each trace that indicates how many times the number of read requests in the original trace should be increased for better representing read workload of SSD-based modern storage systems. Depending on an RAF value N of a trace B, we allocated the same trace B to N different I/O generation threads, and made them to generate I/O requests of the trace B. In order to mimic real-world users with various temporal I/O access patterns, however, each I/O thread varied randomly idle intervals between successive I/O requests. In Table IV, the RAF column represents this RAF for each trace. RAF values were set differently according to the characteristics of original traces.

In addition to `baseline` and `redFTL`, we evaluated three more techniques, `RRB`, `RRP`, and `redFTL+`. Both `RRB` and `RRP` work in the same way as `baseline` except two aspects, one that how pages are migrated during RR activations and the other on the read-disturbance model used. In RRB, when an RR is activated, the pages in the read-disturbed block are moved to an RRB using the dynamic mode selection policy (in Section IV-B). On the other hand, in `RRP`, when an RR is invoked, the pages in the read-disturbed block are moved to a normal block. However, during page copies, `RRP` considers the read-hotness of each page so that more frequently-read pages can be moved to more RRPs. In `RRB`, the read-disturbance differences among different page types are not considered. On the other hand, in `RRP`, as explained in Section III-C, different page types incur different degrees of read disturbance although all the blocks are assumed to be normal blocks. `RedFTL+` works in the same way as `redFTL` except that it maintains exact (but expensive) read counts for all LBAs while `redFTL` uses approximate two-level counters.

Fig. 13 shows how, during RR activations, the execution times for data migrations and block erasures change under different read-disturb management techniques. In Fig. 13, the x-axis indicates benchmark traces and applied techniques, and the y-axis represents the normalized execution time for RRs. As shown in Fig. 13, `RRP`, `RRB`, and `redFTL` decreased the total RR execution time, on average, by 53%, 60%, and 73% over `baseline`, respectively. All the values are normalized over `baseline`. In our experiment, `RRB` outperformed `RRP` because the most RRBs (e.g., ones programmed with W_{mode_2}) in `RRB` can be read up to five times more than a normal block in `RRP`; a normal block in `RRP` can be read on average 2.3 times more than a normal block. As expected, `redFTL` outperformed both `RRP` and `RRB` because it takes advantages of both RRBs and RRPs at the same time.[5] Fig. 13 also shows that the approximate read counters used for `redFTL` work as good as the expensive exact counters of `redFTL+`. As shown in Fig. 12, since data tend to maintain their read-hotness for long times, `redFTL`'s approximate counting is sufficient for read-hot data separation.

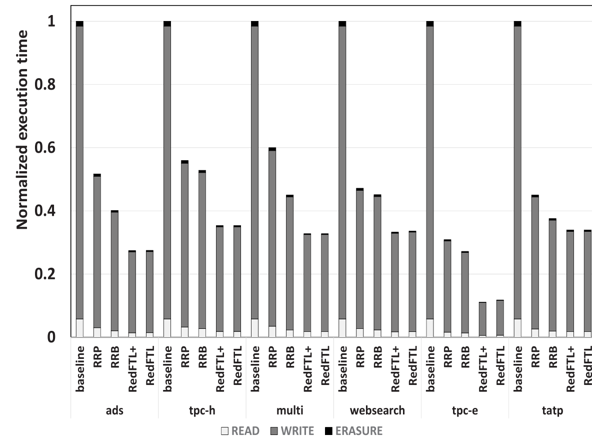

Fig. 13. Normalized execution times for RRs over different techniques.

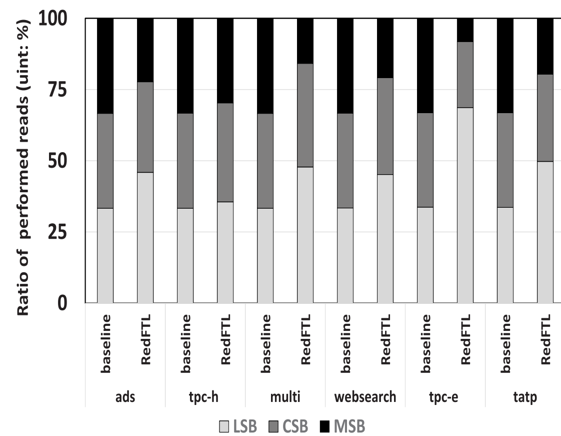

Fig. 14. Breakdown of page type usages in `redFTL`.

Fig. 14 shows how effective our proposed page relocation policy is during an RR procedure. The x-axis represents benchmark traces and applied techniques, and the y-axis indicates the ratio of performed reads for each page type. As shown in Fig. 14, LSB and CSB pages, which are more read resistant than an MSB page, were read more frequently than MSB pages in `redFTL`. Since the page allocation policy in `redFTL`

[5] Note that `redFTL` mixes the valid pages of two source blocks to two free blocks during the distributed migration step for mitigating the skewness of read-hot data, thus increasing the number of extra copies and block erasures. As a result, the execution time of `redFTL` includes the execution time of these extra copies. Although such extra copies may incur the performance overhead, distributing read-hot data to multiple free blocks is helpful for mitigating successive RRs.

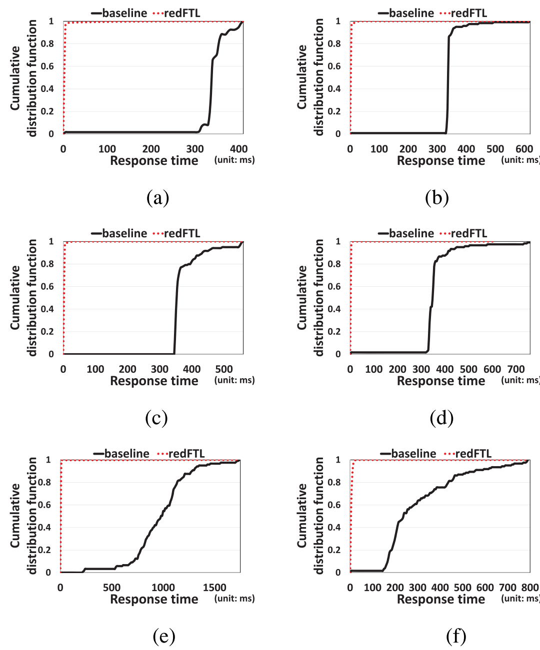

Fig. 15. CDFs of read response times under `redFTL` and `baseline`. (a) ads. (b) tpc-h. (c) Websearch. (d) Multi. (e) tpc-e. (f) tatp.

allocates fast LSB pages and slow MSB pages to read-hot and read-cold data during an RR procedure, respectively, the percentage of LSB-page reads is increased, on average, by about 43% over the `baseline` technique By making more reads from the RRPs, `redFTL` can reduce RR activations.

In order to investigate the effect of our proactive RR approach, we observed the response time fluctuations when RR procedures are activated (using the accuracy mode in FlashBench++). Fig. 15 shows the cumulative distribution functions (CDFs) of read response times while 120 RRs are activated for six benchmarks. As shown in Fig. 15, the read response times of most reads are less than 5 ms in `redFTL`. On average, `redFTL` reduced the read response time during 120 RR activations by 150 times over the `baseline` technique. `RedFTL` spreads performance overhead for a long time period, thus dramatically reducing the read response time. On the other hand, the `baseline` technique does not activate an RR until the number of reads reaches the maximum read threshold. When the RR is activated, a large number of valid pages in the disturbed block are simultaneously moved. As a result, the response times of most reads in the `baseline` technique are over 350 ms.

VI. Related Work

As the read resistance of a NAND block is quickly decreased in a high-density NAND flash memory, several researchers have investigated to mitigate the effect of the read-disturb problem in various design levels [1], [10], [33]–[36]. Jung and Kandemir [10] reported that both program/erase (P/E) cycles and read latency of modern SSDs can be negatively affected by the read-disturb management overhead when read-intensive workloads are serviced. Although this paper demonstrated the effect of read-disturb problem on the performance and endurance of modern SSDs using actual measurement study, they did not propose a solution for the read-disturb problem.

Kang *et al.* [1] proposed a device-level technique that can reduce the effective read disturbance of a read operation by adding an extra dummy cell to a NAND cell string. Although this technique can be useful for mitigating the read-disturb problem, its practical applicability is very limited because it requires many expensive design modifications in the current NAND device design. Cai *et al.* [34] proposed another device-level technique which exploits the error-correction capability of ECC in reducing the read voltage level. When most NAND blocks are healthy, this technique can be very useful in mitigating the read-disturb problem. However, as P/E cycles of NAND blocks increase, it is more likely that ECC cannot recover read-disturb errors from the reduced read voltage, thus limiting the practical applicability of this technique. Our proposed technique is different from these device-level techniques in that our technique is based on an integrated approach exploiting both NAND device physics as well as read-access patterns (such as read skewness) while these techniques focus on the NAND device physics for mitigating the read-disturb problem.

Ha *et al.* [35] and Liu *et al.* [36] presented software-centered read-disturb management techniques for managing the read-disturb problem. Unlike our integrated approach, their techniques focus on mitigating the skewness of read accesses at the FTL S/W level. They convert highly skewed read accesses to a small number of blocks into more balanced read accesses to a large number of blocks by changing data block locations accessed, thus reducing the occurrences of RRs. Our proposed approach is, however, fundamentally different from their techniques in that we exploit the NAND device physics on the read-disturb problem so that we treat each read differently based on its read voltage and read operation time.

VII. Conclusion

We have proposed a novel integrated approach for managing the read-disturb problem in high-density NAND flash memory. Based on read voltage scaling and read time scaling, we introduced RRBs at the device level, and developed a concept of RRPs. By intelligently exploiting RRBs and RRPs at the FTL level, we could dramatically reduce the overhead from frequent RRs. Experimental results show that `redFTL`, which implements the proposed read-disturb management techniques, can reduce the execution time for RRs by 73% over an existing read-disturb management technique. Moreover, I/O response time fluctuations during RR activations are significantly reduced in `redFTL`. As a future work, in order to validate the expected benefit of `redFTL` in a practical setting, we plan to implement and evaluate `redFTL` on a real storage system based on high-density NAND flash memory.

References

[1] M. Kang *et al.*, "Improving read disturb characteristics by self-boosting read scheme for multilevel NAND flash memories," *Jpn. J. Appl. Phys.*, vol. 48, no. 4, 2009, Art. ID 04C062.

[2] Y. Seo, J. Yun, W. Lee, and D. Jung, "Memory controller, method of operating the same and memory system including the same," U.S. Patent 14/081 371, Nov. 2013.

[3] N. Kim and J. Jang, "Nonvolatile memory device, method of operating nonvolatile memory device and memory system including nonvolatile memory device," U.S. Patent 8 203 881, Jun. 2012.

[4] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling," in *Proc. Conf. Design Autom. Test Europe*, Grenoble, France, 2013, pp. 1285–1290.

[5] K. Zhao *et al.*, "LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives," in *Proc. 11th USENIX Conf. File Stor. Technol.*, San Jose, CA, USA, 2013, pp. 243–256.

[6] (2012). *SSDs for Data Centers: The Need for Speed*. [Online]. Available: http://www.samsung.com/global/business/semiconductor/file/media/PM843_Brochure-0.pdf

[7] (2013). *Read Intensive Applications Have Met Their Match in New TOSHIBA Enterprise SSD*. [Online]. Available: https://www.sdd.toshiba.com.tw/english/products/features.aspx?sid=95

[8] (2013). *Lightning Read-Intensive 6Gb/s SAS Enterprise Solid State Drives*. [Online]. Available: http://www.sandisk.com/assets/docs/SanDiskESS-ReadIntensive-ds.pdf

[9] (2014). *Memcached*. [Online]. Available: http://memcached.org

[10] M. Jung and M. Kandemir, "Revisiting widely held SSD expectations and rethinking system-level implications," in *Proc. ACM SIGMETRICS Perform. Eval. Rev.*, Pittsburgh, PA, USA, 2013, pp. 203–216.

[11] Q. Wang *et al.*, "Response time reliability in cloud environments: An empirical study of n-tier applications at high resource utilization," in *Proc. IEEE Symp. Rel. Distrib. Syst.*, Irvine, CA, USA, 2012, pp. 378–383.

[12] M. Phil and A. D. Sena, "Reducing NAND defects and failures with micron NFTL," Micron Software Article Archive, 2011. [Online]. Available: https://www.micron.com/∼/media/Documents/Products/Software%20Article/SWNL_reduce_nand_defects_with_nftl.pdf

[13] H. H. Frost, C. J. Camp, T. J. Fisher, J. A. Fuxa, and L. W. Shelton, "Efficient reduction of read disturb errors in NAND flash memory," U.S. Patent 7 818 525, Oct. 2010.

[14] S. Lee, J. Park, and J. Kim, "FlashBench: A workbench for a rapid development of flash-based storage devices," in *Proc. IEEE Int. Symp. Rapid Syst. Prot.*, Tampere, Finland, 2012, pp. 163–169.

[15] M. Lenzlinger and E. H. Snow, "Fowler-Nordheim tunneling into thermally grown SiO2," *J. Appl. Phys.*, vol. 40, no. 1, pp. 278–283, 1969.

[16] L. M. Grupp, J. D. Davis, and S. Swanson, "The bleak future of NAND flash memory," in *Proc. USENIX Conf. File Stor. Technol.*, San Jose, CA, USA, 2012, p. 2.

[17] A. A. Chien and V. Karamcheti, "Moore's law: The first ending and a new beginning," *IEEE Comput. Mag.*, vol. 46, no. 12, pp. 48–53, Dec. 2013.

[18] (2013). *ioDrive Octal*. [Online]. Available: http://www.fusionio.com/load/-media-/2grjf1/docsLibrary/FIODSOctal.pdf

[19] (2014). *Microsoft Production Server Traces*. [Online]. Available: http://iotta.snia.org/traces/158

[20] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Improving NAND endurance by dynamic program and erase scaling," in *Proc. USENIX Workshop Hot Topics Stor. File Syst.*, San Jose, CA, USA, 2013, p. 4.

[21] 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.

[22] K. Fukuda, Y. Shimizu, K. Amemiya, M. Kamoshida, and C. Hu, "Random telegraph noise in flash memories—Model and technology scaling," in *Proc. IEEE Int. Electron Devices Meeting*, Washington, DC, USA, 2007, pp. 169–172.

[23] Y. Pan, G. Dong, and T. Zhang, "Exploiting memory device wear-out dynamics to improve NAND flash memory system performance," in *Proc. USENIX Symp. Oper. Syst. Design Implement.*, San Jose, CA, USA, 2011, p. 18.

[24] S.-H. Shin *et al.*, "A new 3-bit programming algorithm using SLC-to-TLC migration for 8MB/s high performance TLC NAND flash memory," in *Proc. IEEE Symp. VLSI Circuits*, Honolulu, HI, USA, 2012, pp. 132–133.

[25] I. J. Chang and J.-S. Yang, "Bit-error rate improvement of TLC NAND flash using state re-ordering," *IEICE Electron. Exp.*, vol. 9, no. 23, pp. 1775–1779, 2012.

[26] M. Goldman, K. Pangal, G. Naso, and A. Goda, "25nm 64Gb 130mm^2 3bpc NAND flash memory," in *Proc. IEEE Int. Memory Workshop*, Monterey, CA, USA, 2013, pp. 1–4.

[27] D. W. Lee *et al.*, "The operation algorithm for improving the reliability of TLC (triple level cell) NAND flash characteristics," in *Proc. IEEE Int. Memory Workshop*, Monterey, CA, USA, 2011, pp. 1–2.

[28] C. Gniady, A. R. Butt, and Y. C. Hu, "Program-counter-based pattern classification in buffer caching," in *Proc. USENIX Symp. Oper. Syst. Design Implement.*, vol. 6. San Francisco, CA, USA, 2004, p. 27.

[29] (2007). *Websearch Trace From UMass Trace Repository*. [Online]. Available: http://traces.cs.umass.edu/index.php/Storage/Storage

[30] (2010). *Introducing tpce-Like Workload for MySQL*. [Online]. Available: http://www.percona.com/blog/2010/02/08/introducing-tpce-like-workload-for-mysql

[31] (2011). *Telecom Application Transaction Processing Benchmark*. [Online]. Available: http://tatpbenchmark.sourceforge.net

[32] (2012). *Getting the Hang of IOPS*. [Online]. Available: http://www.symantec.com/connect/articles/getting-hang-iops-v13

[33] N. Papandreou *et al.*, "Using adaptive read voltage thresholds to enhance the reliability of MLC NAND flash memory systems," in *Proc. Great Lakes Symp. VLSI*, Houston, TX, USA, 2014, pp. 151–156.

[34] Y. Cai *et al.*, "Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery," in *Proc. Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Rio de Janeiro, Brazil, 2015, pp. 438–449.

[35] K. Ha, J. Jeong, and J. Kim, "A read-disturb management technique for high-density NAND flash memory," in *Proc. ACM Asia-Pac. Workshop Syst.*, Singapore, 2013, Art. ID 13.

[36] C.-Y. Liu, Y.-M. Chang, and Y.-H. Chang, "Read leveling for flash storage systems," in *Proc. ACM Int. Syst. Stor. Conf.*, Haifa, Israel, 2015, Art. ID 5.

Keonsoo Ha received the B.E. degree in information and communication engineering from Sungkyunkwan University, Suwon, Korea, in 2005, and the Ph.D. degree in computer science and engineering from Seoul National University, Seoul, Korea, in 2015.

His current research interests include storage systems, computer architecture, and embedded systems.

Jaeyong Jeong received the B.S. and M.S. degrees in radio science and engineering from Korea University, Seoul, Korea, in 1996 and 1998, respectively. He is currently pursuing the Ph.D. degree in computer science and engineering with Seoul National University, Seoul.

From 1998 to 2012, he was a Flash Memory Design Engineer with the Memory Division, Samsung Electronics, Seoul. His current research interests include nonvolatile memory design, embedded software, storage systems, and operating systems.

Jihong Kim (M'00) received the B.S. degree in computer science and statistics from Seoul National University (SNU), Seoul, Korea, in 1986, and the M.S. and Ph.D. degrees in computer science and engineering from the University of Washington, Seattle, WA, USA, in 1988 and 1995, respectively.

Before joining SNU in 1997, he was a Technical Staff Member with the DSPS Research and Development Center, Texas Instruments, Dallas, TX, USA. He is currently a Professor with the School of Computer Science and Engineering, SNU. His current research interests include embedded software, low-power systems, computer architecture, and storage systems.