STRAW: A Stress-Aware WL-Based Read Reclaim Technique for High-Density NAND Flash-Based SSDs

Myoungjun Chun[1], Jaeyong Lee[1], Inhyuk Choi[1], Jisung Park[2], Myungsuk Kim[3], Jihong Kim[1]
[1]Seoul National University, [2]POSTECH, [3]Kyungpook National University

I. Introduction

In NAND flash memory, read disturbance is the phenomenon in which a read operation to the target wordline (WL) disturbs non-target WLs in the same block. Although read disturbance has emerged as a major reliability concern, managing read disturbance in modern NAND flash memory has not been thoroughly investigated yet. To our knowledge, all prior works on read disturbance in the literature are based on a simple SSD-management task, called *read reclaim (RR)* [1]. When a block's read count RC (i.e., the number of page reads to the block) exceeds a predefined threshold RC_{MAX}, the SSD controller triggers RR to eliminate read-disturbance-induced errors by rewriting (copying) all valid pages in the block to other free pages.

In this work, we show that a conventional RR approach causes prohibitive performance overhead to guarantee data reliability in recent high-density 3D flash memory. In high-density 3D flash memory, reading a page incurs significantly higher disturbance to the target WL's *exact neighbors* compared to the other WLs in the same block [2]. Such an asymmetry in read disturbance across WLs makes the existing *block-level* RR extremely inefficient. For example, when pages at the k-th WL WL_k are read repeatedly, pages at WL_{k-1} and WL_{k+1} may lose their data at a much lower RC value over when pages are randomly accessed over entire WL's. Although the worst-case access pattern (i.e., repeated reads for the same WL) may not be likely in practice, the existing RR approach should handle such a case safely, thus RC_{MAX} being set based on the worst-case pattern.

To mitigate RR overhead, we propose STRAW (STRess-Aware WL-based read reclaim technique for high-density NAND flash-based SSDs), a new *WL-level* RR technique for modern SSDs which effectively minimizes unnecessary RR at low cost. The key idea of STRAW is to keep track of the accumulated read-disturbance effect to each WL and reclaim only truly necessary (heavily-disturbed) WLs. To this end, we construct a read-disturbance model that can accurately estimate the impact of a page read on the reliability of each non-target WL in the same block. Our evaluation using the state-of-the-art SSD simulator [3] shows that STRAW reduces RR-induced writes by 83.6% compared to existing RR approaches with negligible space overhead.

II. Motivation

Unlike in planar (2D) flash memory, where a page read disturbs all non-target WLs in the block almost equally [1], the reliability impact of read disturbance significantly varies across WLs in high-density 3D flash memory. Fig. 1 illustrates two key factors contributing to the read-disturbance variations. First, when reading a page, a high-density 3D flash chip applies a higher V_{pass} (V_{passH}, approximately 0.4V higher than V_{passL} [2]) to the two adjacent WLs compared to the non-adjacent WLs (Fig. 1(a)). Based on the widely-known Fowler–Nordheim (FN) tunneling equation, the impact of read disturbance is exponentially proportional to V_{pass} [1]. Consequently, reading a page leads to a higher reliability impact on the data stored in adjacent WLs [2]. Second, read-disturbance tolerance varies significantly among WLs due to inherent process variations in high-density 3D flash memory [4]. Fig. 1(b) shows the maximum tolerable read counts for WLs within a block when the block's pages are accessed uniformly. As shown in Fig. 1(b), the worst WL in a block can tolerate only 403K reads before data corruption, while the best WL can reliably endure 559K additional reads.

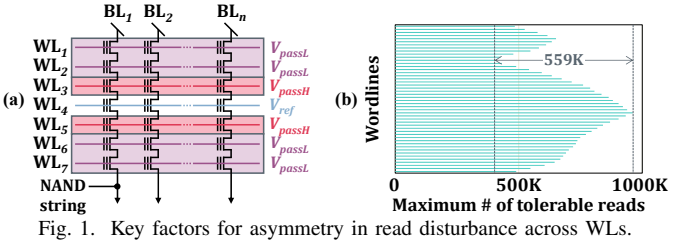

Fig. 1. Key factors for asymmetry in read disturbance across WLs.

Due to the heterogeneous reliability impact of read disturbance, data loss can occur at significantly lower RC values under worst-case access patterns. For example, in the worst-case access pattern, reading data stored in the most vulnerable WL can result in uncorrectable errors after only 54,560 block reads. In contrast, the same block can tolerate up to 518,420 reliable reads under a sequential access pattern. Consequently, block-level RR must conservatively set RC_{MAX} (54,560 in this example), leading to frequent and unnecessary RR operations.

III. STRAW: Stress-Aware WL-Based Read Reclaim

To overcome the limitations of existing solutions, we propose STRAW, which reclaims individual WLs only when necessary. To this end, we develop (*i*) a new read-disturbance model that quantifies the heterogeneous reliability impact of read disturbance (§III-A) and (*ii*) a STRAW-enabled flash translation layer (FTL) that efficiently estimates the actual read disturbance accumulated to each WL (§III-B) by leveraging an approximate counting algorithm [5].

A. New Read-Disturbance Model

We develop a new read-disturbance model through comprehensive characterization of 160 real 3D TLC flash chips from Samsung. Our proposed model quantifies two key factors that contribute to the heterogeneous disturbance impact on non-target WLs during a read operation: (*i*) the inherent process variations across WLs [4] and (*ii*) read-disturbance asymmetry between adjacent and non-adjacent WLs.

We derive a model with two key parameters for four groups, Best, Good, Bad, and Worst, based on their initial RBER values: (*i*) the effective maximum read count, ERC_{MAX}, which denotes the maximum number of reads the worst WL in a group can tolerate from non-adjacent WL reads, and (*ii*) the disturbance rate α, which quantifies the relative impact of adjacent WL reads compared to non-adjacent WL reads. The proposed model allows for determining whether a WL is heavily disturbed by using its current effective read count, derived from the read counts of its adjacent and non-adjacent WLs. Fig. 2 shows the parameters of the final model for the tested flash chips under different PEC. For example, at 2K PEC, $\text{WL}_w^{\text{Good}}$

PEC	ERC_{MAX}	α	ERC_{MAX}	α	ERC_{MAX}	α	ERC_{MAX}	α
0	1947K	7.4	1657K	7.6	1391K	7.8	1210K	8.0
0.5K	1806K	7.7	1521K	7.9	1270K	8.0	1094K	8.3
1K	1567K	7.9	1309K	8.2	1087K	8.4	922K	8.6
1.5K	1310K	8.3	1086K	8.6	899K	8.8	749K	9.0
2K	933K	8.7	767K	9.0	627K	9.2	518K	9.5
2.5K	539K	9.2	438K	9.5	354K	9.7	288K	10.1
3K	111K	9.7	89K	10.0	71K	10.3	58K	10.7
	Best		Good		Bad		Worst	

Fig. 2. Final Model of RC_{MAX} and α under different PEC.

can tolerate 767K non-adjacent reads, and the disturbance rate α is 9.0, respectively.

B. STRAW*FTL*

We implement an STRAW-enabled FTL, called STRAWFTL, by extending the conventional page-level FTL [3] with two key data structures: (*i*) Read-reclaim Parameter Table (RPT) and (*ii*) Resource-Efficient Counters (REC). The RPT is a table to store ERC_{MAX} and α for each PEC, which can be built through offline profiling of target chips (Fig. 2). The REC is a set of per-block counters that keep tracks the RC values of individual WLs within a block.

Fig. 3 illustrates how STRAWFTL estimates the accumulated disturbance impact on individual WLs. For WL_i, which is located in the i-th WL in the k-th block, it first looks up the RC values of WL_{i-1}, WL_i, WL_{i+1}, and BLK_k from the REC (❶). Based on the obtained RC values, STRAWFTL determines the number of reads to adjacent and non-adjacent WLs of WL_i (❷). Then it queries the RPT with the PEC of BLK_k and the WL group to which WL_i belongs (❸). STRAWFTL converts the number of reads to adjacent and non-adjacent WLs of WL_i into the ERC, using the disturbance rate α from the query result (❹). The remaining process is straightforward. If the ERC of WL_i (including the possible additional reads by the next interval) exceeds ERC_{MAX} from the RPT, STRAWFTL identifies WL_i as a heavily-disturbed WL (❺).

Whenever a page is read, STRAWFTL updates the REC for the target block and WL. Every predefined interval (e.g., every 1K reads to the block), STRAWFTL checks all valid WLs in the block to determine whether the accumulated disturbance impact on any valid WL exceeds the threshold or if there is a possibility it will exceed the threshold by the next interval. For such WLs, STRAWFTL copies the valid pages to free pages before the next interval, thereby preventing read-disturbance-induced data corruption. If the block contains no valid pages after the checking procedure, STRAWFTL erases the block and resets all associated counters.

Overhead Optimization. To minimize the storage overhead of per-WL counters, the REC incorporates the Space-Saving (SS) algorithm [5], which efficiently estimates the frequency of elements in a data stream using a limited number of counters. Due to the limited number of counters, the estimated read counts by the REC may introduce some error, but SS ensures that the estimated count value for any element is never underestimated [5]. This guarantees that errors in estimation do not result in read-disturbance-induced data corruption, although they may cause premature RR invocations.

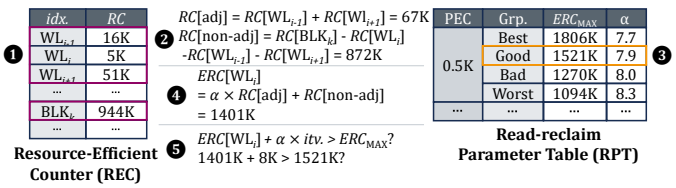

Fig. 3. A procedure for identifying heavily-disturbed WLs in STRAW.

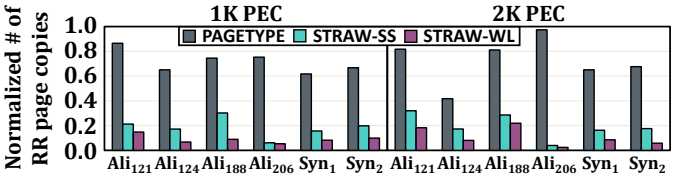

Fig. 4. Comparison of RR-induced page copies under two PEC.

IV. EVALUATION

We evaluate the effectiveness of our proposal using MQSim-E [3], a state-of-the-art SSD simulator. We extend MQSim-E to trigger RR operations according to the read-disturbance characteristics observed in our 19,200 tested blocks. We evaluate two synthetic workloads with different I/O patterns, as well as four real-world workloads obtained from Alicloud traces.

We compare four SSD configurations with different RR techniques, BLOCK, PAGETYPE, STRAW-SS, and STRAW-WL. BLOCK is our baseline SSD that employs block-level RR. PAGETYPE is an SSD that adopts a state-of-the-art read-disturbance management technique [6]. Unlike block-level RR, PAGETYPE classifies pages within a block according to their page types (e.g., MSB, CSB, and LSB pages in TLC flash memory) and migrates them based on their vulnerability to read disturbance. Both STRAW-SS and STRAW-WL are SSDs that implement STRAWFTL; however, STRAW-SS utilizes the SS algorithms [5] for WL-level counters (32 counter entries for a block), while STRAW-WL employs naive WL-level counters.

Fig. 4 compares the number of RR-induced page copies in four SSD configurations, normalized to BLOCK, under two different PECs. We make two observations. First, both STRAW-SS and STRAW-WL significantly reduce the number of RR-induced page copies compared to BLOCK, by preventing premature RR invocations. For example, STRAW-SS (STRAW-WL) reduces the number of RR-induced page copies over BLOCK by 83.8% (91.5%) on average at 2K PEC. Second, with significantly less space overhead, STRAW-SS achieves efficiency comparable to STRAW-WL under random-read patterns and remains competitive under sequential-read patterns.

In conclusion, we have proposed a new WL-level read reclaim technique, STRAW, which significantly improves SSD lifetime and performance by reducing the frequency of RR invocation. Unlike block-level RR that performs RR at block granularity, STRAW identifies heavily-disturbed WLs within blocks and reclaims them in a timely manner. Our evaluation results showed that STRAW effectively enhances SSD lifetime and performance.

V. ORIGINAL PUBLICATION

M. Chun *et al.* 2024. STRAW: A Stress-Aware WL-Based Read Reclaim Technique for High-Density NAND Flash-Based SSDs. IEEE Computer Architecture Letters (IEEE CAL). https://arxiv.org/abs/2501.02517

REFERENCES

[1] K. Ha *et al.*, "An Integrated Approach for Managing Read Disturbs in High-Density NAND Flash Memory," *IEEE TCAD*, 2015.
[2] Q. Xiong *et al.*, "Characterizing 3D Floating Gate NAND Flash: Observations, Analyses, and Implications," *ACM TOS*, 2018.
[3] D. Lee *et al.*, "MQSim-E: An Enterprise SSD Simulator," *IEEE CAL*, 2022.
[4] Y. Shim *et al.*, "Exploiting Process Similarity of 3D Flash Memory for High Performance SSDs," in *MICRO*, 2019.
[5] A. Metwally *et al.*, "Efficient Computation of Frequent and Top-k Elements in Data Streams," in *ICDT*, 2005.
[6] S. Han *et al.*, "Page Type-Aware Data Migration Technique for Read Disturb Management of NAND Flash Memory," *IEEE TVLSI*, 2023.