Instant Data Sanitization on Multi-Level-Cell NAND Flash Memory

Md Raquibuzzaman
The University of Alabama in Huntsville
mr0068@uah.edu

Matchima Buddhanoy
The University of Alabama in Huntsville
mb0194@uah.edu

Aleksandar Milenkovic
The University of Alabama in Huntsville
milenka@uah.edu

Biswajit Ray
The University of Alabama in Huntsville
biswajit.ray@uah.edu

ABSTRACT

Deleting data instantly from NAND flash memories incurs hefty overheads, and increases wear level. Existing solutions involve unlinking the physical page addresses making data inaccessible through standard interfaces, but they carry the risk of data leakage. An all-zero-in-place data overwrite has been proposed as a countermeasure, but it applies only to SLC flash memories. This paper introduces an *instant page data sanitization method for MLC flash memories that prevents leakage of deleted information* without any negative effects on valid data in shared pages. We implement and evaluate the proposed method on commercial 2D and 3D NAND flash memory chips.

CCS CONCEPTS

• **Hardware → Non-volatile memory**; • **Security and privacy → Hardware security implementation**.

KEYWORDS

Multi level cell (MLC), Data Sanitization, 3D NAND

ACM Reference Format:
Md Raquibuzzaman, Matchima Buddhanoy, Aleksandar Milenkovic, and Biswajit Ray. 2022. Instant Data Sanitization on Multi-Level-Cell NAND Flash Memory. In *The 15th ACM International Systems and Storage Conference (SYSTOR '22), June 13–15, 2022, Haifa, Israel.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534056.3534941

SYSTOR '22, June 13–15, 2022, Haifa, Israel

ACM ISBN 978-1-4503-9380-5/22/06…$15.00
https://doi.org/10.1145/3534056.3534941

1 INTRODUCTION

Due to the increasing popularity of flash memories for storing private data in smartphones, SD cards, USB flash drives, and solid-state drives (SSDs), instant sanitization of user data from the flash media has become extremely important to preserving user privacy. According to the Data Protection Act (DPA) 2018 [15], the deletion of information must be real, i.e., the content should not be recoverable in any way. Unfortunately, the standard data deletion methods of today's solid-state storage devices do not offer any instant data sanitization capabilities to the end-user [9, 10, 16, 17, 22]. Although the state-of-the-art data deletion methods make the data inaccessible through standard memory interfaces, recent research efforts demonstrate that data is partially or fully recoverable by employing advanced memory characterization techniques. According to a recent report from Blancco Technology Group, 42 of used SSDs sold on eBay hold sensitive data [6, 20]. Furthermore, the report states that the data is recoverable even from SSDs that were subjected to standard data sanitization methods. This highlights a major concern that while sellers clearly recognize the importance of data deletion, they are, in fact, using methods that are inadequate and do not ensure true sanitization of data.

NAND flash memories present several challenges to instant data deletion due to their organization and operation. First, flash memory chips are organized in flash blocks, where each block consists of multiple pages. Basic flash operations are page write (program), page read, and block erase. Thus, flash-write operations that store data into the flash memory take place at a page-level granularity, whereas erase operations take place at a block-level granularity. Second, NAND flash memories employ an erase-before-write paradigm that requires that blocks be erased before pages within a block are written into. This requirement makes in-place page update operations impractical because all valid pages in a block would need to be copied into another block before the block is erased and the page is updated. Instead, the updated data is

written into a different page, whereas the existing page is simply "unlinked" or marked as invalid. However, the original content remains in the "unlinked" flash page. Third, NAND flash memories have finite endurance, meaning that only a fixed number of program and erase operations is allowed on a NAND block during its lifetime; once this number is exceeded, the block becomes unreliable for storing data. To manage these basic flash operations (erase block, program page, read page) and map high-level storage abstractions into physical pages and blocks, an intermediate firmware layer called Flash Translation Layer (FTL) is employed [2]. This firmware typically runs on a dedicated flash memory controller that employs sophisticated algorithms for wear leveling, garbage collection, error correction, and others. These algorithms try to minimize read latency, ensure data integrity, and maximize the lifetime of physical media.

In order to alleviate significant overhead of erase-based techniques and achieve page-level sanitization of deleted data in flash-based storage, a page-overwrite with all-zeros technique was introduced by Wei et al. [23] and later improved by others [1, 5, 7]. The method creates an all-zero page by an over-write operation, thus removing the information from the page. Even though all-zero sanitization works for single-level cell (SLC) memories, it cannot be directly applied to MLC memories where two logical pages share the same set of memory cells. An all-zero overwrite on one logical page may corrupt data on the other shared page. Thus, instant data sanitization remains an open problem for multi-level cell (MLC), triple-level cell (TLC), or quad-level cell (QLC) flash memories.

This paper introduces an instant page sanitization method for MLC flash memories that prevents leakage of deleted information without any negative effects on valid data in shared pages. Our experimental evaluation on commercial NAND flash memory chips shows that the proposed method ensures true data sanitization that leaves no traces of the original data even in the analog threshold voltage distribution of the sanitized pages. The proposed method incurs minimal disturbance on valid data residing in neighboring pages, it does not require any hardware modification, and it can be implemented using standard user-mode flash commands in the FTL firmware.

The rest of the paper is organized as follows. Section 2 describes the background through a brief NAND flash preliminary and a summary of existing techniques and academic research in the area of NAND flash data sanitization. Section 3 describes our proposed sanitization method. Section 4 describes the results of the experimental evaluation, and Section 5 concludes the paper.

2 BACKGROUND

2.1 Nand Flash Memory Preliminaries

A basic building block of a NAND flash memory is a memory cell, a MOSFET transistor with a floating gate (FG) that can trap negative charge. The presence of negative charges on the FG effectively increases the transistor's threshold voltage (V_t) relative to the case when there is no charge on the FG. Thus, a flash memory cell is a charge-based analog memory that stores information in the form of threshold voltages – cells with no charge are erased holding a logic '1' and cells with trapped charge are programmed holding a logic '0.' This type of memory cell holds one bit of information – a.k.a. SLC. A NAND flash memory block is organized as a two-dimensional array of memory cells, as shown in Fig. 1(a). Cells in a row constitute a page, and their control gates are connected to a shared word line (WL). The page size varies from 2-16 kilobytes, depending on the manufacturer. A collection of pages (e.g., 2048) forms a flash memory block. The cells in a vertical column of a memory block are connected via select transistors to a metal bit line (BL) at one end and to the ground at the other end. In NAND flash memories, data are read or programmed at the page level, whereas erase operations are performed at the block level. Any flash cell that is set to a logic '0' by a program operation can only be reset to a logic '1' by erasing the entire block. Thus, flash memories do not support in-place data update to change the content of a logical page, the new content is programmed into a fresh flash page.

Cell V_t distribution in MLC. MLC flash memory cells store 2 bits of data, and they are programmed to have their V_t in one of four different states, L_0, L_1, L_2 and L_3 as shown in Fig. 1(b). The data bits corresponding to each V_t state are shown in two colors to indicate the most significant bit (MSB) in red and the least significant bit (LSB) in blue. The Gray's code is commonly used to encode the states ($L_0 = 11, L_1 = 01, L_2 = 00$ and $L_3 = 10$). The MSB bits of all memory cells' states in a row form the MSB page. Similarly, the LSB bits of all the memory cells' states form the LSB page. Thus, every LSB page has a corresponding MSB page, and they are called shared pages because they share the same set of physical memory cells. When a block is erased, all charges are removed from the cells' floating gate (FG), placing the cells into the erased state (L_0). Programming an LSB page involves injecting charges on selected cells' FG to move their state from L_0 to L_2 as illustrated in Fig. 1(b). The corresponding MSB page is programmed after the LSB page programming is finished. During MSB page programming, certain cells transition from L_0 to L_1 and certain cells transition from L_2 to L_3. Thus, after programming both pages, four V_t states are created in the memory array. Two read reference voltages

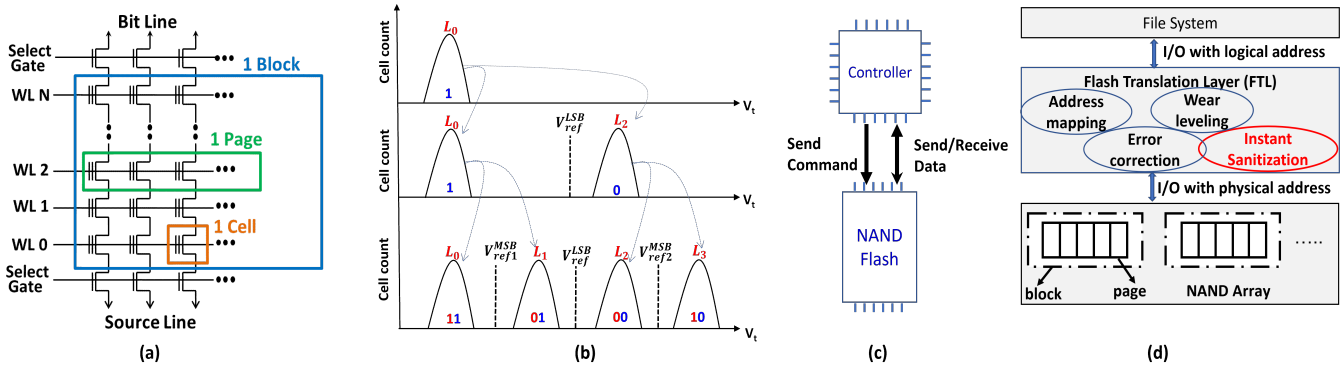

Figure 1: (a) NAND flash memory block organization. (b) Threshold voltage distribution plot for MLC flash memories. (c) Simplified view of a storage system containing a memory chip and controller. (d) Main functions of FTL.

are used to read the MSB page data, whereas only one read reference voltage is needed to read the LSB page data, as shown in Fig. 1(b).

Flash-based storage system. Fig. 1(c) shows a simplified view of a flash memory-based storage system consisting of a flash memory controller and one or more NAND flash memory chips. The flash controller manages the flash media by executing several tasks defined in the FTL. FTL provides an interface between the host file system on one side and the physical NAND flash memory chips on the other side by mapping the logical addresses to physical addresses in NAND flash, as shown in Fig. 1(d). In addition, FTL contains firmware modules that perform error correction, garbage collection, and wear-leveling [4]. The garbage collection module periodically reclaims all the invalid pages in the media to perform a block erase operation, which will free up memory space for new data. The wear-leveling module manages the limited endurance of the flash media by ensuring uniform program-erase operations across different NAND flash memory chips and blocks within a chip.

2.2 Data Sanitization in NAND Flash

Standard overwrite-based erasure techniques used in hard drives do not work in NAND flash-based storage systems since in-place updates are not possible in NAND flash memories. Instead, the following methods are typically employed by the flash controller for data sanitization.

Block erase. The block erasure is a basic NAND command to remove data from all pages in a flash block. This method essentially removes charges from all flash cells in the block and hence physically erases the data from the media. Typically, the garbage collection function of the FTL uses this method to remove old invalid data once the drive is almost full. A major drawback of using the block erasure for instant sanitization is its poor performance caused by significant overhead due to valid data migration [3, 19, 21] and a block erase operation that takes more time than a page program operation. In addition, this technique increases wear level and thus limits the effective capacity/operating time of storage systems. Thus, this command is sparingly used by the flash controller.

Logical data deletion. Since the block erasure suffers from poor performance, solid-state storage devices usually perform logical deletion of data by invalidating the page addresses of obsolete data [13, 14, 24]. The page address mapping is handled by the FTL, which performs one-to-one mapping between a logical page address and a physical page address on the flash media. Thus, for any page update operation, the FTL will write the new contents to another fresh page (or block) and update the address map table to point to the new page. As a result, the old version of the data remains in the physical storage medium, from where it can be retrieved by an adversary with advanced memory interfaces. Thus, this method does not ensure true sanitization of data.

Encryption-based data deletion. Several authors have recently proposed data deletion methods based on encryption [10, 16–18]. The basic idea in this method is to encrypt the user file with an encryption key and store the encrypted data and the key in two separate NAND blocks. A secure data deletion is achieved by removing the keys, which can be done efficiently as keys require less space in memory. Even though encryption-based techniques are promising, they suffer from the following drawbacks. First, the encryption-based deletion method carries the risk of data recovery as its implementation may have certain issues, e.g., random

number generation (for encryption key) that can be compromised by a motivated adversary. Second, encryption-based deletion requires proper removal of encryption keys and any other derived values that might be useful in cryptanalysis. Third, many existing storage systems and embedded platforms do not include hardware modules for accelerating encryption/decryption tasks and rely on software solutions that can severely limit system performance.

All zero-overwrite-based page sanitization. In order to achieve the page-level deletion in flash medium, the idea of "data scrubbing" was proposed by Wei et al. [23] and later improved by others [1, 5, 7]. The "scrubbing" based sanitization relies on programming an all-zero data, which is equivalent to the deletion of data from that page. Thus, "scrubbing" provides an alternative route to *digital sanitization* by reprogramming all the cells in the page. Unfortunately, the "scrubbing" method only applies to SLC flash memories, leaving the problem of instant data sanitization unsolved for MLC flash memories.

Sanitization using one-shot programming. Lin et al. [11] propose a technique for partial sanitization for MLC flash memories. Specifically, they utilize one-shot programming to sanitize data in MSB and LSB pages. Whereas this technique can achieve fast sanitization, it relies on proprietary flash commands for one-shot programming that are not typically exposed through standard NAND flash interfaces. In addition, as it does not involve any verification steps, it requires prior characterization to determine one-shot programming voltage for individual cell states, limiting its practicality. Next, one-shot programming does not completely prevent data leakage as it eliminates only one of the possible four cell states. Note that to prevent any data leakage from a sanitized shared page, the flash cells in the shared page should be in at most two cell states.

Evanesco. Kim et al. [8] introduce Evanesco – a hardware-supported technique for efficient data sanitization in modern flash-based storage systems. It provides data sanitization by blocking access to invalidated data either on page or block level. Whereas this approach is very effective, it requires additional hardware and is not applicable to the existing flash memory chips. In addition, the data stored in the flash medium is not physically removed, making it susceptible to future direct or indirect data retrieval attacks.

3 PROPOSED SANITIZAITON METHOD

To alleviate problems of the existing approaches, we introduce a method for instant sanitization that does not require any special hardware support and can be fully implemented in the FTL using non-privileged commands on commercial chips. The proposed method allows for the sanitization of (a) an LSB page while preserving the data in the corresponding MSB page, (b) an MSB page while preserving the data in the corresponding LSB page, and (c) both LSB and MSB shared pages. Even though shared LSB and MSB pages are logically independent of each other, physically, they are interdependent as they share the same set of memory cells. Thus, overwriting an LSB or MSB page may corrupt the data stored in the corresponding shared page. Hence, data sanitization requires careful implementation in order to ensure data integrity of the corresponding shared page.

3.1 Sanitization of LSB page only

We start off with valid data in both LSB and MSB shared pages. Fig. 2(a) illustrates V_t distribution among all four states. If random data are written in both pages, memory cells sharing a single word line will be evenly split among four states (25% in each). Fig. 2(b) shows state transitions of cells that are required to sanitize an LSB page while preserving data in the corresponding MSB page. Specifically, all cells in the L_1 state need to transition into the L_2 state, thus converting the LSB bits from 1 to 0 while retaining values of the corresponding MSB bits. Similarly, all cells from the L_0 state need to transition into the L_3 state. After these transitions are carried out, all cells will be in either L_2 or L_3 state. An LSB read will return all 0s because all the cells will have $V_t > V_{ref}^{LSB}$. An MSB read will return the original content from the MSB page.

To carry out the state transitions described above using common flash operations, the flash controller needs to carry out a sequence of steps as illustrated in Fig. 2(e). The first step is to read the MSB page and store it in a temporary buffer. Next, the controller sends all-zero data for the LSB page and the buffer data for the MSB page and initiates a page program operation.

The overhead of the proposed sanitization method includes the time needed for (a) one MSB page read, (b) sending data for both LSB and MSB pages, and (c) programming data into the selected memory cells. However, this overhead is orders of magnitude smaller than the overhead required to copy all valid pages from the target flash block to another fresh block and then perform a block erase operation. It should be noted that MLC flash memory chips use two different approaches when programming MLC pages. The chip used in this research requires both pages to be loaded into the internal buffers of the flash memory chip before a program command triggers program operation on both pages. Other flash memory chips allow LSB and MSB pages to be independently programmed. In that case, the overhead of LSB sanitization includes the time needed for one page read, sending data for both pages, programming an LSB, and programming an MSB page.

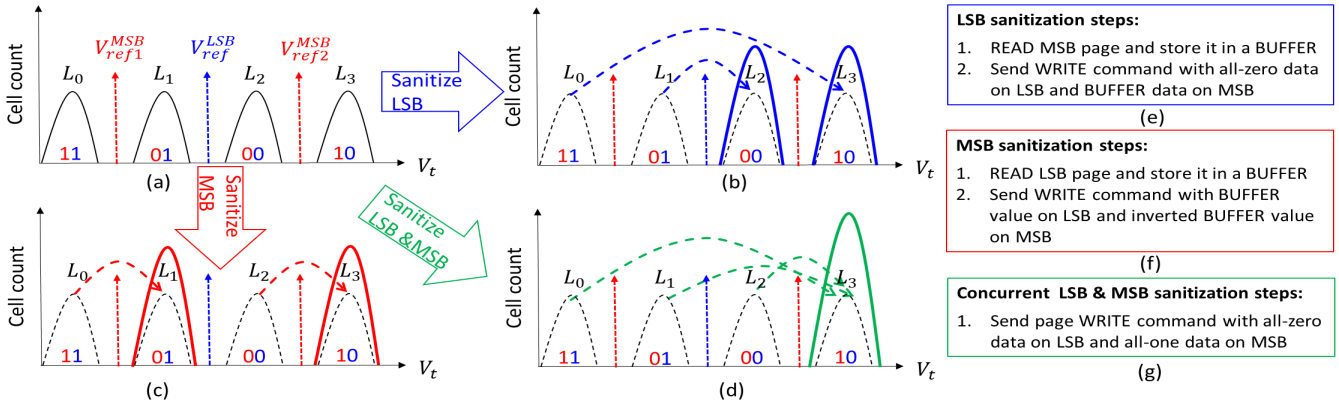

Figure 2: (a) Four V_t states of MLC memory cells with read reference voltages. State transitions for the proposed (b) LSB only sanitization, (c) MSB only sanitization, and (d) concurrent LSB and MSB sanitization. Sequence of steps to carry out (e) LSB sanitization, (f) MSB sanitization, and (g) concurrent LSB and MSB sanitization.

3.2 Sanitization of MSB page only

Fig. 2(c) shows the required state transitions of memory cells to sanitize an MSB page while preserving data in the corresponding LSB page. Memory cells in the L_0 state need to transition into L_1 state, converting the MSB bits from 1 to 0. To create an all-zero MSB page, we would need to transition cells from the L_3 into L_2 state. However, cell V_t cannot be decreased with a page program operation. Thus, we cannot create an all-zero MSB page without affecting LSB data. To circumvent this limitation, we propose to move cells from the L_2 into L_3 state. This way, the new content of the MSB page is a mirrored image of the data from the LSB page. Still, the sanitization goal is achieved as there are no traces of the original data from the MSB page, while the LSB page data is fully preserved. We can expect that the reliability of the LSB data will even improve after the sanitization of the MSB page. The voltage margin between L_1 and L_3 states is significantly higher than before the sanitization, so $0 \rightarrow 1$ bit flips in the LSB page are less likely.

Fig. 2(f) illustrates the sequence of steps needed to sanitize only an MSB page. First, the flash controller reads data from the corresponding LSB page and stores it into a buffer. Then, the inverted buffer content is transferred to the chip for the MSB page programming. Finally, the program command is initiated that programs the selected physical page. Please note that we typically do not have to resend the content for the LSB page as it can be copied internally in the flash memory chip. The overhead of the proposed MSB page only sanitization includes the time needed for (a) one LSB page read, (b) sending data for the MSB page, and (c) programming data into the selected memory cells. Please note that we do not count the controller's processing time as inverting the content of the buffer is a very simple operation that is not going to place any computational burden to the flash controller.

3.3 Concurrent LSB and MSB page sanitization

In order to sanitize concurrently both the LSB and MSB shared pages, all the cells of the memory layer need to be moved into the L_3 state, as illustrated in Fig. 2(d). In that case, the sanitized LSB page will contain all 0s, and the sanitized MSB page will hold all 1s. Fig. 2(g) shows the sequence of steps needed to carry out the concurrent sanitization of shared pages. The controller sends all 0s for the LSB page and all 1s for the MSB page into the respective flash memory buffers and initiates a shared page program operation. This method incurs even smaller overhead than the LSB only or MSB only sanitization, as it is agnostic of the original data stored in the shared page and does not require any page read operation. Please note that the proposed method allows for serialization of individual page sanitizations in the case of NAND flash chips with independent programming of LSB and MSB pages. For example, to sanitize the corresponding MSB page when its LSB pair is already sanitized, the flash controller will send all 1s data pattern and issue an MSB page program operation. To sanitize the corresponding LSB page when its MSB pair is already sanitized, the flash controller will send all 0s data pattern and issue an LSB page program operation.

4 EXPERIMENTAL EVALUATION

The experimental evaluation is performed on multiple 2D and 3D NAND memory chips from two different major NAND manufacturers. The chips support so-called read offset operations that allow the flash controller to adjust

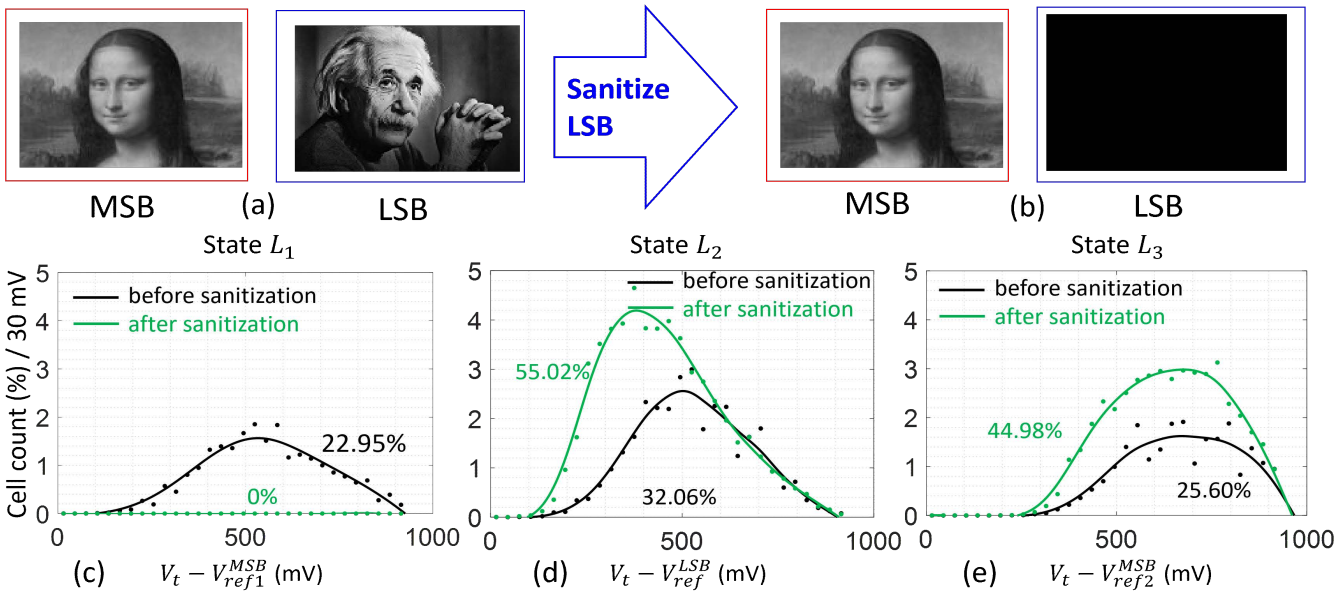

Figure 3: (a) MSB and LSB page before sanitization. (b) MSB and LSB page after LSB sanitization. (c)-(e) Measured V_t distribution of cells before and after sanitization for the L_1, L_2, and L_3 states, respectively. Cell V_t is measured with respect to the default read reference voltage of the page shown on x-axis.

read reference voltages V_{ref}^{LSB}, V_{ref1}^{MSB}, V_{ref2}^{MSB} (Fig. 2(a)). Nominally, this feature is used to help in recovering data when standard ECC correction fails. In this research, we progressively increase an offset and thus shift the reference voltages to the right from the default level to V_{ref} + ΔV × i, $i = 0, \ldots, 127, \Delta V = 7.5$ mV and then read data from the LSB and MSB pages. This way, we can extract the V_t distribution of memory cell states. For example, by shifting V_{ref1}^{MSB} from its default value to the right by ~ 952 mV in steps of 7.5 mV, we can extract the distribution of the L_1 state.

4.1 Evaluation of LSB page only sanitization

To evaluate the effectiveness of the proposed LSB only sanitization method, we conduct the following experiment. First, we write two different images of 16 KiB into shared pages; an Albert Einstein image is written to an LSB, and a Mona Lisa image is written to the corresponding MSB page. We read the written images and plot them in Fig. 3(a).The next step involves the characterization of memory cells' states before sanitization is performed. We utilize read offset operations on the read reference voltages V_{ref1}^{MSB}, V_{ref}^{LSB}, V_{ref2}^{MSB} to extract V_t distributions for the L_1, L_2, and L_3 states, respectively. Black dots in Fig. 3(c)-(e) show the percentage of cells from the shared page with V_t in each 30 mV range ($4 \times 7.5 = 30$ mV), whereas the black curves approximate the extracted V_t distribution for the L_1, L_2 and L_3 states, respectively. Please note that V_t of cells in the L_0 state is not accessible with the given range of V_{ref} sweep.

The next step in the evaluation is to carry out the LSB page only sanitization as described in Fig. 2(e). The pages are then read and plotted in Fig. 3(b). The LSB page with the blue frame is fully sanitized (contains all 0s), whereas the MSB page is intact. Thus, LSB page is fully sanitized with no information leakage. We calculate the raw bit error rate (BER) for the Mona Lisa image after sanitization. We find that it is the same or even lower than before sanitization. In addition, we extract the V_t distribution of cells' states after sanitization. Green dots in Fig. 3(c)- (e) show the percentage of cells with V_t in 30 mV range after LSB sanitization, whereas the green curves approximate the extracted V_t distribution for the L_1, L_2 and L_3 states, respectively. There are several takeaways from this evaluation. First, V_t distribution after sanitization does not distinguish the newly transferred cells from the original ones indicating true sanitization of the LSB page. Even if an adversary performs cell V_t analysis on a sanitized page, he/she will not be able to recover the original data. Second, the V_t distribution forms slightly longer tails after sanitization operation. Since there is a sufficient voltage margin between the reference voltage and cell V_t, BER of the valid data in the MSB page is not affected by these long tails.

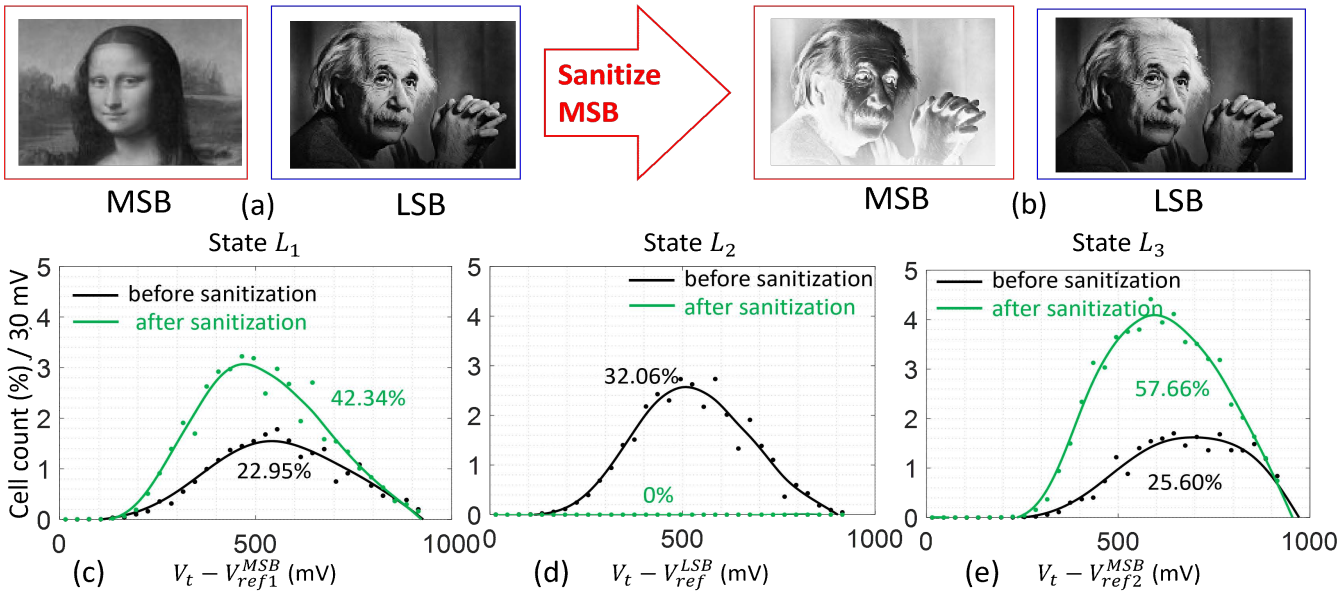

Figure 4: (a) MSB and LSB page before sanitization. (b) MSB page and LSB page after MSB sanitization. (c)-(e) V_t distribution of cells before and after sanitization for the L_1, L_2 and L_3 states, respectively. Cell V_t is measured with respect to the default read reference voltage of the page as shown on the x-axis.

4.2 Evaluation of MSB page only sanitization

Fig. 4 summarizes our evaluation results for the MSB page sanitization. The selected MLC layer is programmed to contain images of Einstein and Mona Lisa images on shared LSB and MSB pages, respectively, as shown in Fig. 4(a). The cell states are characterized like in the previous experiment, and then the steps for the MSB page sanitization are performed as described in Fig. 2(f). The MSB and LSB pages are read and plotted in Fig. 4(b). We find that the LSB data remains intact, and an inverted Einstein appears instead of the original Mona Lisa image in the MSB page. The effectiveness of the proposed method is further analyzed through V_t distribution measurements on the shared pages. Fig. 4(c)-(e) show the V_t distribution of flash cell states before sanitization in black and after sanitization in green. As expected, we find that after MSB page sanitization, there are only two V_t states, L_1 and L_3. We also verify that cells from the L_0 state have moved to L_1 and cells from the L_2 state have moved to L_3. We would like to emphasize several important points that we can infer from the V_t distributions. First, the evaluation confirms that the MSB page is truly sanitized as the original Mona Lisa image is replaced by an inverted Einstein image. V_t distribution of transferred cells and the original cells in a given state remains indistinguishable. Second, the BER of the LSB data improves after sanitization. Indeed, the separation between L_1 and L_3 is quite high to cause any errors due to noise or retention loss. Thus, we expect the reliability of the LSB data to increase after the sanitization of the corresponding MSB page. Third, an inverted LSB page data is copied on the MSB page. This can be utilized as a redundant copy of the LSB data to correct errors in the original LSB page if required.

It should be noted that having inverted copy of data from the LSB page in the sanitized MSB page does not imply any information leakage. The sanitized MSB page does not contain any trace of the original data, and a copy of the data that is already in the flash memory cannot be considered as information leakage. However, the FTL will have to keep track of all sanitized pages. Thus, if the LSB page (Einstein image in our example) is deleted after the corresponding MSB page is deleted (which now holds inverted Einstein image), we need to sanitize both pages as described in Fig. 2(d). All the cells of the shared pages are moved to L_3 state after both LSB and MSB pages are sanitized, creating an all zero LSB page and all one MSB page.

4.3 Evaluation of concurrent LSB and MSB page sanitization

Finally, we evaluate the effectiveness of the concurrent LSB and MSB page sanitization. We start by writing Einstein and Mona Lisa images on shared LSB and MSB pages, respectively (Fig. 5(a)). Then, we sanitize both pages following the steps described in Fig. 2(g). After both the pages are sanitized,

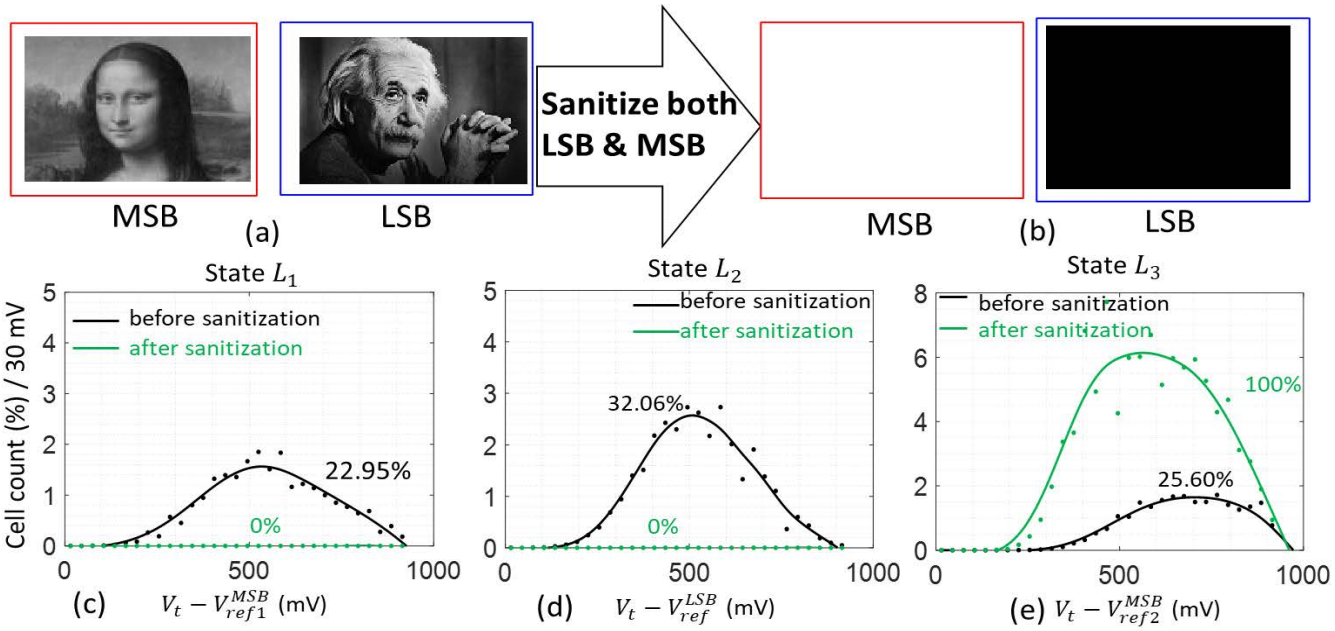

Figure 5: (a) MSB and LSB page before sanitization. (b) MSB and LSB page after concurrent MSB and LSB sanitization. (c)-(e) V_t distribution of cells before and after sanitization for the L_1, L_2 and L_3 states, respectively. Cell V_t is measured with respect to the default read reference voltage of the page as shown on the x-axis.

we find all-zero data on the LSB page and all-one data on the MSB page (Fig. 5(b)). The corresponding V_t distributions before and after sanitization are shown in Fig. 5(c)-(e). We find that all cells are transferred to the L_3 state after concurrent sanitization of LSB and MSB pages.

4.4 Effects of page sanitization on retention of valid data

We evaluate the page sanitization effects on the retention errors of the valid data in the shared pages by baking the memory chip at a higher temperature (120°C) for 1, 2, or 3 hours. Using the acceleration factor based calculation, we find that the 3 hours of baking time corresponds to 5 years at room temperature assuming activation energy for charge loss in 3D NAND as E_A= 1 eV [12]. Fig. 6 summarizes our evaluation results. We compare retention errors of unsanitized standard LSB/MSB pages with the retention errors on the valid data of the LSB/MSB pages after the corresponding shared pages are sanitized. We find that the retention reliability of the LSB page improves after the corresponding MSB sanitization. However, the retention reliability of the MSB page after LSB sanitization degrades compared to the unsanitized MSB pages. Since V_t margin between program states increases after MSB sanitization as shown in Fig. 2(c), retention reliability improves for the LSB page. However, V_t margin between L_2 and L_3 states reduces after LSB sanitization as shown in Fig. 2(b), and hence MSB page shows more retention errors after LSB sanitization. Nevertheless, the BER after 3 hours of baking remains significantly lower than what can be corrected by the standard error correction code (ECC). Thus, we find that the proposed mechanism does not severely impact the retention of shared valid pages.

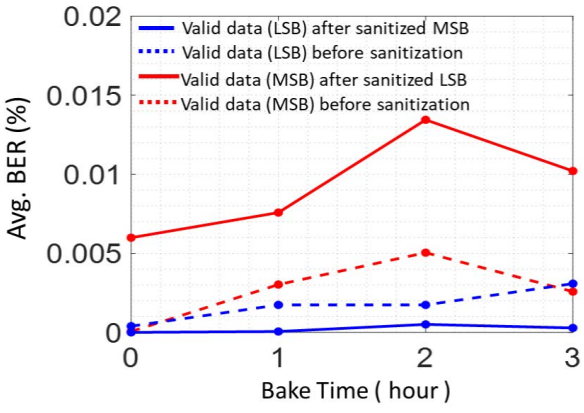

Figure 6: Effects of page sanitization on the retention errors of valid data of the shared page.

Table 1: Multiple chip sanitization results

Chip Type	LSB Page BER		MSB Page BER	
	Before MSB Sanitization	After MSB Sanitization	Before LSB Sanitization	After LSB Sanitization
64-Layer 3D NAND	$3.9x10^{-4}\%$	0%	$5.6x10^{-5}\%$	0.006%
19-nm 2D NAND	0.007%	0.001%	0.09%	0.475%

4.5 Effects of page sanitization on valid data in neighbor layers

Since sanitization of a page involves an extra write operation on a memory page, it may disturb the valid data on neighboring memory layers. We evaluate the effects of MLC page sanitization on the BER of the neighboring memory layers. We first write random data in the entire memory block. Then, we sanitize shared memory pages in the MLC layer one by one. There are 12 shared pages (12-LSB and 12-MSB pages) in an MLC layer of the 3D NAND memory block. We perform concurrent LSB and MSB sanitization as it causes the worst-case disturbance on the neighboring memory layers.

Fig. 7 summarizes the evaluation results where we plot the BER of the valid pages in the neighboring memory layers as a function of the number of sanitized memory pages in a given memory layer. We observe an increase in BER for the data stored in the nearest neighboring layer (Layer $n-1$) as we increase the number of sanitized pages in a target layer (Layer n). The maximum BER increase is ~ 5 times after all the pages of a given layer are sanitized. Though this is seemingly a significant increase in BER, the actual magnitude is still significantly below what can be corrected using

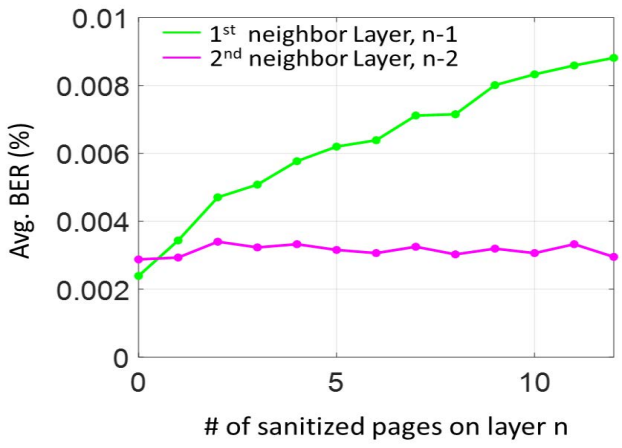

Figure 7: Effects of proposed page sanitization on the valid data on neighboring memory layers.

standard ECC. Interestingly, other memory layers ($n-2$ and beyond) remain unaffected by the sanitization process.

4.6 Evaluation on multiple chips

Table-1 summarizes our evaluation results that demonstrate that the proposed technique is applicable to 2D and 3D MLC chips manufactured by two different vendors. Table-1 quantifies the impact that the proposed sanitization has on the valid data in the corresponding shared page by comparing its average BER before and after sanitization is performed. The first row in the table corresponds to a 64-layer 3D NAND chip that has been used in the evaluation thus far. The second row in the table shows the results from a 19 nm 2D NAND chip from a different vendor. We find that sanitization of an MSB page improves the BER of valid data on the corresponding LSB page, whereas sanitization of an LSB page increases the BER of valid data on the corresponding MSB page. These findings are in line with those described in Section 4.1 and Section 4.2.

4.7 Limitations

It should be noted that some flash memory chips may not support overwriting of memory pages once a block is fully programmed, thus preventing any overwrite-based data sanitization. Next, internal data randomization of NAND memory chips can also pose challenges to overwrite-based sanitization. However, we believe these are not fundamental limitations and can be addressed by chip vendors.

4.8 Future work

The proposed mechanism has been tested for 2D and 3D MLC flash memory chips. This work can be extended in several directions as follows. The first is to evaluate the proposed mechanism in TLC/QLC flash memories. Although the proposed mechanism should work for TLC/QLC chips with some minor modifications, a detailed evaluation is needed to determine its effectiveness and robustness. The higher number of bits per cell in TLC/QLC chips results in lower

threshold voltage margins between the cell states. Thus, sanitization steps may result in increased BERs in shared and neighboring pages.

The second direction is to further investigate original data recoverability after sanitization is performed. Whereas our evaluation establishes a full sanitization by observing the states of individual cells, it is conceivable that more sophisticated attacks may try to uncover the differences in threshold voltages between cells that were originally in a given state and cells that are moved into the given state through the sanitization. This evaluation requires a detailed flash cell characterization.

The third direction requires investigation of reliability issues in worn-out pages. Our experiments mainly focused on fresh memory blocks and more rigorous evaluation is needed on worn-out blocks.

The fourth direction involves analyzing subpage sanitization. With subpage sanitization a portion of a page is sanitized while the rest of it retains the original data. We have performed a set of experiments with subpage sanitization and they indicate that this approach is feasible in all combinations discussed in this paper. However, more detailed experiments are needed to investigate its robustness.

5 CONCLUSION

This paper describes an instant page sanitization method for MLC flash memories with minimal impact on the corresponding shared memory page. The proposed method does not affect memory endurance (no block erase operation is used), does not reduce the available memory space, and can be implemented within the standard FTL with a minimal firmware change.

6 ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable suggestions. This work was supported in part by U.S. National Science Foundation (NSF) grant CNS-2007403. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Bo Chen and Radu Sion. 2015. HiFlash: A History Independent Flash Device. *CoRR* abs/1511.05180 (2015). arXiv:1511.05180 http://arxiv.org/abs/1511.05180

[2] Feng Chen, Tong Zhang, and Xiaodong Zhang. 2017. Software Support Inside and Outside Solid-State Devices for High Performance and High Efficiency. *Proc. IEEE* 105, 9 (2017), 1650–1665. https://doi.org/10.1109/JPROC.2017.2679490

[3] Sarah Diesburg, Christopher Meyers, Mark Stanovich, Michael Mitchell, Justin Marshall, Julia Gould, An-I Andy Wang, and Geoff Kuenning. 2012. TrueErase: Per-File Secure Deletion for the Storage Data Path. In *Proceedings of the 28th Annual Computer Security Applications Conference* (Orlando, Florida, USA) *(ACSAC '12)*. Association for Computing Machinery, New York, NY, USA, 439–448. https://doi.org/10.1145/2420950.2421013

[4] Understanding Flash. 2016. *The Fall and Rise of Flash Memory.* Retrieved February 26, 2022 from https://flashdba.com/tag/understanding-flash/

[5] Md Mehedi Hasan and Biswajit Ray. 2020. *Data Recovery from "Scrubbed" NAND Flash Storage: Need for Analog Sanitization.* USENIX Association, USA. https://dl.acm.org/doi/10.5555/3489212.3489291

[6] Michael Hill. 2019. *Report: 42% of Used Drives Sold on eBay Hold Sensitive Data.* Retrieved February 26, 2022 from https://www.infosecurity-magazine.com/news/used-drives-sold-sensitive-data-1-1/

[7] Shijie Jia, Luning Xia, Bo Chen, and Peng Liu. 2016. NFPS: Adding Undetectable Secure Deletion to Flash Translation Layer. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16)*. Association for Computing Machinery, New York, NY, USA, 305–315. https://doi.org/10.1145/2897845.2897882

[8] Myungsuk Kim, Jisung Park, Genhee Cho, Yoona Kim, Lois Orosa, Onur Mutlu, and Jihong Kim. 2020. Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. Association for Computing Machinery, New York, NY, USA, 1311–1326. https://doi.org/10.1145/3373376.3378490

[9] Jaeheung Lee, Junyoung Heo, Yookun Cho, Jiman Hong, and Sung Y. Shin. 2008. Secure Deletion for NAND Flash File System. In *Proceedings of the 2008 ACM Symposium on Applied Computing* (Fortaleza, Ceara, Brazil) *(SAC '08)*. Association for Computing Machinery, New York, NY, USA, 1710–1714. https://doi.org/10.1145/1363686.1364093

[10] Jaeheung Lee, Junyoung Heo, Yookun Cho, Jiman Hong, and Sung Y. Shin. 2008. Secure deletion for NAND flash file system. In *Proceedings of the 2008 ACM symposium on Applied computing (SAC '08)*. Association for Computing Machinery, New York, NY, USA, 1710–1714. https://doi.org/10.1145/1363686.1364093

[11] Ping-Hsien Lin, Yu-Ming Chang, Yung-Chun Li, Wei-Chen Wang, Chien-Chung Ho, and Yuan-Hao Chang. 2018. Achieving fast sanitization with zero live data copy for MLC flash memory. In *Proceedings of the International Conference on Computer-Aided Design*. ACM, San Diego California, 1–8. https://doi.org/10.1145/3240765.3240773

[12] Yixin Luo, Saugata Ghose, Yu Cai, Erich F. Haratsch, and Onur Mutlu. 2018. HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature Awareness. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 504–517. https://doi.org/10.1109/HPCA.2018.00050

[13] Rino Micheloni and Piero Olivo. 2017. Solid-State Drives (SSDs) [Scanning the Issue]. *Proc. IEEE* 105, 9 (Sept. 2017), 1586–1588. https://doi.org/10.1109/JPROC.2017.2727228

[14] Sparsh Mittal and Jeffrey S. Vetter. 2016. A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems. *IEEE Transactions on Parallel and Distributed Systems* 27, 5 (May 2016), 1537–1550. https://doi.org/10.1109/TPDS.2015.2442980

[15] Expert Participation. 2018. *Data Protection Act.* Retrieved February 26, 2022 from https://www.legislation.gov.uk/ukpga/2018/12/section/1 Publisher: Statute Law Database.

[16] Joel Reardon, David Basin, and Srdjan Capkun. 2013. SoK: Secure Data Deletion. In *2013 IEEE Symposium on Security and Privacy*. 301–315. https://doi.org/10.1109/SP.2013.28

[17] Joel Reardon, David Basin, and Srdjan Capkun. 2014. On Secure Data Deletion. *IEEE Security Privacy* 12, 3 (May 2014), 37–44. https://doi.org/10.1109/MSP.2013.159

[18] Joel Reardon, Srdjan Capkun, and David Basin. 2012. Data Node Encrypted File System: Efficient Secure Deletion for Flash Memory. 333–348. https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/reardon

[19] Joel Reardon, Claudio Marforio, Srdjan Capkun, and David Basin. 2012. User-Level Secure Deletion on Log-Structured File Systems. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (Seoul, Korea) *(ASIACCS '12)*. Association for Computing Machinery, New York, NY, USA, 63–64. https://doi.org/10.1145/2414456.2414493

[20] Privacy for Sale. 2022. *Data Security Risks in the Second-Hand IT Asset Marketplace.* Retrieved February 26, 2022 from https://www.blancco.com/resources/rs-privacy-for-sale-data-security-risks-in-the-second-hand-it-asset-marketplace/

[21] Kyoungmoon Sun, Jongmoo Choi, Donghee Lee, and Sam H. Noh. 2008. Models and Design of an Adaptive Hybrid Scheme for Secure Deletion of Data in Consumer Electronics. *IEEE Transactions on Consumer Electronics* 54, 1 (Feb. 2008), 100–104. https://doi.org/10.1109/TCE.2008.4470030

[22] Wei-Chen Wang, Chien-Chung Ho, Yuan-Hao Chang, Tei-Wei Kuo, and Ping-Hsien Lin. 2018. Scrubbing-Aware Secure Deletion for 3-D NAND Flash. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (Nov. 2018), 2790–2801. https://doi.org/10.1109/TCAD.2018.2857260

[23] Michael Wei, Laura M. Grupp, Frederick E. Spada, and Steven Swanson. 2011. Reliably Erasing Data from Flash-Based Solid State Drives. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies* (San Jose, California) *(FAST'11)*. USENIX Association, USA, 8.

[24] Lorenzo Zuolo, Cristian Zambelli, Rino Micheloni, and Piero Olivo. 2017. Solid-State Drives: Memory Driven Design Methodologies for Optimal Performance. *Proc. IEEE* 105, 9 (Sept. 2017), 1589–1608. https://doi.org/10.1109/JPROC.2017.2733621