# *Efficient Sanitization Design for LSM-based Key-Value Store over 3D MLC NAND Flash*

# Efficient Sanitization Design for LSM-based Key-Value Store over 3D MLC NAND Flash

Liang-Chi Chen
National Cheng Kung University
Tainan, Taiwan
p76114391@gs.ncku.edu.tw

Shu-Qi Yu
National Taiwan University
Taipei, Taiwan
r11922038@csie.ntu.edu.tw

Chien-Chung Ho
National Cheng Kung University
Tainan, Taiwan
ccho@gs.ncku.edu.tw

Wei-Chen Wang
Massachusetts Institute of Technology
Massachusetts, United States
wweichen@mit.edu

Yung-Chun Li
Macronix International Co., Ltd.
Hsinchu, Taiwan
monixslee@mxic.com.tw

## ABSTRACT

Conventional LSM tree designs delete data by inserting a delete mark to the specified key, and they thus it leaves several out-of-date values to the specified key on the LSM tree. As a result, the LSM tree encounters a serious data security issue due to the undeleted values when there arises the need for data sanitization. *Sanitization* is a time-consuming process that involves completely removing sensitive data from storage devices. Flash-based SSDs are widely used in many systems, but they lack an in-place update feature, which makes it difficult for LSM trees to maintain both privacy and performance on these devices. This work proposes an efficient sanitizable LSM-tree design for LSM-based key-value store over 3D NAND flash memories. Our proposed efficient sanitizable LSM-tree design focuses on integrating the processes of key-value pair updating and the execution of sanitization by exploiting our proposed influence-conscious programming method. The capability of the proposed design is evaluated by a series of experiments, for which we have very encouraging results.

## CCS CONCEPTS

• **Computer systems organization** → **Firmware**; • **Security and privacy** → *Data anonymization and sanitization*;

## KEYWORDS

LSM tree, Flash memory, SSD, sanitization, secure deletion

**ACM Reference Format:**
Liang-Chi Chen, Shu-Qi Yu, Chien-Chung Ho, Wei-Chen Wang, and Yung-Chun Li. 2023. Efficient Sanitization Design for LSM-based Key-Value Store over 3D MLC NAND Flash. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-March 31, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3555776.3577780

## 1 INTRODUCTION

Key-value stores based on Log-Structured Merge-Tree (LSM-tree) [10] are widely adopted to manage large-scale data in many modern data-intensive applications (e.g., RocksDB [3]). Although the design of LSM-tree significantly improves the efficiency of storage devices, it also leads to new challenges in securing sensitive data. As data security has risen to be one of the most critical concerns, data sanitization techniques have been proposed to meet the need to completely remove sensitive data from storage devices physically. However, deleting data in LSM-tree is done by inserting a delete mark to the specified key, and it thus leaves several out-of-date values in the specified key. Those out-of-date data are still obtainable on the storage device before they are deleted by compaction operations, and thus leave the risk of being stolen and referenced illegally. Meanwhile, 3D NAND flash-based devices, i.e., solid-state drives (SSDs) are widely deployed to speed up the system performance. *Since flash memory lacks the in-place-update feature, it is not trivial to implement sanitization on flash-based storage systems.* The situation could deteriorate further as LSM-based key-value stores are deployed on SSDs. Although lots of existing studies have addressed the sanitization designs for flash memories [5, 8, 11, 13], they cannot maintain the performance of LSM-tree management due to additional sanitization overheads. Thus, The objective of this work is to enable sanitizable LSM-tree design for LSM-based key-value stores over 3D NAND flash memories. To improve the sanitization efficiency, we propose a novel influence-classification programming method to sanitize out-of-date values. In addition, a hot-cold separation allocation policy is proposed to improve space utilization.

## 2 BACKGROUND AND MOTIVATION

### 2.1 Log-Structured Merge-Trees

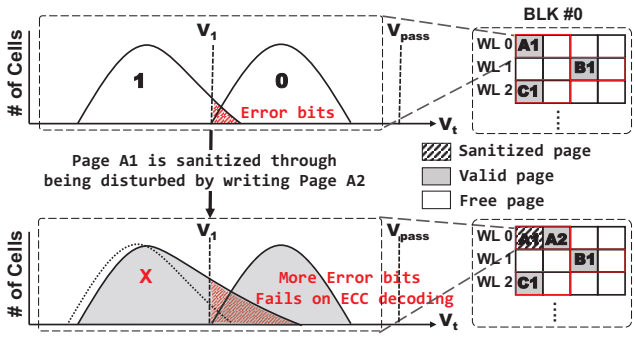
SSDs have been widely deployed to speed up the system's performance in recent years. To fully utilize the parallelism of SSDs, a separating structure of LSM-tree, named *WiscKey*, was proposed to reduce the write amplification by eliminating the need for value compaction. WiscKey proposed to separate the value from the key-value entry [9]. While the value is inserted into a value log, the key with the value offset pair is inserted into the existing LSM-tree. When it needs to insert a new key-value pair into the LSM-tree, the corresponding inserted value is appended to an on-SSD sequential value log. After that, the key and value address pair is added to the in-memory MemTable, which is sorted by keys. Once in-memory MemTable reaches its size limit, it will be switched to immutable MemTable and written to on-SSD *Level* 0, as an SSTable. When any level *Level k* reaches its size limit, a compaction procedure with applying the merge sort is triggered to merge SSTable files in between the two adjacent levels (*Level k* and *Level k* + 1).

In the key-value separating LSM-tree, the data deletion to a specific value might be delayed due to several reasons. First, the data stored on the device level are not erased immediately right after

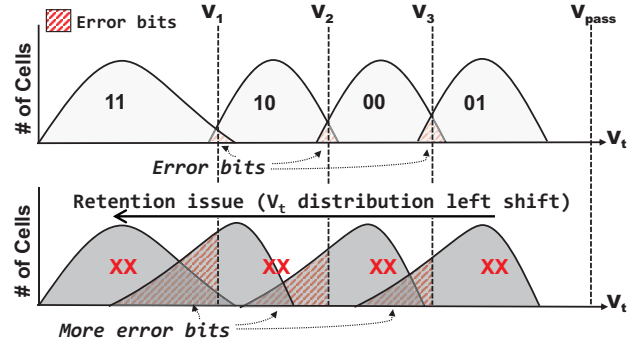

*SAC '23, March 27-March 31, 2023, Tallinn, Estonia*

ACM ISBN 978-1-4503-9517-5/23/03.
https://doi.org/10.1145/3555776.3577780

any key-value pair is deleted. Second, deleting data in LSM-tree is proceeded by inserting a delete mark to the specified key, the deleted value will not be deleted until the compaction process or specific garbage collection. The garbage collection of Wiskey [9] deletes out-of-date values that do not have a corresponding key in the LSM-tree. This garbage collection is a software process that is completely different from the general GC in flash-based storage. As security becomes one of the most important concerns, it is necessary to explore an efficient and effective sanitization design for LSM-based key-value stores over 3D flash-based SSDs.




(a) How instantaneous sanitization sanitizes out-of-date $A1$ as $A2$ is written to the flash memory.




(b) Directly adopting the instantaneous sanitization on 3D MLC flash memory can lead to the retention error issue.

**Figure 1: Instantaneous sanitization and its potential issue on 3D MLC flash.**

## 2.2 Instantaneous Sanitization

Taking the inherently high rate of insertion and deletion on LSM-tree and SSD's characteristics into consideration, the conventional rewriting-based [5, 8, 13] and erasure-based [11] sanitization methods are not suitable for LSM-based key-value store over 3D flash-based SSDs. The additional live-page copying or migration overhead is needed for SSDs to prevent data from being disturbed, and it thus competes for I/O bandwidth with the normal management and execution of LSM-tree. To speed up the sanitization process, the *instantaneous sanitization (IS)* method, which exploits the disturbance effect to sanitize the adjacent pages [14], shows its great potential to enable the sanitize feature for LSM-based key-value store over 3D flash-based SSDs. As shown in Figure 1(a), $A1$ page is a 3D SLC flash page to store the first version of data $A$ with the purposely generated overlapping on $V_t$ distribution. When data $A$ is updated, the new version of $A$ will be programmed to the adjacent page $A2$. When page $A2$ is programmed, it can disturb the $V_t$ distribution of page $A1$ and lead the number of error bits in page $A1$ to exceed the capability of error correction code (ECC). That is, the out-of-date version of $A$ stored on page $A1$ is sanitized. Compared to the conventional sanitization methods, the IS does not incur the additional blocks erase or live-page-copying overheads.

## 2.3 Research Motivation

In this work, we dedicate to designing a sanitization design for 3D MLC flash memory since its higher density. Since the IS is designed based on the property of 3D SLC flash memories, it can lead to a potential issue when it is deployed on 3D MLC flash memories. As illustrated in Figure 1(b), if the idea is directly adopted to deliberately create a certain number of disturbance-induced errors on a 3D MLC flash page in the beginning stage while it is programmed, the 3D MLC flash page will suffer from the serious retention error issue when the time expires; in other words, the reliability of MLC is weaker than that of SLC [4]. Moreover, using the IS can lead to space utilization issue due to the constraint of its corresponding allocation method. The problem can be simply explained with the example shown in Figure 1(a). For example, the two remaining free pages in wordline ($WL$) 0 cannot be allocated until pages $A2$ and $B1$ are sanitized. Otherwise, programming data to the two remaining free pages in $WL0$ can result in sanitizing pages $A2$ and $B1$ unintentionally. Therefore, there will be difficulty in how alleviating the problem of space utilization when we try to have efficient sanitization. To realize an efficient sanitizable LSM-tree design, we aim at proposing a novel *influence-classification programming* method which simultaneously considers the property of IS and $V_t$ distribution characteristics of 3D MLC flash memories. In the following sections, we shall propose to consider the properties of LSM-tree and 3D MLC flash memory jointly.

# 3 EFFICIENT SANITIZABLE LSM-TREE

## 3.1 System Architecture Overview

The system architecture of our proposed efficient sanitizable LSM-tree design. All the lookup (i.e., read), insert (i.e., write), and delete operations are issued by LSM-tree. Those requests are handled by the flash translation layer (FTL). All the 3D MLC flash blocks are partitioned into two regions, the *value region* and *key region*. Keys are only used for indexing, and values are a core factor for achieving data security. It is more important to sanitize values instead of keys in LSM-trees in most situations. As a result, FTL uses blocks of the value region and blocks of the key region to store the value data and key data of the inserted key-value pairs respectively. Blocks of value region are further partitioned into *hot area* blocks and *cold area* blocks. All the inserted key-value pairs will be classified into hot and cold data and then stored in hot and cold areas accordingly.

To sanitize the old versions of data, a sanitizer in FTL examines the hotness of corresponding key-value pairs and applies the different methods to sanitize the correspondingly out-of-date values. If any inserted key-value pair is identified as hot data (i.e., the last value version is stored on the hot area), our proposed *influence-classification programming* method will be applied to write the new version value and sanitize out-of-date value simultaneously and instantaneously. If the inserted key-value pair is identified as the cold data (i.e., the last value version is stored on the cold area), the data allocator first writes the new value to a new page with the conventional ISPP [2], and then sanitizes the page where contain the

old-version value by applying the overwriting-based sanitization method [13].

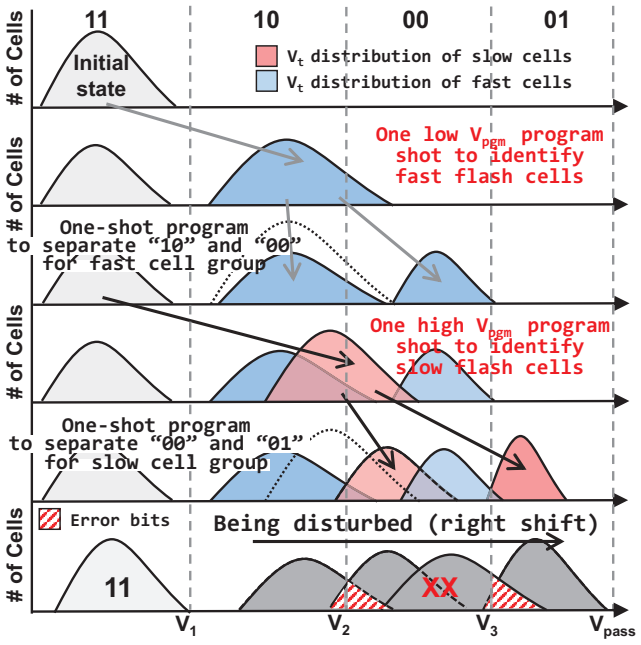



**Figure 2: Processes of the proposed ICP method.**

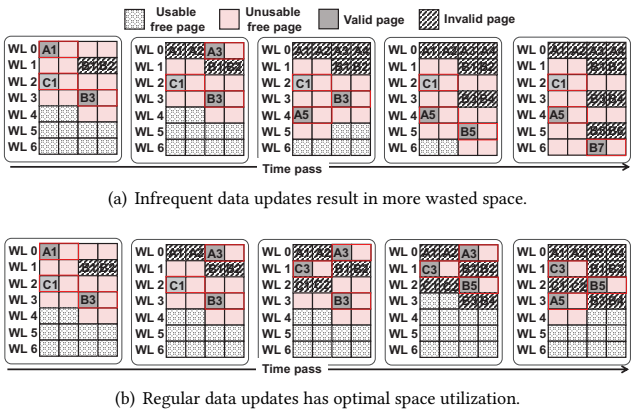
## 3.2 Influence-Classification Programming

This section shows how the proposed influence-classification programming (ICP) method works. *The objective of ICP is to enable IS for 3D MLC NAND flash so that the older version value of a specific key-value pair can be sanitized as the new version value of it is written.* Due to the imperfection in the process of 3D NAND manufacture, the tendency degree of a programmed/disturbed flash cell could be different [1]. Those cells that are much easy (resp. difficult) to be programmed/disturbed are referred to as fast (resp. slow) cells. It had been reported in existing studies that the programmability of flash cells could be evaluated by exploiting a low voltage programming shot [6, 7].

By considering cells' programmability and error properties, the proposed ICP constructs an ideal $V_t$ distribution that can not only achieve efficient sanitization but also mitigate data retention and disturbance issues. As shown in Figure 2, the proposed ICP programs 3D MLC flash cells and deliberately creates the proposed ICP pattern which prevents cells' $V_t$ distribution from being affected due to the retention error. This is because our proposed ICP method programs fast cells to the higher $V_t$ state so as to leave the wider $V_t$ window. As a result, the fast cells can have more flexibility to avoid the retention error issue, and slow cells can exploit their property to keep in the current $V_t$ state as long as possible. The reason why the ICP pattern is an appropriate $V_t$ distribution for enabling efficient sanitization over 3D MLC NAND is that fast cells are programmed to the right in each $V_t$ window, and slow cells are programmed to the left in each $V_t$ window. With such an approach, we could decrease the probability of retention error since slow cells are difficult to be erred beyond the capability of ECC. Meanwhile, the effectiveness of efficient sanitization can be guaranteed since fast cells are still easy to be disturbed.

## 3.3 Hot-Cold Separation Allocation Policy

This section presents *hot-cold separation allocation policy (HC)*. The objective is to appropriately allocate data in the value region for achieving high sanitization efficiency and low management overhead; because the proposed ICP sanitizes the out-of-date version and writes the new version instantaneously and simultaneously, it may disturb other valid values stored on the adjacent pages.




(a) Infrequent data updates result in more wasted space.

(b) Regular data updates has optimal space utilization.

**Figure 3: The impact of inconsistent data update patterns to space utilization in a block which sanitizes data with ICP.**

As illustrated in Section 3.1, the cold data are programmed and sanitized by the general ISPP and overwriting-based sanitization method, respectively. *The proposed HC focuses on monitoring the space utilization on blocks of the value region and keeping data in the same block that has similar update patterns.* As shown in Figure 3, the space utilization can be improved if the data update patterns within a block are more similar. This phenomenon highly depends on the access pattern and insertion order of key-value pairs, and it can affect the number of remaining usable pages. For example, with inserting the different 13 key-value pairs, the numbers of remaining usable free pages in Figure 3(a) and Figure 3(b) are 0 and 10 respectively.

# 4 EXPERIMENT

## 4.1 Experiment Setup

In this section, we examine the performance of the proposed design in terms of the number of reads/writes/erases and memory space utilization. All experiments are built on MQSim [12], and a 3D MLC NAND flash memory with a 4GB chip size is adopted as the experimental device. The latencies of page programming, page read, and block erasure are set to 75$\mu$s, 750$\mu$s, and 3.8ms, respectively.

In the experiments, four compared approaches (*Erasure-base (ER), Scrubbing (SC), ICP, ICP w/ hot-cold separation(ICPHC)*) are implemented. *ER* is the conventional erasure-based method that adopts the built-in erase function in 3D NAND to sanitize data whenever the data update request comes. This needs to copy all the remaining valid pages to other blocks before applying the erase operation to sanitize the desired data. *SC* denotes the overwriting-based method (i.e., Scrubbing [13]). It needs to copy all adjacent valid pages to other blocks before the scrubbing operation is applied to the to-be-sanitized pages. *ICP* is our proposed ICP to program and sanitize data. *ICP w/ hot-cold* is the proposed ICP method supported

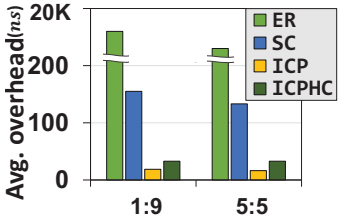



**Figure 4: Sanitization performance results**

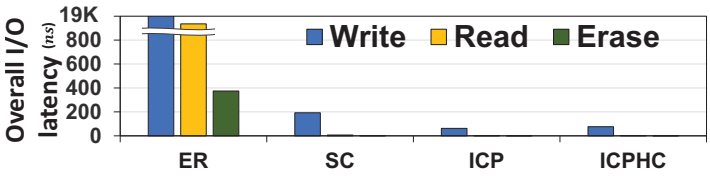



**Figure 5: Overall performance under hot/cold ratio 1:9 workload. Hot-cold ratio 5:5 workload is skipped because it has same trend with 1:9.**

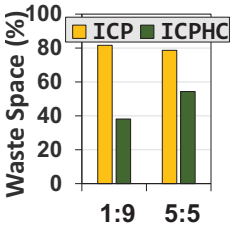



**Figure 6: Space utilization analysis.**

with a hot-cold separation allocation policy. We also generate different workloads with different types of hot/cold proportions (i.e., 1:9, 5:5) as our data benchmarks in order to understand how our design works in different situations.

## 4.2 Experiment Results

Figure 4 shows the sanitization performance of the four investigated sanitization methods under different workloads. The x-axis denotes the four sanitization methods, and the y-axis denotes the sanitization performance in terms of the average sanitization time. Note that the sanitization time is defined as the time to completely finish processes to sanitize an out-of-date version value stored on the 3D MLC flash memory page. If the sanitization needs to migrate the live pages and erase the corresponding block, the time delay caused by these additional operations will also be involved in the sanitization time. According to the experimental results, it is observed that the sanitization performance of the erasure-based method and scrubbing are poor in every situation due to the need for live-page copyings. On the contrary, compared to the scrubbing, our proposed ICP method can improve the sanitization performance by 8.3× and 8.2× under the hot-cold ratio of 1:9 and 5:5, respectively. It is because our proposed ICP method sanitizes data by exploiting programming disturbance and would not incur any additional process (i.e. scrubbing or erasing) or live-page copying. Furthermore, Figure 5 demonstrates the overall performance of the four investigated sanitization methods under a hot-cold ratio of 1:9 workload. Our proposed ICP method is able to outperform the scrubbing method by 3.13×, and the performance of the erasure-based method is still the worst. The trend of overall performance is consistent with that of sanitization performance.

Figure 6 shows the space utilization analysis about applying the proposed hot-cold separation allocation policy. The x-axis denotes different hot/cold data ratios, and the y-axis denotes the wasted space utilization over 3D MLC NAND flash memory. The wasted space utilization is defined as the ratio of unavailable space to all the space, where the unavailable space includes all the valid pages, all the invalid pages, and those free pages that cannot be used due to the ICP restriction. It is found that the proposed ICP with the hot-cold separation allocation policy can significantly reduce the wasted space by 43.5% and 24.3% respectively. The reason is that the proposed hot-cold separation allocation policy monitors the space utilization and keeps data in the same block that has similar update patterns.

## 5 CONCLUSION

This paper aims at resolving the data security for LSM-based key-value separating storage on 3D MLC NAND flash memories. The ICP is used to write the new version value and sanitize out-of-date values simultaneously. The proposed hot-cold separation allocation policy focuses on improving space utilization by managing data with access patterns. As a result, updating data in the value region can achieve high sanitization efficiency and low management overhead. The results of our study demonstrate the effectiveness of the design in enhancing the security and efficiency of LSM-based storage on 3D MLC NAND flash memories.

## ACKNOWLEDGMENTS

This work was supported in part by the National Science and Technology Council, Taiwan under grant no. 109-2221-E-006-215-MY3.

## REFERENCES

[1] Yu Cai, Yixin Luo, Erich F. Haratsch, Ken Mai, and Onur Mutlu. 2015. Data retention in MLC NAND flash memory: Characterization, optimization, and recovery. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 551–563. https://doi.org/10.1109/HPCA.2015.7056062

[2] Hang-Ting Lue et al. 2008. study of incremental step pulse programming (ISPP) and STI edge effect of BE-SONOS NAND Flash. In *2008 IEEE International Reliability Physics Symposium*.

[3] Facebook. 2015. RocksDB. http://rocksdb.org/

[4] Laura M Grupp, John D Davis, and Steven Swanson. 2012. The bleak future of NAND flash memory.. In *FAST*, Vol. 7. 10–2.

[5] Md Mehedi Hasan and Biswajit Ray. 2020. Data Recovery from {"Scrubbed"}{NAND} Flash Storage: Need for Analog Sanitization. In *29th USENIX Security Symposium (USENIX Security 20)*. 1399–1408.

[6] Chien-Chung Ho, Yung-Chun Li, Yuan-Hao Chang, and Yu-Ming Chang. 2018. Achieving Defect-Free Multilevel 3D Flash Memories with One-Shot Program Design. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC.2018.8465858

[7] Chih-Chang Hsieh, Hang-Ting Lue, Yung Chun Li, Ti Wen Chen, Hsiang-Pang Li, and Chih-Yuan Lu. 2015. A novel dichotomic programming algorithm applied to 3D NAND flash. In *2015 Symposium on VLSI Technology (VLSI Technology)*. T180–T181. https://doi.org/10.1109/VLSIT.2015.7223669

[8] 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*. 305–315.

[9] Lanyue Lu, Thanumalayan Sankaranarayana Pillai, Hariharan Gopalakrishnan, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. 2017. Wisckey: Separating keys from values in ssd-conscious storage. *ACM Transactions on Storage (TOS)* 13, 1 (2017), 1–28.

[10] Patrick O'Neil, Edward Cheng, Dieter Gawlick, and Elizabeth O'Neil. 1996. The log-structured merge-tree (LSM-tree). *Acta Informatica* 33, 4 (1996), 351–385.

[11] Steven Swanson and Michael Wei. 2010. Safe: Fast, verifiable sanitization for ssds. *San Diego, CA: University of California-San Diego* (2010).

[12] Arash Tavakkol, Juan Gómez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu. 2018. MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices. In *16th USENIX Conference on File and Storage Technologies (FAST 18)*. 49–66.

[13] 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 (2018), 2790–2801.

[14] Wei-Chen Wang, Ping-Hsien Lin, Yung-Chun Li, Chien-Chung Ho, Yu-Ming Chang, and Yuan-Hao Chang. [n. d.]. Toward Instantaneous Sanitization through Disturbance-induced Errors and Recycling Programming over 3D Flash Memory. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.