NS-FTL: Alleviating the Uneven Bit-Level Wearing of NVRAM-based FTL via NAND-SPIN

Wei-Chun Cheng, Shuo-Han Chen, Yuan-Hao Chang, Kuan-Hsun Chen, Jian-Jia Chen, Tseng-Yi Chen, Ming-Chang Yang, Wei-Kuan Shih
Institute of Information Science, Academia Sinica, Taipei, Taiwan
TU Dortmund, Department of Computer Science, Dortmund, Germany
Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan
The Chinese University of Hong Kong, Department of Computer Science and Engineering, Hong Kong
National Tsing Hua University, Department of Computer Science, Hsinchu, Taiwan

Citation: 10.1109/NVMSA51238.2020.9188172

BibTeX:

```
@inproceedings{9188172,
  author={W. -C. {Cheng} and S. -H. {Chen} and Y. -H. {Chang} and K. -H. {Chen}
  and J. -J. {Chen} and T. -Y. {Chen} and M. -C. {Yang} and W. -K. {Shih}},
  booktitle={2020 9th Non-Volatile Memory Systems and Applications Symposium (NVMSA)},
  title={NS-FTL: Alleviating the Uneven Bit-Level Wearing of NVRAM-based FTL via NAND-SPIN},
  year={2020},
  volume={},
  number={},
  pages={1-6},
  doi={10.1109/NVMSA51238.2020.9188172}
}
```

NS-FTL: Alleviating the Uneven Bit-Level Wearing of NVRAM-based FTL via NAND-SPIN

Wei-Chun Cheng, Shuo-Han Chen, Yuan-Hao Chang, Kuan-Hsun Chen*, Jian-Jia Chen*,
Tseng-Yi Chen†, Ming-Chang Yang‡, Wei-Kuan Shih§
Institute of Information Science, Academia Sinica, Taipei, Taiwan
*TU Dortmund, Deaprtment of Computer Science, Dortmund, Germany
†Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan
‡The Chinese University of Hong Kong, Department of Computer Science and Engineering, Hong Kong
§National Tsing Hua University, Department of Computer Science, Hsinchu, Taiwan
e-mail: {weichun, qoolili, johnson}@iis.sinica.edu.tw, *{kuan-hsun.chen,jian-jia.chen}@cs.uni-dortmund.de,
†tychen@g.ncu.edu.tw, ‡mcyang@cse.cuhk.edu.hk, §wshih@cs.nthu.edu.tw

***Abstract*—Non-Volatile random access memory (NVRAM) has been regarded as a promising DRAM alternative with its nonvolatility, near-zero idle power consumption, and byte addressability. In particular, some NVRAM devices, such as Spin Torque Transfer (STT) RAM, can provide the same or better access performance and lower power consumption when compared with dynamic random access memory (DRAM). These nice features make NVRAM become an attractive DRAM replacement on NAND flash storage for resolving the management overhead of the flash translation layer (FTL). For instance, when adopting NVRAM for storing the mapping entries of FTL, the overheads of loading and storing the mapping entries between the non-volatile NAND flash and the volatile DRAM can be eliminated. Nevertheless, due to the limited lifetime constraint of NVRAM, the bit-level update behavior of FTL may lead to the issue of uneven bit-level wearing and the lifetime capacity of those less-worn NVRAM cells could be underutilized. Such an observation motivates this study to utilize the emerging NAND-like Spin Torque Transfer memory (NAND-SPIN) for alleviating the uneven bit-level wearing of NVRAM-based FTL and making the best of the lifetime capacity of each NAND-SPIN cell. The experimental results show that the proposed design can effectively avoid the uneven bit-level wearing, when compared with page-based FTL on NAND-SPIN.**

***Index Terms*—NAND-SPIN, FTL, NAND Flash, NVRAM**

I. Introduction

Flash translation layer (FTL) has been used extensively to manage the constraints of NAND flash memory on NAND flash storage devices. FTL remaps write requests to writable free space and records the mapping between logical address space and physical address space as logical-to-physical mapping entries. These mapping entries are loaded into dynamic random access memory (DRAM) for updates and stored back to NAND flash for persistence. Vulnerabilities of DRAM-based FTL include load/store overhead and possible data loss during sudden power outages. To resolve these issues, non-volatile random access memory (NVRAM) has been investigated to replace DRAM for hosting FTL with its nonvolatility and similar-to-DRAM access performance [3, 4, 9]. Examples of NVRAM include Phase Change Memory (PCM) and Spin Torque Transfer (STT) RAM. The major design consideration of NVRAM-based FTL is the lifetime constraint of NVRAM. In particular, the bit-level update pattern during mapping entry updates could cause uneven bit-level wearing to NVRAM cells. When part of the cells in a mapping entry wear out prematurely, the lifetime capacity of other less-worn cells in the same entry is underutilized. Even though previous researchers [4] have tried to distribute the updates of mapping entries evenly across the whole NVRAM for avoiding uneven inter-entry wearing, *the issue of uneven intra-entry bit-level wearing receives much less attention.*

To alleviate the issue of uneven bit-level wearing, this study proposes to utilize the emerging NAND-like Spin Torque Transfer memory (NAND-SPIN) [10] as the underlying NVRAM for FTL. NAND-SPIN, which is evolved from STT-RAM and Spin Orbit Torque (SOT) RAM, has shorter access latency and provides higher read/write throughput, when compared with DRAM. In addition, NAND-SPIN has a higher areal density than both STT-RAM and SOT-RAM for providing larger capacity with the same die area. NAND-SPIN allows bit-level programs[1] to alter individual cells from 0 to 1. However, unlike previous NVRAMs, changing NAND-SPIN cells from 1 to 0 needs to be carried out in the unit of string and is referred to as the *string-based erase*. Note that a NAND-SPIN string is composed of multiple NAND-SPIN cells, each of which stores one bit.

The unique string-based erase of NAND-SPIN provides the possibility of alleviating the uneven bit-level wearing issue of NVRAM-based FTL. This is because, according to our investigation, the uneven bit-level wearing is mainly attributed to the unpredictability of bit-level patterns during FTL mapping updates, and string-based erases can partially resolve the unpredictability by ensuring cells of the same string receive the same number of erases. Nevertheless, since changing a cell from 1 to 0 requires whole-string erases, directly deploying conventional FTL [1] on NAND-SPIN could lead to excessive erases and cause uneven wearing between mapping entries. In summary, the technical difficulty of hosting FTL on NAND-SPIN lies in *how to alleviate both the uneven inter- and intra-entry wearing while minimizing the number of string-based erases.*

To utilize the nice features of NAND-SPIN for NVRAM-based FTL, this study proposes a NAND-SPIN FTL (NS-

[1]In this study, the term "program" and "write" are used interchangeably for referring to the operation of changing the value in an NVRAM cell.

FTL) to alleviate both the uneven inter-entry and intra-entry bit-level wearing issue of NVRAM-based FTL and make the best of the lifetime capacity[2] of every NAND-SPIN cell. NS-FTL alleviates the uneven bit-level wearing by trying to ensure cells of a NAND-SPIN string can all be programmed through bit-level programs before string-based erases. Such an approach makes cells of the same NAND-SPIN string have similar numbers of programs and erases. In addition, this approach also allows mapping entries to be updated multiple times by changing bits from 0 to 1 with bit-level programs and avoids 0-to-1 updates to minimize the number of string-based erases. NS-FTL introduces three different components: (1) the inverted mapping strategy, (2) the zig-zag space allocator, and (3) the dual rotation wear leveler. The inverted mapping strategy is firstly used to minimize the number of '1' bits during updates by flipping the bits of to-be-program mapping entry, when the number of to-be-program '1' bits is more than that of '0' bits. Next, the zig-zag space allocator is included to manipulate the allocation strategy of NAND flash storage for enhancing the possibility of the 0-to-1 update pattern. Finally, the dual rotation wear leveler is utilized to alleviate the condition of uneven inter-entry wearing.

The contributions of this study can be highlight as follows:

- This study is a pioneer design proposed to resolve the uneven bit-level wearing issue of NVRAM-based FTL.
- The proposed NS-FTL achieves its design goals by allowing an FTL mapping entry to receive multiple updates through the bit-level program feature of NAND-SPIN via maximizing the possibility of the 0-to-1 update pattern with the proposed components.
- Experimental results show that the proposed NS-FTL can effectively reduce the standard deviation of the bit-level program/erases by up to 88.86%, when compared with naively deploying page-based FTL [1] on NAND-SPIN.

The rest of this study is organized as follows. The background and motivation are described in Section II. The design of NS-FTL is then introduced in Section III. Next, the evaluation results are explained in Section IV. Finally, Section V presents the concluding remarks of this study.

II. Background and Motivation

A. NVRAM-based FTL

NAND flash memory has become the mainstream storage medium in computer systems with its high access (read and write) speed and shock resistance. Nevertheless, NAND flash has several management constraints, including the erase-before-write property, asymmetric program/erase (P/E) units, and limited P/E cycles. For instance, a NAND flash block is the minimum erase unit and consists of a fixed number of pages, while a NAND flash page is the smallest read/write unit. To manage the constraints of NAND flash, FTL is utilized to redirect write requests to the free (erased) area and maintains logical-to-physical mapping entries. As shown in Figure 1, when NAND flash storage revives data updates,

[2]Lifetime capacity refers to the total number of writes that an NVRAM cell can endure before worn out.

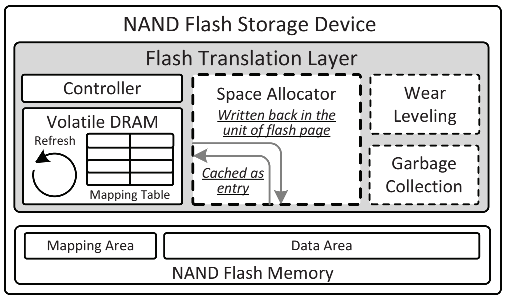

Fig. 1. **Conventional DRAM-based FTL**, *in which mapping entries are loaded onto DRAM for updates and stored back to NAND flash for persistence. Notably, the modules of space controller, wear leveling, and garbage collection are used to manage the constraints of NAND flash.*

mapping entries are loaded from NAND flash pages into DRAM for mapping entries updates. Then, those updated entries are stored back to NAND flash pages for persistence.

These load and store operations of mapping entries induce internal management overhead on NAND flash devices. In addition, due to the size difference between a mapping entry and the minimal access unit of NAND flash, storing mapping entries back to NAND flash also induces the problem of write amplification. For instance, even when only few bits are updated for a mapping entry update, a whole flash page needs to be written for preserving those updated bits. To avoid the write amplification and lower the management overhead of FTL, NVRAM has been considered to replace DRAM and used to store mapping entries of FTL.

As the primary design issue of NVRAM-based FTL is the lifetime constraint of NVRAM, studies have been proposed to avoid intensive mapping updates. For instance, Liu et al. [4] proposed a write-activity-aware PCM-assisted flash memory management scheme, also known as PCM-FTL, to reduce the maximum number of bit flips on the PCM. PCM-FTL employs page-based mapping for random writes, block-based mapping for sequential writes, and an inter-entry wear leveler. Nevertheless, the bit-level update behaviors of FTL are not considered within the design of PCM-FTL. In particular, *uneven bit-level wearing may lead to premature wearing to some of the NVRAM cells in a mapping entry and leave the lifetime capacity of the rest NVRAM cells underutilized.*

B. NAND-SPIN

NAND-SPIN is regarded as a promising successor to STT-RAM and SOT-RAM with comparable access latency and

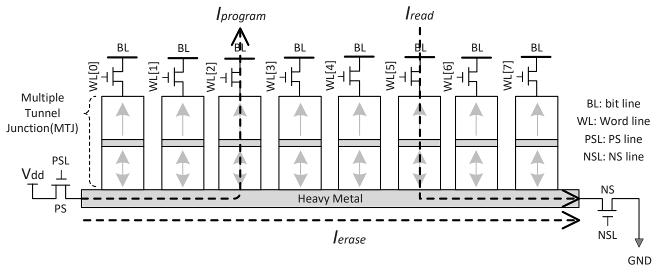

Fig. 2. **NAND-SPIN Memory**, *in which 2^n MTJs are integrated onto the same heavy-metal strip and share the same PMOS (PS) transistor. Compared with STT-RAM and SOT-RAM, NAND-SPIN can achieve higher density, better energy efficiency and faster access speed.*

better energy efficiency, as summarized in Table I. NAND-SPIN resolves the source degeneration[3] issue of STT-RAM via unidirectional program/erase current and achieves higher density than SOT-RAM via sharing the heavy metal between cells. As shown in Figure 2, NAND-SPIN connects 2^n Magnetic Tunnel Junction (MTJ) onto one heavy-metal strip. MTJ is the basic storage unit of NAND-SPIN memory and consists of 2 ferromagnetic layers, which are the *reference layer* with fixed magnetization direction and the *free layer* with alterable magnetization direction. The value stored in the MTJ cell is determined by the parallelity between the fixed and free layers. In this study, we define the parallel state as the value of 1 and the antiparallel state as the value of 0.

TABLE I
COMPARISON OF STT-RAM, SOT-RAM, AND NAND-SPIN [10].

Parameters	STT-RAM	SOT-RAM	NAND-SPIN
Size (F^2)	40.3	9*2	7.95
Read Latency (ns)	1.62	1.63	1.65
Write Latency (ns)	'0' to '1': 6 '1' to '0': 4	'0' to '1': 0.7 '1' to '0': 1	Erase: 1 Program: 4
Read Energy (fJ)	'0': 15.336 for '1': 16.285 for	'0': 15.987 '1': 16.78	'0': 19.173 '1': 20.134
Write Energy (fJ)	'0' to '1': 627 '1' to '0': 1387	'0' to '1': 178.6 '1' to '0': 127.2	Erase: 30.91 /bit Program: 369.7 /bit

On NAND-SPIN, the parallelity of an MTJ cell is changed through either erase operation or write operation. For erase operation, NAND-SPIN cells need to be erased to antiparallel state in the unit of string. Erase operation is achieved by activating the selection transistors (PS and NS) of the selected string and passing a charge current through the heavy-metal strip to generate the SOT (I_{erase} in Figure 2). Erased cells are in the antiparallel state and represent the value of 0. On the other hand, program operation switches bit-cells to parallel state with the value of 1 by activating the access transistors and PS transistors of those bit-cells to be switched and grounding bit-lines (BLs) for a current, which induces the STT, to flow from the free layer to reference layer of MTJs ($I_{program}$ in Figure 2). Meanwhile, read operations require the access transistors and NS transistors to be activated. Then, BL is connected to the sensing amplifier for sensing the resistance level of MTJs for determining the stored data.

Based on NAND-SPIN, previous studies [11, 12] mainly focus on reducing the number of erases when NAND-SPIN is utilized as the processor cache. For instance, Wu et al. [11] propose an adaptive buffer entry (ABE) write policy to adaptively extend the write data length under a fixed maximum supply current. Meanwhile, a non-volatile look-up table is proposed by Zhang et al. [12] to enable a shared random access module for fast configuration and readout operations. *The applicability of alleviating the uneven bit-level wearing on NVRAM via NAND-SPIN receives much less attention.*

C. Motivation

Designs of NVRAM-based FTL have been proposed to store the management data of FTL on NVRAM. However, most of previous NVRAM-based FTL designs [4] focus on reducing write traffic of FTL to reduce the number of overall mapping entry updates on NVRAM. *The problem of uneven bit-level wearing is neglected in previous studies.* The bit-level uneven wearing on NVRAM is induced by the fact that updates to mapping entries are conducted in bit level, and the number of writes to each NVRAM cell is not balanced.

With the emergency of NAND-SPIN memory, we observe the opportunity of alleviating the uneven bit-level wearing condition of NVRAM-based FTL via the string-based erases. String-based erases can naturally ensure that cells in the same string receive the same number of erases. In other words, if a mapping entry is stored on a single NAND-SPIN string, cells of a mapping entry will have the same number of erases. Nevertheless, string-based erases can not solely resolve the uneven bit-level wearing issue because the content of mapping entry updates is unpredictable. Thus, the number of programs on each cell can not be guaranteed. On the other hand, directly deploying conventional FTL [1] on NAND-SPIN could lead to excessive erases because updating a cell from 1 to 0 requires the string-based erase. In summary, *the major issue of alleviating the uneven bit-level wearing via NAND-SPIN lies in how to alleviate uneven intra-entry wearing and avoid uneven inter-entry wearing by minimizing the number of string-based erases.*

III. NAND-SPIN-BASED FLASH TRANSLATION LAYER

A. Overview

To revolve the uneven bit-level wearing of NVRAM-based FTL, this section presents a NAND-SPIN-based FTL (NS-FTL) to ensure bits of a NAND-SPIN string can revive similar numbers of programs and erases. NS-FTL realizes this design goal by allowing a mapping entry to receive multiple mapping updates with bit-level programs to change bits from 0 to 1 before string-based erases are used to change all bits in a NAND-SPIN string from 1 to 0. Therefore, as most bits are programmed before erased, the uneven wearing condition is diminished. To the best of our investigation, few studies have investigated the applicability of utilizing NAND-SPIN to revolve the uneven bit-level wearing issue of NVRAM-based FTL. The system architecture of the proposed NS-FTL is illustrated in Figure 3.

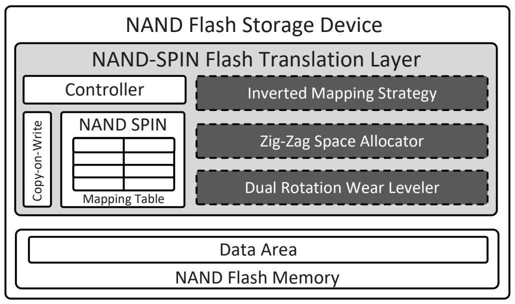
NAND Flash Storage Device
NAND-SPIN Flash Translation Layer
Controller
Inverted Mapping Strategy
Copy-on-Write
NAND SPIN
Mapping Table
Zig-Zag Space Allocator
Dual Rotation Wear Leveler
Data Area
NAND Flash Memory

Fig. 3. **System architecture of the proposed NS-FTL**, *in which mapping entries of the FTL are stored on NAND-SPIN and a small copy-on-write buffer is utilized to prevent inconsistency during updating a mapping entry.*

The proposed NS-FTL is designed as an FTL to manage both the mapping entries of FTL and the storage space of

[3]Source degeneration refers to the degraded drivability of the access transistor in STT-RAM due to the smaller gate-source bias when current flows from the source line (SL) to bit line (BL) [10].

NAND flash memory. NS-FTL is a page-based FTL and records the logical-to-physical mapping for each flash page. Each mapping entry is stored on a single NAND-SPIN string. In other words, a mapping entry can be erased independently without affecting other entries. During updating an existing entry, a small copy-on-write buffer is used to prevent inconsistency if a power failure happens during mapping entry updates. The core concept of NS-FTL is to allow a mapping entry to receive multiple updates without erases. To enforce this concept, NS-FTL first minimizes the number of '1' bits in mapping entry updates by the inverted mapping strategy (see Section III-B) to flip the bits of a mapping entry if the number of to-be-program '1' bits is larger than that of '0' bits. Secondly, to update a mapping entry multiple times without string-based erases, the zig-zag space allocator (see Section III-C) is included to maximize the possibility of the 0-to-1 update pattern. With the first two components, NS-FTL aims to alleviate the uneven intra-entry wearing. Finally, to avoid uneven inter-entry wearing, the dual-rotation wear leveler (see Section III-D) is included to periodically rotate the mapping entries for avoiding excessive updates to a small region of NAND-SPIN strings.

B. Inverted Mapping Strategy

Due to the unique physical layout of NAND-SPIN, updating a NAND-SPIN cell from 1 to 0 requires the string-based erase, which consumes the valuable lifetime of NAND-SPIN cells. To avoid excessive string-based erases, NS-FTL minimizes the number of '1' bits during updating mapping entries by flipping the mapping entry when the number of '1' bits is larger than that of '0' bits. Then, future mapping entry updates can be achieved via altering '0' bits to '1' bits with the bit-level programs of NAND-SPIN, thus lowering the number of string-based erases. The mechanism of inverted mapping strategy can be summarized as Figure 4. Note that an extra invert bit is included in each mapping entry to record the flip status and is set to 1 if the entry is inverted when stored on the NAND-SPIN string.

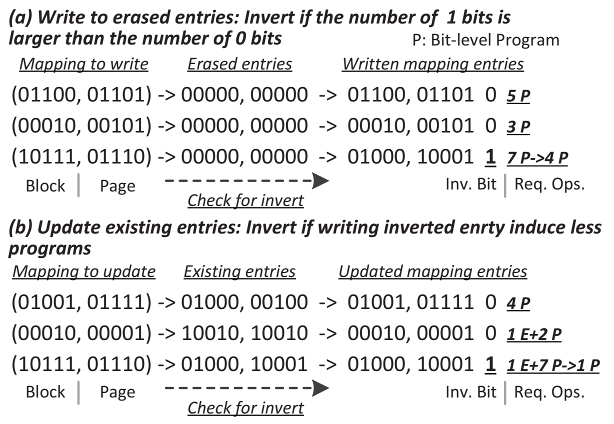
(a) Write to erased entries: Invert if the number of 1 bits is larger than the number of 0 bits
P: Bit-level Program
Mapping to write
Erased entries
Written mapping entries
(01100, 01101) -> 00000, 00000 -> 01100, 01101 0 5 P
(00010, 00101) -> 00000, 00000 -> 00010, 00101 0 3 P
(10111, 01110) -> 00000, 00000 -> 01000, 10001 1 7 P->4 P
Block
Page
Check for invert
Inv. Bit
Req. Ops.
(b) Update existing entries: Invert if writing inverted enrty induce less programs
Mapping to update
Existing entries
Updated mapping entries
(01001, 01111) -> 01000, 00100 -> 01001, 01111 0 4 P
(00010, 00001) -> 10010, 10010 -> 00010, 00001 0 1 E+2 P
(10111, 01110) -> 01000, 10001 -> 01000, 10001 1 1 E+7 P->1 P
Block
Page
Check for invert
Inv. Bit
Req. Ops.

Fig. 4. **The inverted mapping strategy**, *in which mapping entry updates are checked for counting the number of to-be-programmed '1' bits and inverted if the number of to-be-programmed '1' bits is larger than the number of to-be-programmed '0' bits.*

As shown in Figure 4, the inverted mapping strategy categorizes mapping entry updates into two different scenarios. The first scenario is writing to erased mapping entries. As shown in Figure 4(a), mapping entries are inverted if the number of to-be-programmed '1' bits is larger than that of to-be-programmed '0' bits. For instance, when writing the mapping entry (10111, 01110) to an erased NAND-SPIN string, the mapping will be inverted for inducing less 0-to-1 programs. Such an approach makes most of the bits to be '0' after initial entry updates and lowers the possibility of the 1-to-0 update pattern during future mapping entry updates for avoiding string-based erases, as most of the bits are left as 0.

Leaving most of the bits in an entry as 0 also provides the possibility of updating the entry multiple times via bit-level programs without string-based erases. For instance, as shown in Figure 4(b), writing a mapping update (01001, 01111) into an existing entry (01000, 00100) requires only 0-to-1 programs and induce no erases. On the other hand, when writing updated mapping to an existing entry, bits of the to-be-updated entry are flipped if the flipped entry induces less 0-to-1 programs or no erases, which is the most common condition at runtime of NS-FTL. In Figure 4(b), bits of the updated mapping (10111, 01110) are flipped when written to the existing entry (01000, 10001) and thus only induce one bit-level program. *Updating entries via bit-level programs without string-based erases provides a higher chance for every bit to be programmed before being erased, thus enabling NS-FTL to alleviate the uneven bit-level wearing.*

C. Zig-Zag Space Allocator

As NS-FTL is designed to store the logical-to-physical mapping of NAND flash, the bit pattern of each mapping entry is dependent on the allocation strategy of NAND flash pages. In other words, inappropriate allocation strategy may induce frequent string-based erases and aggravate the issue of uneven bit-level wearing. For instance, if the allocation strategy is unaware of the string-based erases and forces a mapping to be erased during updates with the 1-to-0 update pattern, bits with the value of '0' may get less wearing when compared with bits with the value of '1'. This is because bits with the value of '1' are programmed after erased, while bits with the value of '0' are not programmed after erased. As this variance accumulates during each entry update, the issue of uneven bit-level wearing becomes significant. To alleviate above condition, the zig-zag space allocator is included to maximize the chance of updating a mapping entry without string-based erases, while complying with the allocation restriction of NAND flash. The allocation strategy of the included allocator can be summarized as Figure 5.

As shown in Figure 5(a), the allocation of NAND flash pages are performed in a zig-zag pattern across multiple blocks. The zig-zag allocator allows pages with different block and page numbers to become allocatable at the same time and provides multiple *candidate pages* for a logical page update. Candidate pages with different block and page numbers also have different bit patterns in their logical-to-physical mappings. Such an approach increases the possibility of updating mapping entry via bit-level programs because there are multiple pages of different bit patterns in their logical-to-physical mapping to choose from during logical

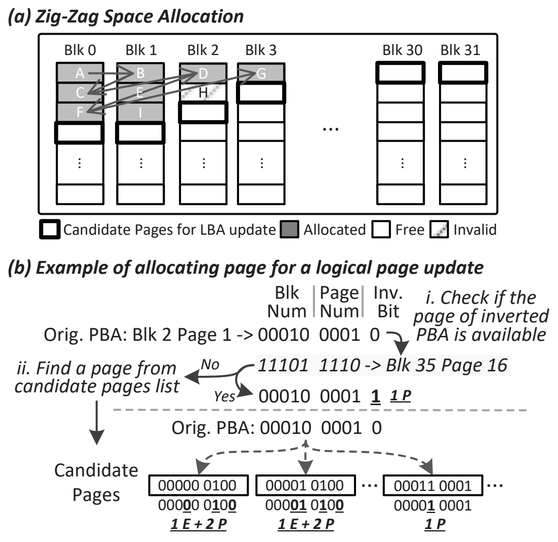
(a) Zig-Zag Space Allocation
Blk 0
Blk 1
Blk 2
Blk 3
Blk 30
Blk 31
A B C D E F G H I
Candidate Pages for LBA update
Allocated
Free
Invalid
(b) Example of allocating page for a logical page update
Blk Num
Page Num
Inv. Bit
i. Check if the page of inverted PBA is available
Orig. PBA: Blk 2 Page 1 -> 00010 0001 0
ii. Find a page from candidate pages list
No
Yes
11101 1110 -> Blk 35 Page 16
00010 0001 1 1 P
Orig. PBA: 00010 0001 0
Candidate Pages
00000 0100
00000 0100
1 E + 2 P
00001 0100
00001 0100
1 E + 2 P
00011 0001
00001 0001
1 P

Fig. 5. **The zig-zag space allocator**, *which allocates NAND-flash pages for logical page updates in a zig-zag pattern to maximize the chance of updating mapping entry via bit-level programs only and with no string-based erases.*

page updates. Notably, because NAND flash only allows pages to be allocated in ascending order, each block has only one available candidate page at a time.

The detailed workflow of the zig-zag allocator can be summarized as Figure 5(b). While updating a mapping entry for a logical page update, the allocator first checks if the physical page with inverted mapping is available or not. If it is available, the updated data content will be stored into the physical page with inverted logical-to-physical mapping. Then, the mapping entry update can be achieved by setting the inverted bit to 1. In contrast, if the physical page with inverted mapping is not available, NS-FTL will go through candidate pages for finding a physical page with the mapping that induces the minimal number of programs or erases. It is worth noting that the zig-zag pattern also makes most of the bits in the mapping entries become zero with smaller block and page numbers. During future updates, these entries have greater chance of being updated by bit-level programs to increase block or page numbers without being erased.

D. Dual Rotation Wear Leveler

In addition to alleviating the uneven bit-level wearing by exploiting the features of NAND-SPIN, the dual rotation wear leveler is designed to resolve the issue of uneven inter-entry wearing by referring to previous hardware-based wear leveler [6]. Uneven inter-entry wearing happens when a few mapping entries are updated multiple times due to intensive logical page updates. As shown in Figure 6, to resolve this issue, NS-FTL divides NAND-SPIN into multiple regions, each of which is composed of a fixed number of NAND-SPIN strings, and designates a *pivot* and an extra string as *interval* in each region. For performing inter-entry wear leveling, the *interval* will be moved forward after a region revives N entry updates. After the *interval* reaches the location of the *pivot*, the *pivot* will be moved backward and the *interval* will be reset to the end of the region again. Above operations are referred to as the in-region rotation and aim to quickly spread the intensive logical page updates across different NAND-SPIN strings without inducing high movement overhead.

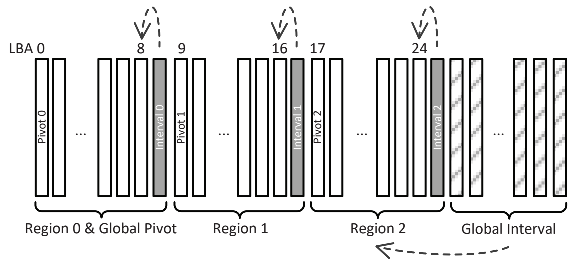

Fig. 6. **The dual rotation wear leveler**, *in which the rotation-based wear leveling is performed at both the in-region and inter-region levels. After a fixed number of entry updates are received in one region or on the NAND-SPIN, the in-region or global interval will be moved forward.*

On the other hand, rotation is also performed at the region level, and it is referred to as inter-region rotation. The goal of inter-region rotation is to avoid intensive mapping updates to a region, owing to strong locality in data update patterns. Similarly, a region is set as the *global pivot*, and one extra region is included as the *global interval*. As the global rotation induces larger overhead when compared with in-region rotation, it is performed less frequently than in-region rotation and is triggered when the NAND-SPIN revived M entry updates, where M is larger than N. The value of N and M can be set according to the lifetime of NAND-SPIN. Based on previous study [7] and assuming NAND-SPIN has the same MTJ size as STT-RAM, the lifetime is predicted to be 4×10^{12}. We then refer to 10^3 and 10^6 as the value of N and M by referring to PCM-based wear leveling study [6].

IV. Performance Evaluation

A. Experiment Setup

In this section, evaluations are conducted by using the Microsoft Research Cambridge (MSR) [5] traces to evaluate the proposed NS-FTL in terms of the bit-level wearing, entry-level wearing, and the throughput of accessing mapping entries on NAND-SPIN. Meanwhile, a self-collected I/O trace is also utilized to assess the effectiveness of NS-FTL with the workload of one-month period personal computer usages. In the experiments, NS-FTL is implemented based on a flash simulator [2] and compared with (1) page-based FTL [1] on NAND-SPIN and (2) page-based FTL with hardware-based wear leveler [6] on NAND-SPIN. These two configuration are referred to as *FTL* and *FTL-WL* in this section. Notably, similar to the dual-rotation wear leveler, the hardware-based wear leveler also includes a pivot and a interval on NVRAM and performs entry-level rotation when the NVRAM receive a fixed number of writes, which is set as 10^3 in this evaluation. The latency parameters of NAND-SPIN can be found in Table I and the size of the NAND flash storage is 64 GB with 16 KB pages [8].

B. Experimental Results

As the main goal of NS-FTL is to mitigate the uneven bit-level wearing of NVRAM-based FTL, Figures 7 and 10 first show the reduced amount of standard deviation and arithmetic mean for bit-level program/erase count. The results show that NS-FTL can effectively reduce the bit-level standard

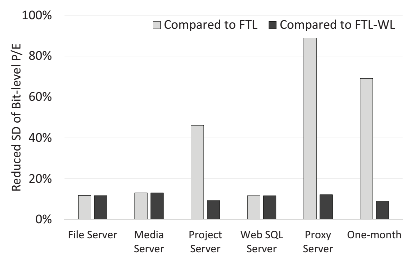

Fig. 7. Standard deviation comparison of bit-level program/erase count.

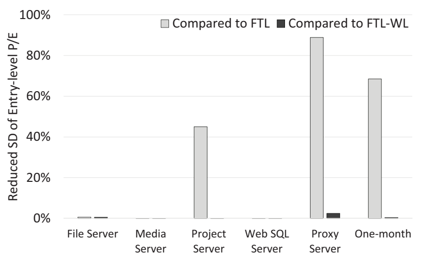

Fig. 8. Standard deviation comparison of entry-level program/erase count.

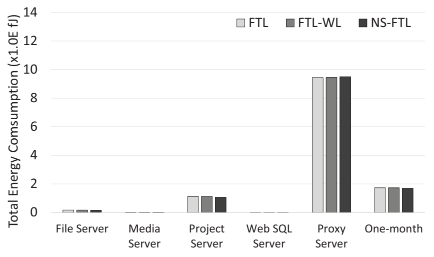

Fig. 9. Total Energy Consumption.

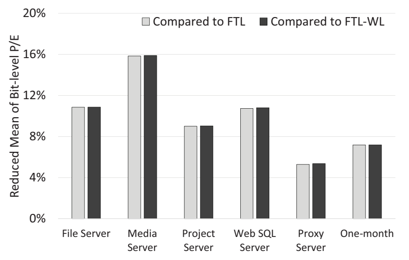

Fig. 10. Arithmetic mean comparison of bit-level program/erase count.

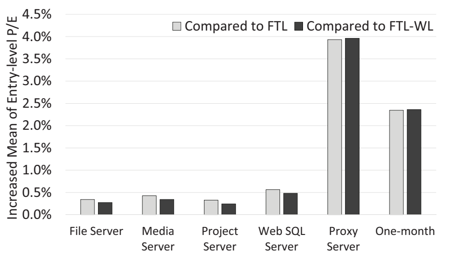

Fig. 11. Arithmetic mean comparison of entry-level program/erase count.

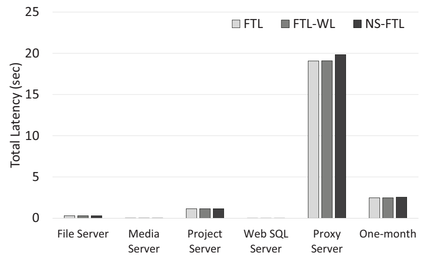

Fig. 12. Total latency.

deviation and arithmetic mean by up to 88.86% and 15.84%, when compared with page-based FTL. The average reductions are 40.11% and 9.86% for bit-level standard deviation and arithmetic mean. These reductions are achieved by both the inverted mapping strategy and zig-zag space allocator. In addition, comparing with FTL-WL, the reductions of bit-level standard deviation and mean are 9.86% and 11.14%, which suggests that NS-FTL can outperform FTL-WL in terms of bit-level wear leveling.

For entry-level comparison, NS-FTL can reduce the standard deviation by 33.88% on average and up to 2.45%, comparing with FTL and FTL-WL, as shown in Figure 8. The reduction of entry-level standard deviation suggests that NS-FTL can effectively utilize NAND-SPIN to alleviate uneven bit-level wearing, while avoiding excessive string-based erases for preventing uneven entry-level wearing. Meanwhile, to further avoid excessive string erases, the arithmetic mean of entry-level program/erase count is slightly increased by 1.30%, as shown in Figure 11, due to the inter-region rotation of the dual rotation wear leveler.

The comparisons of NAND-SPIN energy consumption and latency are reported. Figures 9 suggests that the energy consumption can actually be decreased by 5.16% on average, due to the reduced number of the 0-to-1 update pattern during mapping entry updates. On the other hand, Figures 12 shows that the latency of updating and accessing mapping entry is similar to the page-based FTL, and the difference is only 1.58%, even with the mechanisms of NF-FTL.

V. Conclusion

To resolve the uneven bit-level wearing of NVRAM-based FTL, this study firstly presents the NAND-SPIN FTL (NS-FTL) to utilize the unique bit-level program and string-based erase features of NAND-SPIN. NS-FTL firstly introduces the inverted mapping strategy to minimize the number of '1' bits during mapping entry updates for avoiding excessive string-based erases. Then, the zig-zag space allocator is included to alter the allocation of NAND flash pages for further the probability of the 0-to-1 update pattern. Finally, to avoid uneven inter-entry wearing, the dual rotation wear leveler is utilized to periodically rotate the physical location of each mapping entry. Experimental results show that the standard deviation of bit-level program/erase count can be reduced up to 88.86%, when compared with the page-based FTL.

References

[1] A. Ban. Flash file system, U.S. 5404485 A, Apr. 1995.

[2] Y.-H. Chang and T.-W. Kuo. A management strategy for the reliability and performance improvement of mlc-based flash-memory storage systems. *IEEE Transactions on Computers*, Mar 2011.

[3] J. K. Kim, H. G. Lee, S. Choi, and K. I. Bahng. A pram and nand flash hybrid architecture for high-performance embedded storage subsystems. In *Proceedings of the 8th ACM International Conference on Embedded Software*, 2008.

[4] D. Liu, T. Wang, Y. Wang, Z. Qin, and Z. Shao. Pcm-ftl: A write-activity-aware nand flash memory management scheme for pcm-based embedded systems. In *IEEE 32nd Real-Time Systems Symposium*, 2011.

[5] D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: Practical power management for enterprise storage. In *ACM Transactions on Storage (TOS)*, 2008.

[6] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009.

[7] K. Rani and H. K. Kapoor. Write variation aware buffer assignment for improved lifetime of non-volatile buffers in on-chip interconnects. *IEEE Transactions on Very Large Scale Integration Systems*, 2019.

[8] Samsung. Samsung v-nand@ONLINE, http://www.samsung.com/semiconductor/products/flash-storage/v-nand/, 2017.

[9] Y. SONG, S. LEE, D. H. KANG, and Y. I. EOM. Nvram-aware mapping table management for flash storage devices. *IEICE Transactions on Information and Systems*, 2019.

[10] Z. Wang, L. Zhang, M. Wang, Z. Wang, D. Zhu, Y. Zhang, and W. Zhao. High-density nand-like spin transfer torque memory with spin orbit torque erase operation. *IEEE Electron Device Letters*, 2018.

[11] B. Wu, P. Dai, Z. Wang, C. Wang, Y. Wang, J. Yang, Y. Cheng, D. Liu, Y. Zhang, W. Zhao, and X. S. Hu. Bulkyflip: A nand-spin-based last-level cache with bandwidth-oriented write management policy. *IEEE Transactions on Circuits and Systems*, 2020.

[12] H. Zhang, W. Kang, Z. Wang, E. Deng, Y. Zhang, and W. Zhao. High-density and fast-configuration non-volatile look-up table based on nand-like spintronic memory. In *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2018.