OSwrite: Improving the lifetime of MLC STT-RAM with One-Step write

Wei Zhao[1], Wei Tong[1*], Dan Feng[1], Jingning Liu[1], Jie Xu[1], Xueliang Wei[1], Bing Wu[1], Chengning Wang[1], Weilin Zhu[1], Bo Liu[2]

[1]Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System
School of Computer Science and Technology, Huazhong University of Science and Technology
Ministry of Education of China, Wuhan, China

[2]Hikstor Technology Co., LTD, Hangzhou, China

Email:{weiz, tongwei, dfeng, jnliu, xujie_dsal, xueliang_wei, wubin200, chengningwang, weilinzhu}@hust.edu.cn, liubo@hikstor.com

***Abstract*—Spin-Transfer Torque Random Access Memory (STT-RAM) is a promising cache memory candidate due to high density, low leakage power, and non-volatility. Multi-Level Cell (MLC) STT-RAM can further increase density by storing two bits in the hard and soft domain of a cell respectively. However, MLC STT-RAM suffers from severe lifetime issues because of its two-step write operation. As two-step write could incur extra writes to a cell's soft domain, which drastically degrades the overall lifetime of MLC STT-RAM. Thus, it is necessary to reduce the wear to soft domain so that extend the lifetime of MLC STT-RAM.**

We observe that the most wears to the soft domain are produced by the hard domain bit flips (i.e. Two-step Transition and Hard Transition). Based on the observation, we propose One-Step write (OSwrite) to avoid Two-step Transition (TT) and Hard Transition (HT). Half-Sized Compression (HSC) removes HTs and TTs by writing data only to the soft domain through compression techniques. The compressed data is encoded to further reduce the writes to soft domain. Besides, Hard Transition Removal Encoding (HTRE) scheme is used while data cannot be compressed to less than half-size. HTRE scheme uses a hard flag to record the state of hard domain flipping to avoid changing its value. Then, HTRE compresses the hard flag and encodes the soft data to further reduce the writes to soft domain with encoding tags stored in the saved space of hard flag. Evaluation results show that OSwrite can improve the lifetime of MLC STT-RAM to 2.6×. Our scheme can largely decrease HTs and TTs thus achieve one-step write. The results show OSwrite reduces write energy and improves system performance of MLC STT-RAM by 56.2% and 6.4% respectively. Besides, OSwrite reduces hard bit flips and soft bit flips by 82.8% and 5.3% respectively.

I. INTRODUCTION

The development of big data and in-memory computing has raised the requirement of large capacity of cache memory. For instance, IBM Power 8 equips up to 128MB Last Level cache to alleviate the main memory data communication in these applications [1]. However, traditional SRAM based cache faces low density, high refresh power, and scalability challenges. Emerging non-volatile memory provides feasible solutions to these problems [2], [3]. Spin-Transfer Torque Random Access Memory (STT-RAM) has attracted much attention due to its several advantages such as high density, low leakage power, good compatibility with CMOS, and low cost [4] [5] [6]. STT-RAM can be the cache of many high-end embedded processors due to these excellent characteristics.

Single-Level Cell (SLC) STT-RAM can only store one logic value through the resistance state of Magnetic Tunneling Junction (MTJ) device [4]. Recent research progress in MTJ devices has developed Multi-Level Cell (MLC) STT-RAM to further enhance cell density by stacking two MTJs in a cell [7] [8]. Unfortunately, lifetime is a common problem for STT-RAM. The 10^{15} programming cycles reported by previous works are hard to reach [9] [10], and the best endurance test result for SLC STT-RAM devices so far is less than 4×10^{12} cycles. Recently, in the product filed, the MRAM company Everspin issued their 28nm node 1Gb SLC STT-RAM chip [11], which can program only 10^{10} cycles. While beyond the limited cycles, the bit error rate may start to increase, and system level ECC is then recommended. As for MLC STT-RAM, writing two different MTJs in a cell causes asymmetric write current. The larger write current exponentially degrades the lifetime of memory cells as dielectric breakdown [12]. MLC STT-RAM suffers from severer lifetime issues compared with SLC STT-RAM. Besides, the lifetime problem is further severer due to two-step write [12], which leads to extra writes to the soft domain. Therefore, the predicted programming cycles for MLC STT-RAM are maybe less than 10^{10}. This write endurance is not high enough to satisfy the memory-intensive applications, and some techniques must be used to improve the lifetime.

Many wear-leveling techniques [10] [13] [14] have been proposed to improve the lifetime of non-volatile cache memory. However, their works mainly focus on the intra-set write variations and don't consider the unbalanced write of MLC STT-RAM. Except for the wear-leveling techniques, some works put the two-step write operation of MLC STT-RAM into consideration. They proposed some encoding techniques [9] [15] [16] to reduce the Two-step Transition (TT) thus improve the lifetime. However, these existing encoding techniques mainly focus on reducing TTs, and lifetime improvement

*Corresponding author: Wei Tong (tongwei@hust.edu.cn) is limited. Our scheme can reduce both the TTs and STs. Besides, these techniques produce large additional capacity overhead, and they lead to performance loss due to extra latency to encode and decode data.

In this paper, we make a detailed analysis of the data transitions of MLC STT-RAM, and we observe that Two-step Transition (TT) and Hard Transition (HT) produce the main wear to the soft domain, which could largely damage the lifetime of MLC STT-RAM. Besides, Soft Transition (ST) also makes an impressive contribution to lifetime reduction. Based on this key observation, we propose **One-Step write (OSwrite)** to improve the lifetime of MLC STT-RAM with performance improvement and effective energy reduction. OSwrite includes two techniques, Half-Sized Compression (HSC) and Hard Transition Removal Encoding (HTRE). **Half-Sized Compression (HSC)** avoids HT and TT by writing data only to the soft domain. We observe that many cache lines of L3 cache can be compressed to half or less, these data can be written to the soft domain with one-step write operation. Besides, the size of compressed data varies greatly, so we select a suitable encoding scheme to reduce the soft bit flips of the compressed data with encoding tags placed in the saved space. The encoded data are then written to the soft domain of MLC STT-RAM to achieve one-step write. Next, as for cache lines that cannot be compressed to less than half-size, we propose **Hard Transition Removal Encoding (HTRE)** scheme to remove HT and TT. We use a hard flag to record the state of hard domain flipping. If the old and new hard data are the same, the corresponding hard flag bit is set to '0', while flag data bit is set to '1' if the hard data updates. Then the hard flag can be easily compressed by Frequent Pattern Compression (FPC) due to many existing 0s, and the soft domain can be encoded with encoding tags stored in the saved space of hard flag to reduce soft transitions. HTRE can finish write operation in one-step as well. Besides, we design a simple flag allocation mechanism to ensure low capacity overhead of hard flag. Each cache line equips index data to find the corresponding hard flag. The contributions of this paper are as follows:

- We analyze the write data pattern of MLC STT-RAM, and we observe that Two-step Transition and Hard Transition perform the main contributions to lifetime reduction. Besides, Soft Transition also makes an impressive degradation in lifetime.
- Based on our observation, we propose OSwrite to improve the lifetime with performance improvement and efficient energy reduction.
- Experimental results show that OSwrite can improve the lifetime of MLC STT-RAM to 2.6$\times$, reduce write energy, and improve system performance by 56.2% and 6.4% respectively. Besides, OSwrite reduces hard bit flips and soft bit flips by 82.8% and 5.3% respectively.

The rest of this paper is organized as follows. Section II introduces some basics of STT-RAM and analyzes the data patterns of MLC STT-RAM. Moreover, we analyze the lifetime issue and explain the motivation. Section III describes the proposed OSwrite strategy and its implementation. The evaluation results and comparison of OSwrite with the existing schemes are given in Section IV. Section V shows the related works. Finally, we conclude our paper in Section VI.

II. Background and motivation

A. The Basics of STT-RAM

STT-RAM cell is composed of one transistor and one Magnetic Tunneling Junction (MTJ), i.e. 1T1M structure. MTJ is the main component of STT-RAM [4], and different resistance values of MTJ correspond to different logic values (0 or 1). The transistor can control the current that flows through the MTJ to write and read data bit. Fig. 1(a) shows the structure of the SLC STT-RAM cell, and MTJ is composed of two ferromagnetic layers (Free Layer and Reference Layer) and one oxide barrier layer (MgO). The magnetization direction of the reference layer is fixed, and the free layer's magnetization direction can be parallel (P) or anti-parallel (AP) to the reference layer. P and AP indicate the cell is in low resistance state (logical 0) and high resistance state (logical 1), respectively. STT-RAM is based on MTJ, and the tunneling magnetoresistance (TMR) ratio of MTJ is typically small ($<$200% or $<$2X) [4]. The TMR is defined as:

$$TMR = R_{ap}/R_p - 1 \tag{1}$$

R_{ap} and R_p are the resistance of the AP and P state, respectively. If the MTJ stores two bits, it's difficult for the write and read circuit to sense the difference of four states due to the small TMR. Thereby STT-RAM cannot store two or more bits in an MTJ. Serial MLC (Fig. 1(b)) stores two bits by connecting two different-size MTJs in series. Each MTJ indicates a logical value. Parallel MLC (Fig. 1(c)) utilizes one single MgO layer with two different-size free layers. The write and read operation of two kinds of MLC STT-RAM are the same. In this paper, we assume serial MLC is used.

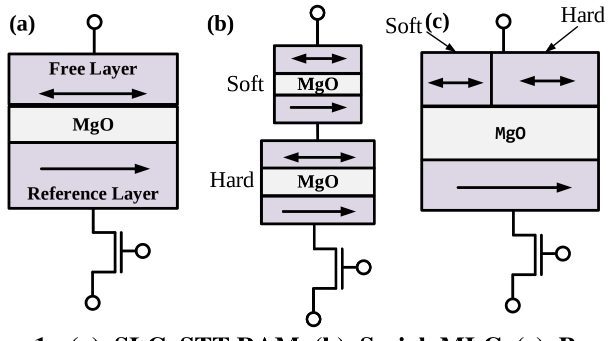

Fig. 1: (a) SLC STT-RAM (b) Serial MLC (c) Parallel MLC.

Each MLC STT-RAM consists of a hard domain and a soft domain, and the hard domain needs a larger write current to switch the logical value compared with the soft domain. MLC stores Most Significant Bit (MSB) in the hard domain, while storing the Least Significant Bit (LSB) in the soft domain. Therefore, writing hard (MSB) changes the value of soft (LSB) due to the larger write current. To write data correctly, two-step write [12] is proposed for MLC STT-RAM. This technique writes hard with a large current I_{hard}, thus both the soft and hard are switched to the same logical value. Then restoring soft with a small current I_{soft}. Fig. 2(a) shows the detailed write data pattern transformation. For example, in the data transition 00→10, I_{hard} is used to update data 00 to 11, and then I_{soft} is applied to write 11 to 10. Similarly, reading the data from an MLC needs two steps as well, which is accomplished by injecting a low current to the MTJ and estimating the voltage using a sense amplifier [17]. The read scheme is based on binary search algorithm as shown in Fig. 2(b). In the first step, a read current is injected to the MTJ and then derived V_R is compared with V_{ref1} to determine the logic value of the soft domain. In the second step, a larger read current is injected and the voltage is compared with V_{ref2} or V_{ref3} to determine the value of hard domain.

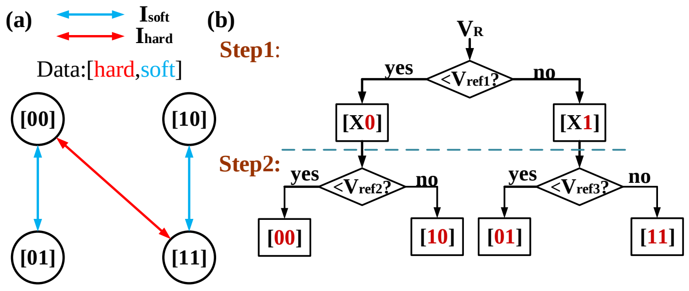

Fig. 2: (a) Write operation of MLC (b) Read operation.

B. Analysis of the write data pattern

In general, write operations of MLC STT-RAM can be summarized as four types:

- **Zero Transition (ZT)**: The new data is the same as old data. e.g. 00 → 00.
- **Soft Transition (ST)**: Only the LSB of new and old data are different. e.g. 00 → 01 or 10 → 11.
- **Hard Transition (HT)**: The MSB of new and old data are different, and the MSB and LSB of new data are the same. e.g. 00 or 01 → 11.
- **Two-step Transition (TT)**: In this case, writing a data needs both the Hard Transition and Soft Transition. i.e. **TT = HT + ST**. e.g. 00 or 01 → 10.

All kinds of data transition are shown in TABLE I. For each data transition, there are two values in the brackets. The two values represent the wear number to the hard and soft domain, respectively. In the process of an HT, a large current flows through both the soft and hard domain, resulting in one wear to both the hard and soft domain. As for TT, soft suffers from two wears due to one extra ST to restore data. Thus, HT and TT result in a large number of wears to the soft domains which largely degrade the lifetime. Besides, in all 16 data pattern transitions, half of these transitions are TT and HT, thus leading to large lifetime damage to the soft domain.

C. Data Mapping for MLC based cache

Since the two bits of MLC STT-RAM are asymmetric, thus accessing the cache line is different from the common cache.

TABLE I: State transitions of MLC STT-RAM [5] [9]

From \ To	00	01	10	11
00	ZT(0, 0)	ST(0, 1)	TT(1, 2)	HT(1, 1)
01	ST(0, 1)	ZT(0, 0)	TT(1, 2)	HT(1, 1)
10	HT(1, 1)	TT(1, 2)	ZT(0, 0)	ST(0, 1)
11	HT(1, 1)	TT(1, 2)	ST(0, 1)	ZT(0, 0)

1. The latency and energy of a TT equal to the sum of an HT and an ST.
2. The write energy of HT and ST are $1.659nJ$ and $0.843nJ$, respectively.

This section describes three typical block-level data mapping methods for MLC STT-RAM cache. For example, given a 512-bit (64-byte) data block and 256 2-bit MLCs, we can put i-th data bit into $i/2$-th MLC, this block organization is *Direct Mapping* (DM). Fig. 3(a) shows this data mapping method. DM is a straightforward method without differentiating the accesses to different parts of the block. Therefore, the latency is always the worst-case since both the soft and hard bits need to be sensed regardless of which data word is accessed [18].

To take advantage of the fact that soft region is fast, Bi *et al.* proposed *Cell Split Mapping* (CSM) [8], which stores one 512-bit cache line data in the soft region of 512 MLCs, while another block data is stored in the hard region of MLCs. Fig. 3(b) shows the cell split mapping cache block. This method can produce a fast way and a slow way. For the hard region cache lines, if the read and write request is frequent, the system performance degrades since the prolonged access latency. Besides, the frequent write causes huge writes to the soft region cache line data thus degrade lifetime and increase write energy.

The third way is to put the lower half (bit 0-255) of block data bits in the soft region of the 256 MLCs, while the rest half data is stored in the hard region. Fig. 3(c) shows this mapping method, which is called *Interleaved Mapping* (IM). IM can write the lower half with only one step and maintain the word level data property in the soft and hard region. Besides, there is no inter-cache line disturb in IM compared with CSM.

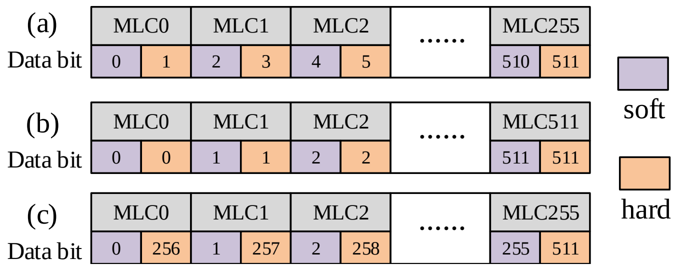

Fig. 3: (a) Direct Mapping (b) Cell Split Mapping (c) Interleaved Mapping.

D. Motivation

We make a detailed discussion about the issues which Hard Transition and Two-step Transition lead to as follows:

1)Limited lifetime: The lifetime of MLC STT-RAM is mainly determined by the soft domain since it suffers from much more wears compared with the hard domain. Besides, the large number of HTs and TTs cause huge wears to the soft domains, which can drastically reduce the lifetime.

2)Write energy: As shown in the bottom of TABLE I, TT and HT consume more energy than ST. These writes are energy-inefficient, which should be removed as more as possible.

3)Write latency: MLC STT-RAM can improve density, thus reduce the cache miss rate. However, the access latency is prolonged because of two-step read and write, which could degrade the system performance.

We evaluate the write data pattern of some memory-intensive benchmarks from SPEC CPU2006 [19]. The system configuration is listed in TABLE V. We only consider ST, HT, and TT. ZT is not considered because ZT doesn't contribute to the wear of the cell and write energy. The result is shown in Fig. 4. We test the proportion of three write transitions, and HT and TT take up most of the write transitions. For some benchmarks like *mcf, sjeng* have a large proportion of HT and TT. Especially in *sjeng*, HT and TT account for 83.58% of all data transitions, which causes too much write energy and degradation of lifetime. Besides, the gray column in Fig. 4 illustrates the contribution of HT and TT to the wear of the soft domain. The contribution is calculated by the following formula.

$$C_{HT,TT} = \frac{wear_{TT,HT}}{wear_{TT,HT,ST}} \tag{2}$$

$C_{HT,TT}$ is the contribution of lifetime degradation that HT and TT make. $wear_{TT,HT}$ represents the wear number caused by TT and HT. $wear_{TT,HT,ST}$ is the overall wear number to the soft domain. The result indicates that nearly at least 70% lifetime degradation is caused by TT and HT. Most previous works only consider reducing the number of HT and TT. However, we find ST also leads to an impressive reduction in lifetime. This observation motivates us to mainly reduce HT and TT, and decrease ST at the same time to improve the lifetime of MLC STT-RAM.

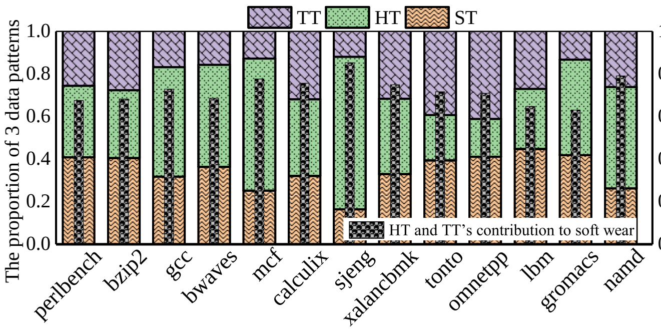

Fig. 4: Write data pattern of MLC STT-RAM.

III. Proposed Design

Based on our motivation, we propose One-Step write (OSwrite) to improve the lifetime and energy efficiency of MLC STT-RAM. OSwrite includes Half-Sized Compression and Hard Transition Removal Encoding scheme these two main techniques.

A. Half-Sized Compression

1) Encoding procedure: From the analysis above, Hard Transition (HT) and Two-step Transition (TT) take a large portion of data transitions, which leads to large lifetime degradation and energy consumption. Compression is an effective way to reduce the bit writes so that improves lifetime and energy efficiency. Previous work [20] [21] shows many cache lines are compressible. We use Frequent Pattern Compression (FPC) [22] to compress last level cache (LLC) lines because of its low implementation and performance overhead. FPC algorithm compresses 8 frequent 64-bit data patterns in this work. The frequent patterns are listed in TABLE II. Fig. 5 shows the size distribution of the last level cache line after compression. We observe that many cache lines can be compressed. To make full use of the advantages of soft region, we adopt *Interleaved Mapping* in our scheme to store as more data bits as possible in the soft region. Therefore, we propose Half-Sized Compression (HSC) to write data only in the soft region thus removing HT and TT. If the compressed cache lines are less than 256 bits, these data can be fully written into the soft domain (❶ in Fig. 6). HSC reduces write energy, access latency, and further improves the lifetime of MLC STT-RAM with one step write. Compression produces extra 24 prefix bits to record the data pattern of each 64-bit word. In order to ensure one-step operation, these bits are stored in the soft domain. Thus, the rest space for compressed data must be less than 232 bits. Fig. 5 shows the ratio of compressed size less than 232 bits. If the cache line cannot be compressed, there is no space to store the compression tag in the cache line data region, we need an extra MLC to store this tag to indicate this cache line is compressible or not.

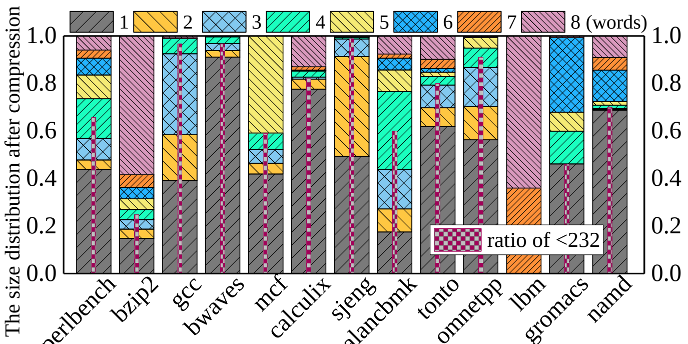

Fig. 5: The size distribution of last level cache lines after compression.

We observed that after compression, the saved space of the soft domain varies greatly. Moreover, writing soft data directly may create many soft transitions, which can reduce the lifetime of MLC STT-RAM. We use encoding techniques to reduce soft bit flips of compressed data thus further improve lifetime and reduce write energy [21]. We adopt Flip-N-Write (FNW) [24] encoding technique due to its low overhead and complexity. For n data bits, if $i(i>n/2)$ bits need to be updated, flipping the data bits can make only $n-i(n-i<n/2)$ data bits to be written. FNW is a simple encoding technique to reduce bit flips. Fine-grained FNW can further reduce bit

TABLE II: The compressible data patterns of 64-bit FPC algorithm [22], [23]. The prefix is indicated in red.

Prefix	Pattern encoded	Example	Compressed example	Encoded size
000	Zero run	0x0000000000000000	0x0	3 bits
001	8-bits sign-extended	0x000000000000007F	0x17F	11 bits
010	16-bits sign-extended	0xFFFFFFFFFFFFB6B6	0x2B6B6	19 bits
011	Half-word sign-extended	0x0000000076543210	0x376543210	35 bits
100	Half-word, padded with a zero half-word	0x7654321000000000	0x476543210	35 bits
101	Two half-words, each two bytes sign-extended	0xFFFFBEEF00003CAB	0x5BEEF3CAB	35 bits
110	Consisting of four repeated double bytes	0xCAFECAFECAFECAFE	0x6CAFE	19 bits

flips [24] [25], so we can use different encoding granularity to adapt to corresponding saved space(❷ in Fig. 6). Optional encoding granularity of HSC is 2, 4, 8, 16. Furthermore, the size and encoding granularity of the compressed data satisfy the following relationship.

$$G = \begin{cases} 2, \ S_{comp} \in [0, 154] \\ 4, \ S_{comp} \in [155, 185] \\ 8, \ S_{comp} \in [186, 206] \\ 16, \ S_{comp} \in [207, 218] \\ no \ encoding, \ S_{comp} \in [219, 256] \end{cases} \quad (3)$$

S_{comp} is the size of compressed data, G is the encoding granularity. The unique G for a compressed cache line can be determined by the formula above.

The overall encoding process of ***Half-Sized Compression (HSC)*** is diagrammed in Fig. 6. 1-bit tag is used to indicate whether the cache line can be compressed or not, and prefix records the data patterns of 8 64-bit words. The compression and encoding process brings extra overhead, and the detailed analysis is discussed in section III.D.

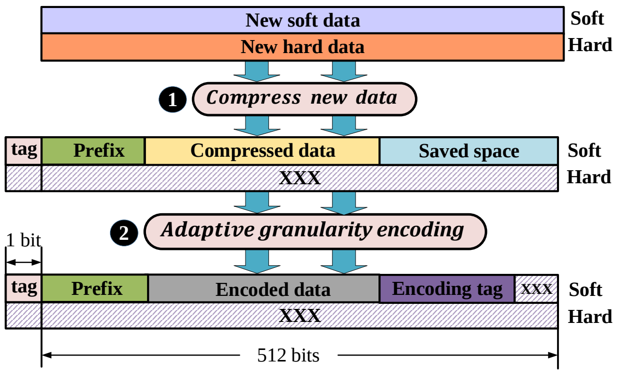

Fig. 6: The encoding process of HSC.

2) Decoding procedure: When the encoded cache line receives a read request, the HSC decoder works as Fig. 7 depicts. The tag bit and prefix data are first read out. Thus, the decoder can get the size of compressed data according to that data bits. The decoder obtains the encoding granularity of FNW through the compressed data size. Then the encoded data is decoded by the FNW decoder module. After that, the data can be decompressed word by word according to the prefix of each word.

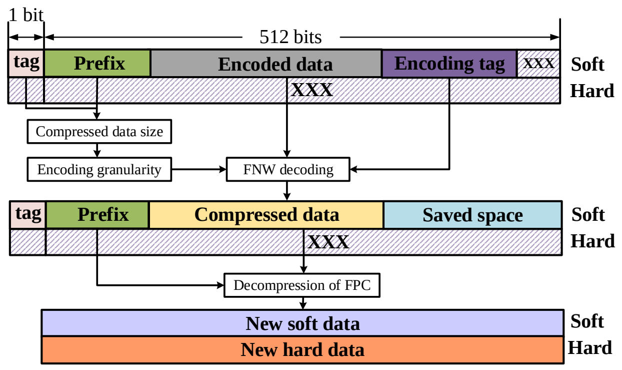

Fig. 7: The decoding process of HSC.

B. Hard Transition Removal Encoding scheme

We propose Hard Transition Removal Encoding (HTRE) scheme to write MLC with only one step while data cannot be compressed to half-size.

1) Removing Hard Transition: We use Data-Comparison Write (DCW) [26] to encode the hard domain (❶ in Fig. 8). Hard flag stores '1' if the new and old value of the hard domain is different, while the hard flag is set to '0' when the data is the same. By applying this simple XOR method, HT and TT can be removed. We use SLC STT-RAM to store hard flag data because SLC STT-RAM has a longer lifetime and only one-step write compared with MLC STT-RAM [5] [8] [18]. In Fig. 8, we use 8 MLCs to briefly explain the HTRE encoding process. The original write operation consumes 5 soft writes and 3 hard writes, which lead to 8 wears to the soft domain. After step ❶, only 5 soft writes occurs on soft region. This operation can reduce the hard transitions thus improve lifetime.

2) Flags compression: The 1st step of *HTRE* can remove HT and TT. But there are still many STs in the soft domain, which can significantly reduce the lifetime of MLC STT-RAM. In previous work [27], about half of clean words exist in cache lines, and clean words are not modified in the cache line update process. Thus, the hard domain exists clean words, which lead to many 0s data existing in the hard flag after the 1st step encoding. The constant zero data in the hard flag is easy to be compressed by FPC(❷), and when the hard flag is compressed, the compression tag bit is set to '1'. We make experiments and find that more than 86% hard flags are compressible. Moreover, we observe that the saved space of the hard flag varies greatly. Thus, we can use the encoding technique (e.g. FNW) to reduce bit flips of soft domain with encoding tags stored in the saved space of hard flag. The encoding granularity varies from 2 to 16, and the optimal granularity is varied according to the saved space. Besides, We can encode soft data together with the compressed hard flag if the saved space is large enough. The encoding granularity is a function of compression hard flag size, and the function is listed as follows.

$$G = \begin{cases} 2, \ S_{comp} \in [1, 76], \ encode \ soft \ and \ flag \\ 2, \ S_{comp} \in [77, 116], \ only \ encode \ soft \\ 4, \ S_{comp} \in [117, 180], \ only \ encode \ soft \\ 8, \ S_{comp} \in [181, 212], \ only \ encode \ soft \\ 16, \ S_{comp} \in [213, 228], \ only \ encode \ soft \\ no \ encoding, \ S_{comp} \in [229, 256] \end{cases} \tag{4}$$

G indicates the encoding granularity, and S_{comp} is the size of compressed hard flag. After the FNW encoding process, hard flag and soft data are written at the same time to achieve one-step write. The example in Fig. 8 shows that the writes to soft domain can be reduced to 3 with flipping the soft data in step ❸. num_{hard_flag} indicates the writes to the hard flag, and the total writes are less than 8. HTRE can improve the lifetime of MLC to 266.7% (8/3) and reduce the total write energy.

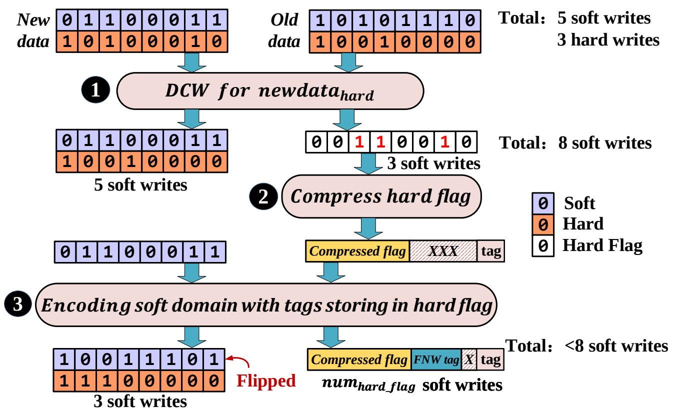

Fig. 8: HTRE encoding process.

3) Hard flag array organization: It's not necessary to equip a hard flag for each cache line, we only consider cache lines that cannot be encoded by Half-Sized-Compression. We use search logic to dynamically allocate a hard flag for this cache line. Moreover, we make a series of experiments to determine suitable flag size due to the trade-off between performance and capacity overhead. We set up a 1MB hard flag array for 16MB MLC STT-RAM Last Level Cache with good performance and low capacity overhead. Fig. 9 shows the organization of the flag array. 1MB array is divided into 128 blocks, and each block has 256 rows, each row has 256 bits to store hard flag data. We assign each row 1 bit to indicate whether it is valid or not, and each block also has a bit to indicate its state. When *Cacheline_htre* will be encoded by HTRE. The **search logic1** finds valid block *block_m*. Then, **search logic2** finds valid row *row_n*. *block_m*(7-bit) and *row_n*(8-bit) are written into index array (MLC), and the state of *row_n* is set 'Invalid'. If all rows of a block are 'Invalid', the state of the block is 'Invalid'. When *Cacheline_htre* is evicted, the hard flag of *Cacheline_htre* turns from 'Invalid' to 'Valid', thus this flag can be used for next write request. When CPU reads *Cacheline_htre*, the decode logic finds corresponding hard flag position via index data thus decodes the data. The search logic leads to extra overhead, detailed information is in the following part.

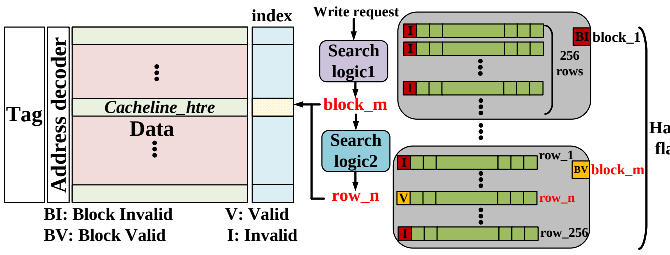

Fig. 9: The process of searching valid hard flag.

4) Decoding procedure: When the encoded cache line receives a read request, the HTRE decoder works as Fig. 10 depicts. The index data is first read out to locate the hard flag. Then, the decoder reads the compression tag bit to ensure if this hard flag is compressed. The decoder can get the size of compressed hard flag according to the prefix, thus the encoding granularity of FNW can be obtained through data size. Next, the soft domain data is decoded by the FNW decoder module. After that, the original hard flag data is decompressed word by word according to the prefix of each word. In the end, the hard domain data can be decoded through the XOR operation between the hard flag data and old hard data.

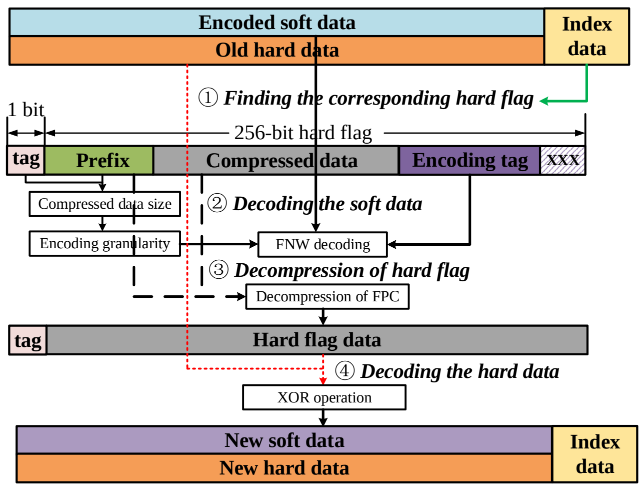

Fig. 10: The decoding process of HTRE.

C. Overall Architecture of OSwrite

Fig. 11 shows the overall encoding procedure of OSwrite. When LLC receives a write request, the compressor determines if new data can be compressed to less than half-size. If the data can be encoded by ***HSC***, then this write is a one-step operation and all the data can be written only in soft domain. Secondly, if the search logic finds the remaining hard flag, this cache line is encoded by ***HTRE***. Finally, when there is no valid hard flag. These cache lines can be encoded by course-grained ES-FNW with encoding tag storing in the index array. ES-FNW is a technique that encoding the hard data and soft data by Flip-N-Write separately [16]. In the overall architecture, each cache line equips 16 MLCs. The index data of HTRE needs 15 bits, and this data is written at the soft domain of 15 MLCs to ensure one-step write. The rest one MLC is used to record which scheme that the cache line is encoded by. To ensure writing this encoding type flag with one-step and pretty low energy, '00', '01', '11' are used to represent HSC, HTRE, ES-FNW respectively. Besides, when a cache line encoded by ES-FNW is evicted, its encoding type flag is reset to '00' to reduce the case of 11→01, which is a two-step write. TABLE III shows the encoding type flag. The overall encoding procedure uses pipeline optimization to reduce performance loss.

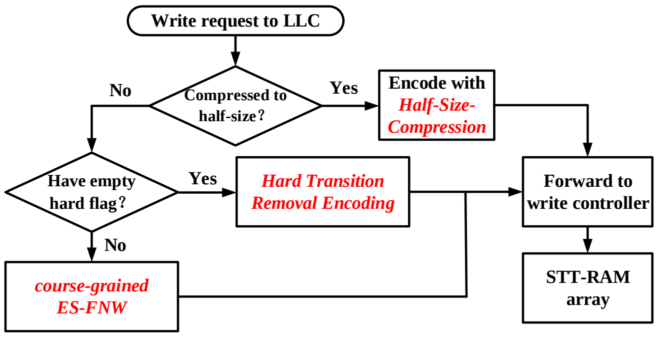

Fig. 11: The encoding data flow of OSwrite.

TABLE III: Encoding type flag for different schemes

Encoding scheme	Encoding type flag
HSC	00
HTRE	01
ES-FNW	11

Fig. 12 shows the decoding procedure of OSwrite. When LLC receives a read request, the decoder reads the encoding type flag to execute the corresponding decoding procedure. The flag value '00', '01', '11' correspond to HSC decoding, HTRE decoding, ES-FNW decoding, respectively. After this process, the correct data can be readout.

D. Overhead Analysis

1) Capacity: OSwrite needs a 1MB SLC STT-RAM flag for 16 MB LLC, and each 256-bit hard flag equips 2 bits to indicate if it is compressible and valid. Besides, OSwrite equips 16 MLCs for each cache line to store the position of hard flag and indicate the encoding type. Thus, capacity overhead is roughly 12.55%.

2) Hardware: OSwrite needs compression and encoding hardware to perform the one-step write. The main hardware logic overhead is shown in TABLE IV. The encoding and decoding overhead of FPC and FNW are cited from [21] [23]. We use the Synopsys Design Compiler to synthesis the search logic and multiplexer in 130*nm* technology, and we use scaling rules of transistors to scale the results down to 22*nm* process node. Thus the latency of search logic is 0.26*ns*, and the energy consumption is 1.9*pJ*. The latency and energy overhead of multiplexer is 1.36*ns* and 1.8*pJ* respectively. The encoding and decoding parameters of ES-FNW are derived from [16].

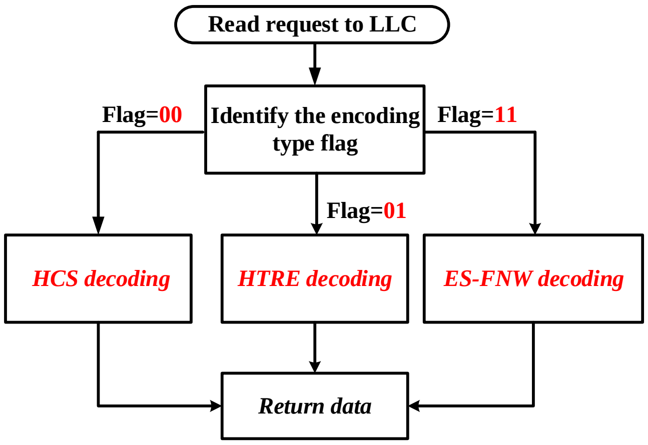

Fig. 12: The decoding data flow of OSwrite.

TABLE IV: Latency and energy overhead of OSwrite

Implementation	Latency	Energy
FPC	Encoding: 2*ns* Decoding: 1*ns*	Encoding: 2.1*pJ* Decoding: 1.2*pJ*
Flip-N-Write	Encoding: 1*ns* Decoding: 0.1*ns*	negligible
Search logic	0.26*ns*	1.9*pJ*
Multiplexer	1.49*ns*	1.68*pJ*

IV. Experimental Setup

We use full system simulator Gem5 [28] to evaluate several schemes. TABLE V lists the system configuration, and we select 14 benchmarks from SPEC CPU 2006 [19] to evaluate the results of our scheme. We run 100 million instructions to warm up the cache and then run 500 million instructions for the evaluation of our design. We use MLC STT-RAM to replace traditional SRAM based Last Level Cache, and the configuration of STT-RAM is listed in TABLE VI.

TABLE V: System configurations

Cores	4-Core, 2.0GHz, out-of-order
L1 I/D cache	private, 32KB per core, 2-way; LRU, 2-cycle latency
L2 Cache	private, 512KB per core, 64B cache line; 8-way, LRU, 10-cycle latency

L3 Cache	MLC STT-RAM based cache, 16MB;
	SLC STT-RAM based cache, 8MB ;
	shared, 64B cache line, 16-way, LRU;

Main Memory	4GB, DDR-1600

We make comparisons with the following schemes:

- SLC: We use 8MB SLC STT-RAM to be the last level cache.
- DCW [26]: Writing data by Data Comparison Write.

TABLE VI: The parameters of STT-RAM [5]

	SLC	MLC
Read Latency (*Cycles*)	5.5	S:4.08 H:5.94
Write Latency (*Cycles*)	15.5	S: 15.34 H:34.24
Read Energy (*nJ*)	0.216	S:0.22 H:0.43
Write Energy (*nJ*)	0.839	S:0.843 H:2.502

- TSTM [9]: Every 3 MLCs represents the data of 2 MLCs, this scheme reduces the number of Two-step Transitions to improve the lifetime.
- ES-FNW [16]: The data bits of the soft and hard domain are encoded by FNW respectively. In this work, we give each 4 data bits 1 flip tag to drastically reduce bit flips.
- HSC+ES-FNW: Encoding the cache lines that can be compressed to half-size with HSC. Besides, the other cache lines are encoded by ES-FNW, and each 4 data bits have 1 flip bit.
- OSwrite: Encoding cache lines with HSC and HTRE to write MLC STT-RAM in one-step.

1) Lifetime: Oswrite can reduce the wear to soft domain thus improve lifetime. We use the following formula to indicate the lifetime of MLC STT-RAM.

$$lifetime = \frac{N_{soft\_wearing}}{capacity} \tag{5}$$

$N_{soft\_wearing}$ represents the wearing number of the soft domain of MLC. The capacity in our scheme doesn't include the SLC hard flag since SLC has a much longer lifetime. Moreover, the hard flag cannot be firstly worn out since it suffers from much fewer wears than MLC STT-RAM. We assume some wear-leveling techniques [14] [29] are used to balance the cell non-uniform write. Fig. 13 shows the normalized lifetime of different schemes. SLC STT-RAM doesn't appear in the figure since it has a much longer lifetime than MLC. TSTM, ES-FNW, HSC+ES-FNW, OSwrite can improve lifetime by 1.62×, 1.7×, 2.03×, 2.6×, respectively compared with DCW. OSwrite reduces a large portion of Hard Transition and Two-step Transition thus significantly reduces the wears to soft domain. Furthermore, OSwrite can further reduce Soft Transition to improve the lifetime of MLC STT-RAM.

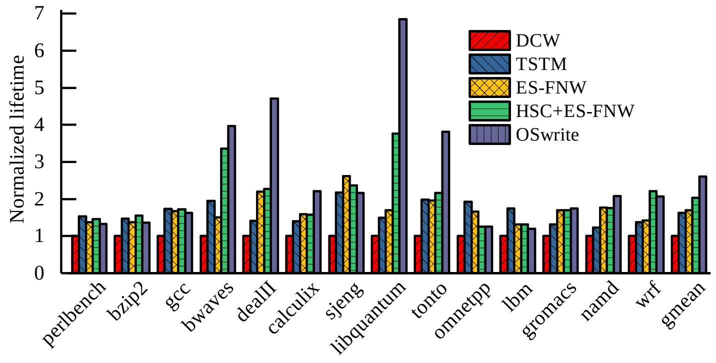

Fig. 13: Normalized lifetime.

2)Bit flips: The calculation of bit flips contains the data bits, tag bits, and the SLC hard flag. The result of normalized soft bit flips of several schemes is shown in Fig. 14, and soft bit flipping is caused by Soft Transition and Two-step Transition. SLC STT-RAM, TSTM, ES-FNW, HSC+ES-FNW, OSwrite can reduce soft bit flips by -169.8%, -17.7%, 29.5%, 25.8%, 5.3% on average, respectively. Due to the small capacity, SLC receives many cache miss write requests from the memory side. TSTM leads to more soft bit flips due to writing 50% extra MLCs. OSwrite brings extra writes on SLC hard flag, thus the soft bit flips reduction is not very significant.

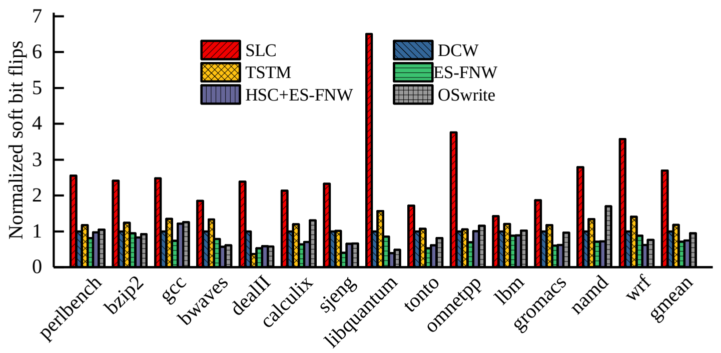

Fig. 14: Normalized soft bit flips.

Fig. 15 shows normalized hard bit flips. Hard bit flipping is caused by Hard Transition and Two-step Transition. TSTM, ES-FNW, HSC+ES-FNW, OSwrite can reduce hard bit flips by 5.2%, 27.0%, 54.4%, 82.8%. OSwrite can largely remove Hard Transition and Two-step Transition thus improve the lifetime. Hard Transition and Two-step Transition can be nearly totally removed in *gcc, bwaves, calculix, sjeng, libquantum* etc. This is because almost all cache lines are encoded by HSC and HTRE.

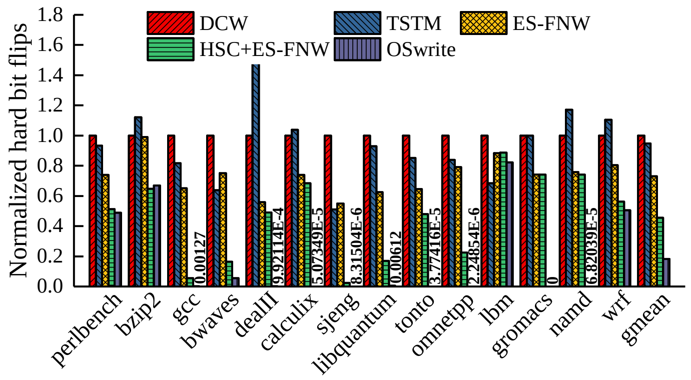

Fig. 15: Normalized hard bit flips.

3) Write energy: Fig. 16 shows normalized dynamic write energy of several schemes, and SLC STT-RAM, TSTM, ES-FNW, HSC+ES-FNW, OSwrite can reduce write energy by 9.4%, -3.0%, 28.2%, 43.8%, 56.2% compared with DCW. TSTM has additional 50% MLCs, which brings extra energy consumption. ES-FNW can reduce soft bit flips and hard bit flips to reduce energy consumption. OSwrite obtains the most significant write energy reduction by greatly decreasing Hard Transition and Two-step Transition.

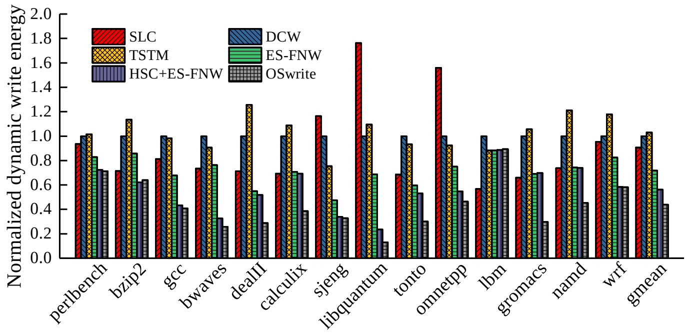

Fig. 16: Normalized dynamic write energy.

4) Performance: The cache access latency is determined by the worst cell write latency. OSwrite can reduce write latency by one-step write, thus improve the performance of the system. Although the encoding and decoding process brings extra latency, this overhead is much smaller compared with write latency.

We use Instruction Per Cycle (IPC) speedup to evaluate system performance. The IPC speedup is defined as follows.

$$IPC_{speedup} = \frac{IPC}{IPC_{baseline}} \qquad (6)$$

Fig. 17 depicts the normalized IPC speedup of several schemes. SLC STT-RAM, TSTM, ES-FNW, HSC+ES-FNW, OSwrite can improve IPC by -9.2%, -0.9%, -0.6%, 3%, 6.4% compared with DCW. Although the SLC has a lower access latency, the capacity miss penalty leads to worse performance than the MLC. ES-FNW and TSTM bring extra latency because of the encoding and decoding process, resulting in little performance loss. OSwrite can half the access latency thus improve the system performance.

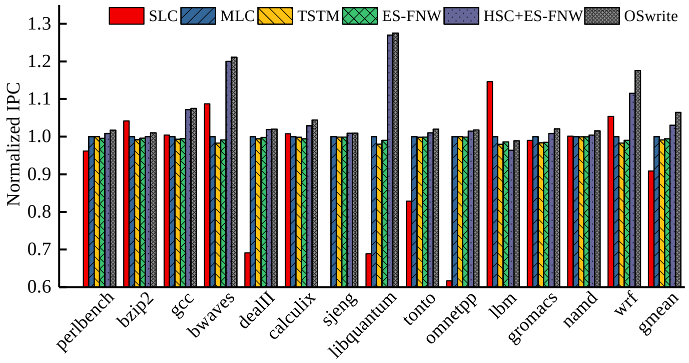

Fig. 17: Normalized IPC speedup.

V. Related works

A. Improving the lifetime of STT-RAM

Some existing works have proposed some wear-leveling techniques to improve the lifetime of STT-RAM. Chen *et al.* [10] proposed a set-remapping wear leveling for MLC STT-RAM cache. Sparsh *et al.* [13] proposed LastingNVcache to improve the lifetime by mitigating the intra-set write variation. i^2WAP [14] improved non-volatile cache lifetime by reducing both the inter-set and intra-set write variations.

Meanwhile, several works proposed encoding techniques to improve lifetime by reducing Two-step Transitions. Luo *et al.* [9] proposed Two-Step State Transition Minimization (TSTM) encoding which minimizes the Two-step Transition by choosing the optimal encoding type. Xu *et al.* [16] proposed ES-FNW to execute FNW encoding separately for the soft and hard domain data, thus improving lifetime by reducing both the soft and hard bit flips. Ahmad *et al.* [15] extended the lifetime by minimizing two-step and hard state transitions in hot bits.

B. Reducing the write energy of STT-RAM

The dynamic write energy of STT-RAM is higher than SRAM. Some existing works aim to reduce energy consumption at several levels. In circuit level, Bishnoi *et al.* [30] proposed Asynchronous Asymmetrical Write Termination (AAWT) which utilizes this asymmetrical behavior to terminate the write operations asynchronously and as a result significantly reduces the write power consumption. In architecture level, Ahn *et al.* [31] proposed Dead Write Prediction Assisted STT-RAM Cache Architecture (DASCA), which predicts and bypasses dead writes for write energy reduction. Wang *et al.* [18] proposed an architectural design to dynamically reconfigure the cache block size for an MLC STT-RAM last-level cache. This method placed certain hot data chunks in smaller blocks so as to benefit from the lower latency and energy. Besides, some encoding schemes such as ES-FNW [16] can reduce the hard and soft bit flips thus reduce the write energy.

C. Compression and Encoding schemes

Some works proposed encoding techniques for emerging non-volatile memory to reduce write energy and lower write latency. Liu [20]*et al.* proposed to use compression techniques to reduce the write energy of MLC STT-RAM. For Triple Level Cell (TLC) NVMs, CRADE [32] and CompEx [23] integrated data compression with expansion coding to reduce write energy and latency. Xu *et al.* [21] proposed COEF to compress the cache line and use encoding schemes to reduce bit flips of Phase Change Main memory. Zhang *et al.* [33] combined Frequent Pattern Compression technique and Incomplete Data Mapping [34] to reduce the write energy and access latency of TLC ReRAM.

VI. Conclusion

In this work, we propose One-Step write (OSwrite), which aims to improve the lifetime of MLC STT-RAM by reducing the wear to the soft domain. We analyze the write data pattern and find Hard Transition (HT) and Two-step Transition (TT) significantly reduce the lifetime of MLC STT-RAM, and Soft Transition (ST) also makes an impressive contribution to lifetime reduction. Then we propose OSwrite to mainly reduces HT and TT. Some light-weight encoding techniques are used to reduce ST as well. Evaluation results show that OSwrite can improve the lifetime of MLC STT-RAM to 2.6×, reduce write energy, and improve system performance by 56.2% and 6.4% respectively. Besides, OSwrite reduces hard bit flips and soft bit flips by 82.8% and 5.3% respectively.

Acknowledgment

We thank the anonymous reviewers for their constructive comments and suggestions. This work was supported in part by the National Natural Science Foundation of China under Grant 61832007, Grant 61821003, Grant 61772222, and Grant U1705261, in part by the Fundamental Research Funds for the Central Universities No.2019kfyXMBZ037, and in part by National Science and Technology Major Project No. 2017ZX01032-101.

References

[1] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, and J. Henkel, "Approximation-aware multi-level cells stt-ram cache architecture," in *Proceedings of the 2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. IEEE Press, 2015, pp. 79–88.

[2] C. Wang, D. Feng, W. Tong, J. Liu, B. Wu, W. Zhao, Y. Zhang, and Y. Chen, "Improving write performance on cross-point rram arrays by leveraging multidimensional non-uniformity of cell effective voltage," *IEEE Transactions on Computers*, 2020.

[3] C. Wang, D. Feng, W. Tong, J. Liu, Z. Li, J. Chang, Y. Zhang, B. Wu, J. Xu, W. Zhao, Y. Li, and R. Ren, "Cross-point resistive memory: Nonideal properties and solutions," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 24, no. 4, Jun. 2019. [Online]. Available: https://doi.org/10.1145/3325067

[4] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, 2016.

[5] X. Chen, N. Khoshavi, J. Zhou, D. Huang, R. F. DeMara, J. Wang, W. Wen, and Y. Chen, "Aos: adaptive overwrite scheme for energy-efficient mlc stt-ram cache," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[6] W. Wen, Y. Zhang, M. Mao, and Y. Chen, "State-restrict mlc stt-ram designs for high-reliable high-performance memory system," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–6.

[7] Y. Zhang, L. Zhang, W. Wen, G. Sun, and Y. Chen, "Multi-level cell stt-ram: Is it realistic or just a dream?" in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2012, pp. 526–532.

[8] X. Bi, M. Mao, D. Wang, and H. Li, "Unleashing the potential of mlc stt-ram caches," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2013, pp. 429–436.

[9] H. Luo, J. Hu, L. Shi, C. J. Xue, and Q. Zhuge, "Two-step state transition minimization for lifetime and performance improvement on mlc stt-ram," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 171.

[10] Y. Chen, W.-F. Wong, H. Li, and C.-K. Koh, "Processor caches with multi-level spin-transfer torque ram cells," in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*. IEEE Press, 2011, pp. 73–78.

[11] S. Aggarwal, H. Almasi, M. DeHerrera, B. Hughes, S. Ikegawa, J. Janesky, H. Lee, H. Lu, F. Mancoff, K. Nagel *et al.*, "Demonstration of a reliable 1 gb standalone spin-transfer torque mram for industrial applications," in *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2019, pp. 2–1.

[12] Y. Chen, X. Wang, W. Zhu, H. Li, Z. Sun, G. Sun, and Y. Xie, "Access scheme of multi-level cell spin-transfer torque random access memory and its optimization," in *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*. IEEE, 2010, pp. 1109–1112.

[13] S. Mittal, J. S. Vetter, and D. Li, "Lastingnvcache: A technique for improving the lifetime of non-volatile caches," in *2014 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2014, pp. 534–540.

[14] J. Wang, X. Dong, Y. Xie, and N. P. Jouppi, "i 2 wap: Improving non-volatile cache lifetime by reducing inter-and intra-set write variations," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 234–245.

[15] I. Ahmad, M. Imdoukh, and M. G. Alfailakawi, "Extending multi-level stt-mram cell lifetime by minimising two-step and hard state transitions in hot bits," *IET Computers & Digital Techniques*, vol. 11, no. 6, pp. 214–220, 2017.

[16] J. Xu, D. Feng, W. Tong, J. Liu, and W. Zhou, "Encoding separately: An energy-efficient write scheme for mlc stt-ram," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 581–584.

[17] S. Hong, J. Lee, and S. Kim, "Ternary cache: Three-valued mlc stt-ram caches," in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*. IEEE, 2014, pp. 83–89.

[18] J. Wang, P. Roy, W.-F. Wong, X. Bi, and H. Li, "Optimizing mlc-based stt-ram caches by dynamic block size reconfiguration," in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*. IEEE, 2014, pp. 133–138.

[19] J. L. Henning, "Spec cpu2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[20] L. Liu, P. Chi, S. Li, Y. Cheng, and Y. Xie, "Building energy-efficient multi-level cell stt-ram caches with data compression," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 751–756.

[21] J. Xu, D. Feng, Y. Hua, W. Tong, J. Liu, and C. Li, "Extending the lifetime of nvms with compression," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1604–1609.

[22] A. Alameldeen and D. Wood, "Frequent pattern compression: A significance-based compression scheme for l2 caches," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2004.

[23] P. M. Palangappa and K. Mohanram, "Compex: Compression-expansion coding for energy, latency, and lifetime improvements in mlc/tlc nvm," in *IEEE International Symposium on High Performance Computer Architecture*, 2016.

[24] S. Cho and H. Lee, "Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 347–357.

[25] A. N. Jacobvitz, R. Calderbank, and D. J. Sorin, "Coset coding to extend the lifetime of memory," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 222–233.

[26] B.-D. Yang, J.-E. Lee, J.-S. Kim, J. Cho, S.-Y. Lee, and B.-G. Yu, "A low power phase-change random access memory using a data-comparison write scheme," in *2007 IEEE International Symposium on Circuits and Systems*. IEEE, 2007, pp. 3014–3017.

[27] J. Xu, D. Feng, Y. Hua, W. Tong, J. Liu, C. Li, G. Xu, and Y. Chen, "Adaptive granularity encoding for energy-efficient non-volatile main memory," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 114.

[28] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[29] V. Young, P. J. Nair, and M. K. Qureshi, "Deuce: Write-efficient encryption for non-volatile memories," *ACM SIGPLAN Notices*, vol. 50, no. 4, pp. 33–44, 2015.

[30] R. Bishnoi, M. Ebrahimi, F. Oboril, and M. B. Tahoori, "Asynchronous asymmetrical write termination (aawt) for a low power stt-mram," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.

[31] J. Ahn, S. Yoo, and K. Choi, "Dasca: Dead write prediction assisted stt-ram cache architecture," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2014, pp. 25–36.

[32] J. Xu, D. Feng, Y. Hua, W. Tong, J. Liu, C. Li, and W. Zhou, "Improving performance of tlc rram with compression-ratio-aware data encoding," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 573–580.

[33] Y. Zhang, D. Feng, W. Tong, J. Liu, C. Wang, and J. Xu, "Tiered-reram: A low latency and energy efficient tlc crossbar reram architecture," in *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2019, pp. 92–102.

[34] D. Niu, Q. Zou, C. Xu, and Y. Xie, "Low power multi-level-cell resistive memory design with incomplete data mapping," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*. IEEE, 2013, pp. 131–137.