Adaptive Granularity Encoding for Energy-efficient Non-Volatile Main Memory

Jie Xu*, Dan Feng*, Yu Hua*, Wei Tong*, Jingning Liu*, Chunyan Li*, Gaoxiang Xu*, Yiran Chen§
*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
§Duke University, Durham, North Carolina, USA
Corresponding author: Wei Tong
{xujie_dsal, dfeng, csyhua, Tongwei, jnliu, lichunyan, xugaoxiang}@hust.edu.cn, yiran.chen@duke.edu

ABSTRACT

Data encoding methods have been proposed to alleviate the high write energy and limited write endurance disadvantages of Non-Volatile Memories (NVMs). Encoding methods are proved to be effective through theoretical analysis. Under the data patterns of workloads, existing encoding methods could become inefficient. We observe that the new cache line and the old cache line have many redundant (or unmodified) words. This makes the utilization ratio of the tag bits of data encoding methods become very low, and the efficiency of data encoding method decreases. To fully exploit the tag bits to reduce the bit flips of NVMs, we propose REdundant word Aware Data encoding (READ). The key idea of READ is to share the tag bits among all the words of the cache line and dynamically assign the tag bits to the modified words. The high utilization ratio of the tag bits in READ leads to heavy bit flips of the tag bits. To reduce the bit flips of the tag bits in READ, we further propose Sequential flips Aware Encoding (SAE). SAE is designed based on the observation that many sequential bits of the new data and the old data are opposite. For those writes, the bit flips of the tag bits will increase with the number of tag bits. SAE dynamically selects the encoding granularity which causes the minimum bit flips instead of using the minimum encoding granularity. Experimental results show that our schemes can reduce the energy consumption by 20.3%, decrease the bit flips by 25.0%, and improve the lifetime by 52.1%.

1 INTRODUCTION

The development of big data and in-memory computing has raised the requirement of large capacity of main memory. Typical capacity of DRAM chip is only 8Gb, 16Gb or 32Gb because the feature size of traditional DRAM has reached its limit. Besides, the dynamic energy of DRAM caused by the refresh operation increases significantly when the feature size reduces. Emerging non-volatile memories such as Phase Change Memory (PCM) and Resistive RAM (RRAM), which have the advantages of high density and non-volatility [2, 14, 17, 26], can compensate the shortcomings of DRAM and are promising to replace DRAM.

The design of NVM-based main memory also faces challenges. Main memory necessarily has unlimited write endurance ($> 10^{16}$) to tolerate the heavy writes. However, the write endurance of emerging NVMs is limited ($< 10^{10}$) [14, 19, 25]. The NVM cells may get stuck at "0" or "1" [16] and the main memory system will fail after certain cells endure the number of 10^{10} writes. NVMs also suffer from high write energy [8]. They will consume significant write energy if they are used as main memory directly. The limited write endurance and high write energy of NVMs are main obstacles and must be tackled if we want to build large and reliable main memory with NVMs.

Data encoding techniques have been proposed to alleviate the disadvantages of high write energy and limited write endurance of NVMs. Encoding methods transform the data into the vectors which have the minimum bit flips (i.e., the writes of "1" → "0" and "0" → "1"). Flip-N-Write [4, 7, 9, 11] reduces the bit flips by flipping the new data. The new data are divided into 8-bit, 16-bit or 32-bit block. Each block has a tag bit to indicate whether the bits should be flipped or not. Other encoding methods [6, 18, 21] map the data bits into a set of vectors and select the vector which has the minimum bit flips. Theoretical analysis shows that data encoding methods can significantly reduce the bit flips, and they can reduce more bit flips with smaller encoding granularity (i.e., fewer data bits are given one tag bit) [4, 6].

The efficiency of encoding schemes are verified in theory with random input data. However, encoding methods may become inefficient when considering the data patterns and characteristics of workloads. Existing works [2, 22] show that the dirty cache lines written into the main memory have a large number of clean (or unmodified) words. The tag bits assigned for the clean words are actually unused because the clean words need not to be encoded. Our experimental results show that 42.8% of the tag bits are wasted on average. The detailed is discussed in Section 3.1.1. To fully exploit the tag bits for reducing bit flips, we propose to adaptively assign the tag bits to the dirty (or modified) words only. The encoding granularity reduces and the bit flips/write energy decrease due to the fine-grained encoding. The high utilization ratio of READ also causes heavy bit flips of the tag bits. We further propose Sequential flips Aware Encoding (SAE) to reduce bit flips of the tag bits. SAE is designed based on the observation that sometimes each bit of the new data is opposite to the corresponding bit of the old data, e.g., the new data are "0xFFFF" and the old data are "0x0000". There

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317760

are 16 "Sequential flips" when writing the new data. Theoretical analysis of encoding methods shows that smaller encoding granularity can reduce more bit flips. However, in the case of "Sequential flips", smaller encoding granularity leads to more bit flips of tag bits. SAE calculates the bit flips of different encoding granularities and dynamically select the encoding granularity which causes the minimum bit flips. The main contributions of this work are as follows:

- We observe that a large number of clean words exist in the dirty cache lines, and this leads to low utilization ratio of the tag bits and inefficiency of encoding methods. To fully exploit the tag bits for reducing bit flips, we propose to share the tag bits among all the words of the cache line and assign the tag bits to the dirty words only.
- We observe that main memory could be written into the opposite data, and smaller encoding granularity results in more bit flips of the tag bits under the circumstance. To reduce the bit flips of the tag bits, we propose to dynamically select the encoding granularity which causes the minimum bit flips.
- Experimental results show that our schemes can reduce the energy consumption by 20.3%, decrease the bit flips by 25.0%, and improve the lifetime by 52.1%.

The rest of this paper is organized as follows. Section 2 describes the background and related Work. Section 3 introduces the observations and techniques proposed. Section 4 and 5 present the experiments and conclusion.

2 BACKGROUND AND RELATED WORK

2.1 Background

Emerging Non-Volatile Memories (NVMs) such as Phase Change Memory (PCM) and Resistive RAM (RRAM) have advantages in low leakage power, non-volatility and high density [8, 20]. They are promising replacements of DRAM and can be used to build large-size main memory. However, they also face the challenges of high write energy and limited write endurance. The write energy of PCM and RRAM are ten times more than DRAM [4, 22]. NVMs will consume significant write energy if they are used without optimization. Besides, the write endurance of NVM is $10^8 \sim 10^{10}$, which is six orders of magnitude fewer than DRAM. The NVM cells may fail after they endure a certain amount of writes, and the failure may result in the breakdown of the computer system [14, 16].

2.2 Related Work

Data encoding techniques have been proposed to alleviate the limited lifetime and high write energy problems of NVMs. Data Comparison Write (DCW) [23] reads the old data out and eliminates the writes to the redundant bits. Recently manufactured NVM [10] even integrates the DCW function into the chip. Flip-N-Write [4] gives every N data bits one tag bit. If the number of bit flips of writing the N-bit data with its tag bit exceeds $(N+1)/2$, the data bits will be flipped and the tag bit will be set. CAFO [9] rearranges each N-bit data into an $n \times m$ matrix, where $N = n \times m$. Flip-N-Write is applied in all the rows and columns. FlipMin [6] and Pres-Random

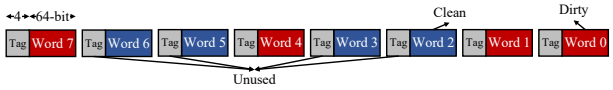

Figure 1: The dirty cache line has four clean words and half of the tag bits are unused.

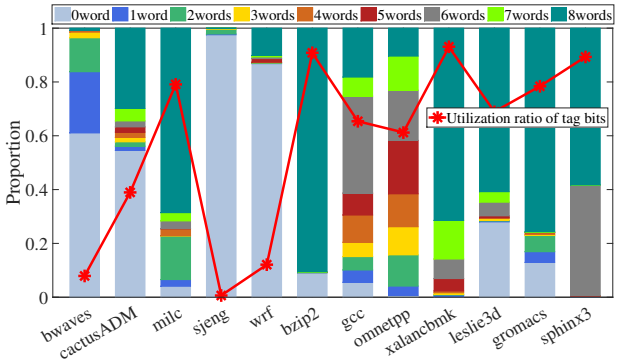

Figure 2: The number of dirty words and the utilization ratios of the tag bits in different benchmarks.

Encoding [18] are designed based on coset coding. FlipMin [6] maps the data bits into a set of vectors and selects the vector which causes the minimum bit flips. Pres-Random Encoding [18] also maps the data bits into vectors, but it can increase the randomness of the vectors generated. Other works [7, 11, 21] combine Flip-N-Write or coset coding with compression and access locality. AFNW [11] compresses the words first, and then adapts the tag bits to the compressed words. Captopril [7] can reduce the bit flips of the hot locations. COE [21] stores the tag bits of the data encoding methods in the space saved by compression. The encoding methods are all theoretically efficient with random data. The accesses of workloads exhibit many characteristics. Considering the features of the data, existing encoding methods become inefficient.

3 OBSERVATIONS AND TECHNIQUES

We observe that the utilization ratio of the tag bits is low due to the large number of clean words. To fully exploit the tag bits for reducing bit flips, we propose REdundant word Aware Data encoding (READ). Since READ leads to increased bit flips of the tag bits, we also propose Sequential flips Aware Encoding (SAE) to reduce the bit flips of the tag bits.

3.1 Improving the utilization ratio of tag bits

In this subsection, we propose READ based on the observation that the utilization ratio of the tag bits is very low.

3.1.1 Observation. Prior works [2, 22] have already shown that a large number of words in the dirty cache lines are clean. A large number of clean words are generated owing to two reasons. On the one hand, the write granularity of the CPU is word. Even if only one word of the cache line is written, the entire cache line will be marked as dirty and evicted into the next-level memory. On the other hand, CPU may write the same data. For example, the 64-bit data "0x0000000000000000" are very common and may be written twice. Although the same data are written, the cache line will be marked as dirty. The clean words result in the low utilization ratio of the tag bits. If we give every 64-bit word four

tag bits, the clean words need not to be encoded and the tag bits will actually be wasted in the clean words. For a dirty cache line with four clean words, only half of the tag bits are actually used, as shown in Figure 1. We also analyze the utilization ratios of the tag bits in the benchmarks selected from SPEC CPU 2006 [5] through experiments. The word size and cache line size are 64 and 512 bits, respectively. The number of dirty words and utilization ratios of the tag bits in different benchmarks are shown in Figure 2. In some benchmarks such as bwaves and sjeng, the cache lines with fewer dirty words dominate all the cache lines. Therefore, the utilization ratios of the tag bits in the two benchmarks are very low. About 60% of the cache lines in bwaves have zero modified word and the utilization ratio of the tag bits in bwaves is only 8.0%. In the xalancbmk benchmark, 90% of all the cache lines have seven or eight modified words and the utilization ratio of the tag bits in xalancbmk is the highest (93.0%). The average utilization ratio of the tag bits in the benchmarks is 57.2%. About half of the tag bits are unused.

3.1.2 REdundant word Aware Data encoding. To fully exploit the tag bits for reducing the bit flips and write energy, we propose REdundant word Aware Data encoding (READ). In the prior encoding techniques [4, 6, 9], each word has a constant N-bit tag. The tag bits of the clean words are wasted and cannot be used by the dirty words. Different from them, we globally assign the tag bits to the entire cache line. If the word of the cache line is clean, no tag bit will be assigned. Meanwhile, the saved tag bits are assigned to the dirty words. Through sharing the tag bits among all the words and assigning the tag bits to the dirty words only, the dirty words can have smaller encoding granularity. On the other hand, data encoding techniques [4, 6, 21] reduce more bit flips with smaller encoding granularity. Figure 3 shows the relationship between the encoding granularity and bit flip reduction in Flip-N-Write. The bit flip reduction of every four data bits one tag bit is 21.9%. The reduction decreases to 14.6% if every sixteen data bits are given one tag bit. READ can reduce the encoding granularity, thereby reducing more bit flips.

Figure 4 illustrates an example of READ. The total number of bits of the tag bits is 32. We assume that four of the eight words in the cache line are dirty. The eight words of a cache line share the 32-bit tag in READ. For the clean words, READ does not assign tag bits to them because they need not to be encoded. Each of the four dirty words now has 8-bit tag. One tag bit indicates whether the 8-bit data needs to be flipped. The encoding granularity decreases from 16 to 8, and therefore the efficiency of the encoding increases. To indicate whether the word is clean or dirty, we give each word one dirty flag.

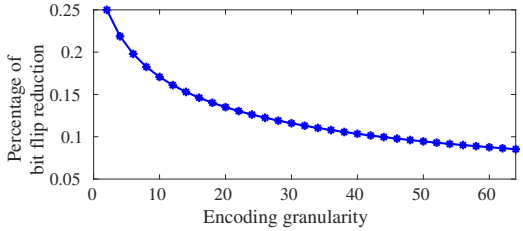

Figure 3: The relationship between the encoding granularity and bit flip reduction.

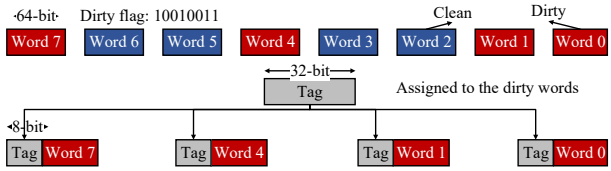

Figure 4: REdundant word Aware Encoding (READ) globally assigns the tag bits to the dirty words.

3.2 Reducing bit flips of the tag bits

The high utilization ratio of tag bits in READ leads to serious bit flips of the tag bits. The bit flips of the tag bits increase by 145.7% through our experiments. The detailed experimental results are shown in Section 4.2.3. Based on the observation of "Sequential flips", we propose Sequential flips Aware Encoding (SAE) to reduce the bit flips of the tag bits.

3.2.1 Observation. Theoretical analysis shows that data encoding methods can reduce more bit flips with more capacity overhead. As shown in Figure 3, Flip-N-Write reduces more bit flips with smaller encoding granularity. Considering the data patterns of workloads, smaller encoding granularity may lead to more bit flips of the tag bits rather than reduce the bit flips. Both the 32-bit data "0x00000000" and "0xFFFFFFFF" are frequent values [1, 12]. If the new data and the old data are 32-bit "0xFFFFFFFF" and 32-bit "0x00000000" respectively, each bit of the new data and the old data will exactly be the opposite. There are 32 sequential bit flips when writing the new data. Since the new data and old data are the opposite, actually only one tag bit is required to indicate that the all new data bits are flipped. If we give one tag bit for the 32-bit new data, the tag bit will incur one additional bit flip. If more tag bits are used, the bit flips of the tag bits will be higher and the total number of the bit flips will increase rather than decrease. Figure 5 illustrates an example which shows the bit flips of the tag bits can be more serious with smaller encoding granularity. If the number of the tag bits is sixteen (every four data bits have one tag bit), the bit flips of the data bits and the tag bits will be zero and sixteen respectively. For the sequential flips, the total number of bit flips is subject to the tag bits, and fewer tag bits can reduce more bit flips. We evaluate the frequency of sequential flips at the granularity of byte. About 11.7% of the writes are sequential flips in the "sjeng" benchmark.

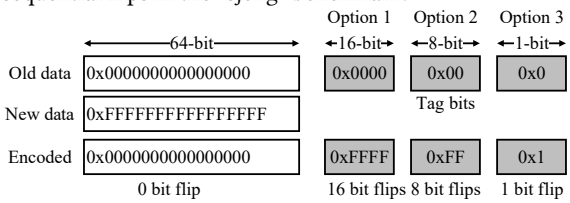

Figure 5: Smaller encoding granularity leads to more bit flips in the case of "Sequential flips".

3.2.2 Sequential flips Aware Encoding. To reduce the bit flips of the tag bits, we propose Sequential flips Aware Encoding (SAE). SAE dynamically selects the encoding granularity which leads to the minimum bit flips, instead of fixedly using the most fine-grained encoding. We use a 2-bit granularity flag to indicate the encoding granularity. To reduce the capacity overhead of the granularity flag, the 2-bit granularity flag is used for the entire cache line. We generate four alternative encoding granularities by setting the value

of the granularity flag. Initially, each word has an N-bit tag and the granularity flag is "00". We reduce the number of the tag bits to $N/2$, $N/4$ or $N/8$ when the granularity flag is "01", "10" or "11". The new cache line is encoded with the four encoding granularities in parallel, as described in Figure 6. After calculating the bit flips of the four different choices, the encoding granularity which causes the minimum bit flips is selected. Due to the selective encoding, SAE can reduce the total number of bit flips. At the same time, the bit flips of the tag bits reduce due to the coarse-grained encoding. Note that SAE aims to reduce the bit flips rather than balance the wear rate, since many efficient wear-leveling schemes [14, 15, 19, 24] have already been proposed.

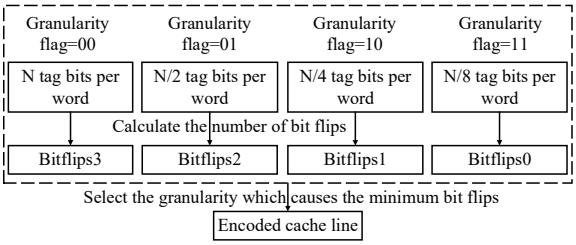

Figure 6: Sequential flips Aware Encoding (SAE) dynamically selects the encoding granularity which leads to the minimum bit flips.

3.3 Putting together

READ and SAE aim to reduce the total number of bit flips and the bit flips of the tag bits respectively. The two proposed techniques can work together. For the write operations, we first use the READ scheme to improve the utilization ratio of the tag bits. Then, we use SAE to reduce the bit flips of the tag bits. An example is shown in Figure 7. We give each word one dirty flag, and the cache line has an 8-bit dirty flag. To find the dirty words, we read the old data out and compare the old data with the new data at the granularity of word. The ith bit of the dirty flag is set if the ith word is dirty. We use M and N to represent the number of the dirty words and the number of the tag bits. The minimum encoding granularity is every $M \times 64/N$ data bits one tag bit. We generate the other three options of the encoding granularity according to the granularity flag. The encoding granularity can be obtained according to Table 1. Then, we calculate the bit flips of the four different encoding schemes. The encoded cache line which has the minimum bit flips is selected.

Table 1: Encoding granularities of READ+SAE

Granularity flag	Tag bits (per cache line)	Encoding granularity
00	N	$64 \times M/N$
01	N/2	$128 \times M/N$
10	N/4	$256 \times M/N$
11	N/8	$512 \times M/N$

For the read operations, the encoded cache line is decoded before forwarding. An example of the decoding process is illustrated in Figure 8. We first identify the dirty words of the cache line according to the dirty flag. The encoding granularity is determined according to the granularity flag and the number of dirty words. In the example shown in Figure 8, the number of the tag bits is "32"

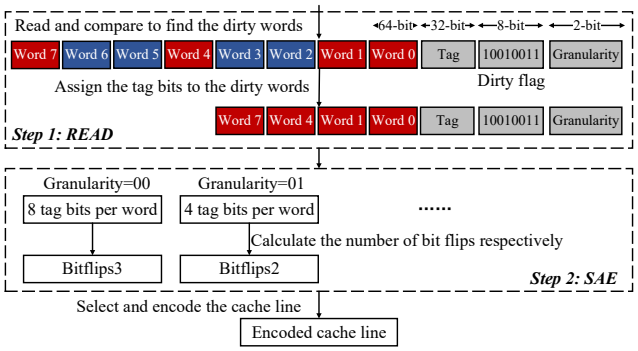

Figure 7: Encoding process of READ+SAE.

and the granularity flag is "11". Four (32/8) tag bits are used. The dirty flag is "10010011". Therefore, four words (i.e., Word 7, Word 4, Word 1 and Word 0) need to be decoded by Flip-N-Write. The encoding granularity is every 64 data bits 1 tag bit according to Table 1. We use the decoding process of Flip-N-Write to decode the dirty words. After decoding, the decoded words are merged with the clean words.

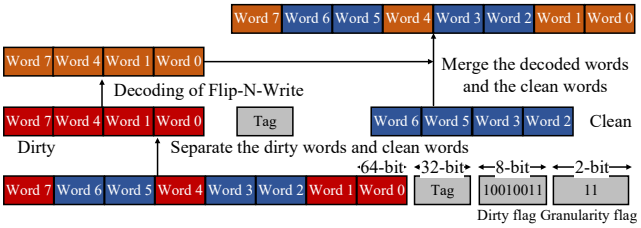

Figure 8: Decoding process of READ+SAE.

3.4 Overhead

We discuss the overhead of our schemes in this subsection.

3.4.1 Capacity overhead. The 512-bit cache line shares the 32-bit encoding tag. Besides, we have an 8-bit dirty flag to indicate whether the word is clean or not. Another 2-bit granularity flag is used to indicate the percentage of the tag bits used. The total number of bits of the additional space is $32 + 8 + 2 = 42bits$ per cache line. The estimated capacity overhead is 42/512=8.2%.

3.4.2 Latency/logic overhead. The decoding and encoding processes of our schemes incur latency and logic overhead. The decoding process of our scheme is similar to Flip-N-Write, and the latency/logic overhead of decoding is negligible. To estimate the encoding overhead, we use Verilog hardware description language to implement the encoding process. Then, we synthesize the logic in Synopsys Design Compiler based on $90nm$ technology size. The estimated logic overhead is about $171K$ gates. Assuming that the size of a logic gate is the same as an NVM cell, the logic overhead is only 0.004% of a $4GB$ main memory. The energy consumption of encoding is $81.65pJ$. Since the write energy of a PCM cell is about $20pJ$ [8], the energy consumption of encoding is less than 1% of the write energy of a cache line. We scale the encoding latency down to $22nm$ technology size. The estimated encoding latency is $3.47ns$. The system performance is mainly determined by the read performance. Since our schemes have negligible decoding latency and hardly increase the read latency, the performance degradation is negligible.

4 EVALUATION

4.1 Experimental configuration

We use the cycle-accurate system simulator Gem5 [3] to evaluate the efficiency of our schemes. For the simulation of main memory, we use the NVMain simulator [13], which is capable of simulating emerging NVMs at the architectural level. We also modify the main memory model of NVMain to support our schemes. The system configuration is shown in Table 2. The four-core system consists of three levels of caches. The main memory is based on 4GB PCM. Twelve memory-intensive benchmarks selected from SPEC CPU 2006 [5] are used in our experiments. To verify the efficiency of our schemes, we evaluate the following seven different schemes.

- DCW [23]: DCW reads the old data and compares them with the new data before writing. Only the modified bits are written.
- Flip-N-Write [4]: Every eight data bits have one tag bit. The capacity overhead of Flip-N-Write is 12.5%.
- AFNW [11]: The new data are compressed before writing. The tag bits are assigned to the compressed data. We give each 64-bit word four tag bits.
- COEF [21]: The space saved by compression are exploited to store the tag bits of data encoding methods. The capacity overhead is 0.2%.
- CAFO [9]: The 512-bit cache line is rearranged into a 32×16 matrix. Both rows and columns of each matrix have one tag bit. The capacity overhead of CAFO is 9.4% (48/512).
- READ: The 512-bit cache line shares 32-bit tag. The 32-bit tag is assigned only to the dirty words. The cache line has 8-bit dirty flag. The capacity overhead is 7.8%.
- READ+SAE: Besides the working of READ, the encoding granularity is adaptive and the granularity which yields the minimum bit flips is used to encode the data. The capacity overhead is 8.2%.

Table 2: System configuration

Cores	4-Core, 3.2GHz, out-of-order
L1 I/D cache	private, 32KB/core, 2-way, 2-cycle latency
L2 Cache	private, 1MB/core, 8-way, 20-cycle latency
L3 Cache	shared, 16MB, 16-way, 50-cycle latency
Memory Organization	4GB PCM, Read 100ns, Write 150ns

4.2 Experimental results

The seven different schemes are evaluated in terms of the total number of bit flips, energy consumption, bit flips of the tag bits and lifetime. DCW is used as the baseline.

4.2.1 Total number of bit flips. The calculation of bit flips includes the flips of the data bits, tag bits and other additional flags (e.g., compression tag of AFNW, dirty flag of READ and granularity flag of SAE). The bit flips comparison of the seven different schemes is shown in Figure 9. Compared with the baseline (DCW), Flip-N-Write, AFNW, COEF, CAFO, READ and READ+SAE can reduce the bit flips by 15.1%, 5.1%, 12.5%, 17.8%, 23.2% and 25.0% respectively. Experimental results confirm that our schemes are the most efficient among all the seven schemes. READ can fully exploit the tag bits to encode the cache line, and therefore READ reduces 9.9% more bit flips than Flip-N-Write. READ+SAE reduces more bit flips than READ, because READ+SAE generates four different encoding choices and selects the encoding granularity which leads to the minimum bit flips. AFNW is less efficient than FNW because compression results in more bit flips than DCW. COEF exploits both Flip-N-Write and FlipMin to encode the compressed and gets the similar result to Flip-N-Write. CAFO is more efficient than Flip-N-Write because it encodes both the rows and columns of the cache lines.

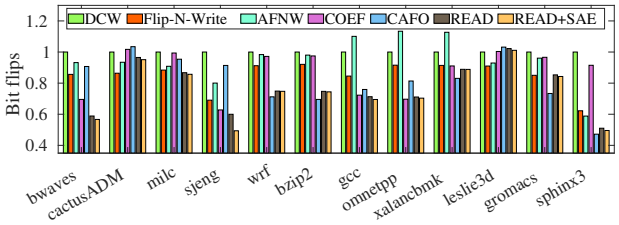

Figure 9: Comparison of bit flips.

4.2.2 Energy consumption. The energy consumption mainly includes the read energy and write energy of data bits, tag bits and flag bits. The encoding energy consumption is also considered in READ and READ+SAE. AFNW, COEF, CAFO, READ and READ+SAE can reduce the energy consumption by 12.4%, 3.6%, 9.2%, 16.6%, 19.2% and 20.3% compared with DCW, as described in Figure 10. The reduction of energy consumption exhibits the same tendency as bit flips because the write energy (caused by bit flips) dominates the total energy consumption. The energy consumption of other operations such as reads is the same in all the seven schemes. Hence, the reduction of energy consumption is less than the reduction of bit flips. READ and READ+SAE reduce the most energy consumption among all the schemes due to their efficiency in reducing bit flips.

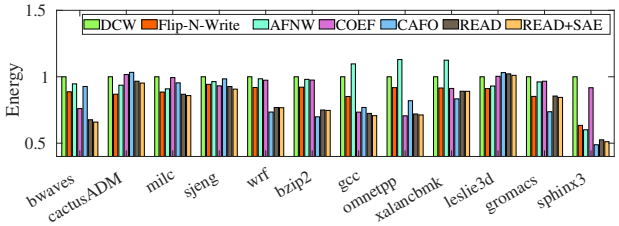

Figure 10: Comparison of energy consumption.

4.2.3 Bit flips of the tag bits. Figure 11 illustrates the bit flips of the tag bits of Flip-N-Write, AFNW, CAFO, READ and READ+SAE. DCW has no tag bit and COEF only uses 1-bit tag. Therefore, DCW and COEF are not shown in the figure. The bit flips of tag bits of AFNW, CAFO, READ and READ+SAE increases by 23.4%, -32.4%, 145.7% and 113.9% compared with Flip-N-Write. All the tag bits are used to encode the cache line in READ, and therefore the number of the bit flips of the tag bits is the largest. Compared with READ, READ+SAE can reduce the bit flips of the tag bits by 21.8%. The reason is SAE dynamically selects the granularity which causes the minimum bit flips instead of always using all the tag bits. Although the bit flips of the tag bits increase significantly, the total number of bit flips is reduced, and we can use the existing wear-leveling techniques such as Security Refresh [15], Start-Gap [14] and HWL [24] to balance the wear rate.

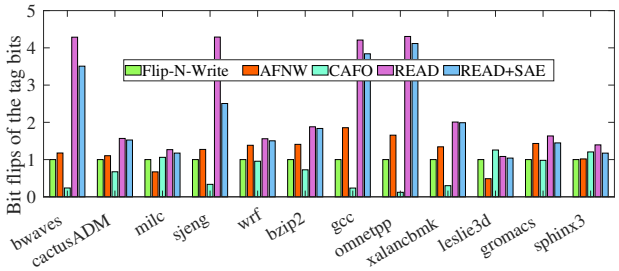

Figure 11: Comparison of the bit flips of the tag bits.

4.2.4 Lifetime. The total number of bit flips that the main memory can endure is limited. Our schemes can reduce the bit flips, and thus the lifetime is improved. Although different cells may suffer from non-uniform writes, wear-leveling techniques [14, 15, 19, 24] can balance the wear rate. For example, HWL [24] can obtain nearly the same lifetime as perfect wear leveling. We assume that wear-leveling techniques are applied, and the lifetime improvements are proportional to the bit flip reduction. The lifetime improvements of Flip-N-Write, AFNW, COEF, CAFO, READ and READ+SAE compared with DCW are 34.3%, 15.3%, 17.9%, 35.1%, 46.2% and 52.1%, as shown in Figure 12. The improvement of READ+SAE is the most since its bit flip reduction is the most.

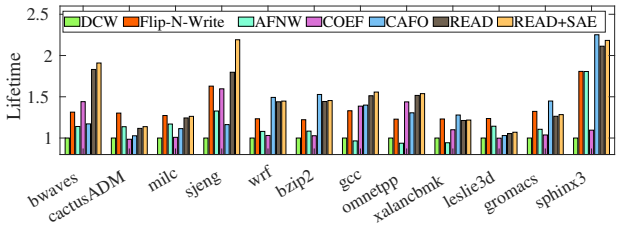

Figure 12: Comparison of lifetime.

5 CONCLUSION

Emerging NVMs have disadvantages of limited write endurance and high write energy. We observe that a large number of words in dirty cache lines are clean, and the new data and old data are opposite in some cases. These characteristics lead to the low utilization ratio and increased bit flips of the tag bits. To improve the utilization of tag bits, we propose READ to share the tag bits among all the words of the cache line and assign the tag bits to the modified words. To reduce the bit flips of tag bits, we propose SAE, which selects the encoding granularity which causes the minimum bit flips. Experimental results show that our schemes can reduce the energy consumption by 20.3%, improve the lifetime by 52.1% and decrease the bit flips by 25.0%.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61821003, Grant 61832007, Grant 61772222, Grant U1705261 and Grant 61772212, in part by the National High Technology Research and Development Program under Grant 2015AA015301, and in part by the Shenzhen Research Funding of Science and Technology under Grant JCYJ20170307172447622. This work was also supported by Key Laboratory of Information Storage System, Ministry of Education, China.

REFERENCES

[1] Angelos Arelakis, Fredrik Dahlgren, and Per Stenstrom. 2015. HyComp: A Hybrid Cache Compression Method for Selection of Data-type-specific Compression Methods. In *Proceedings of MICRO (MICRO-48)*. ACM, New York, NY, USA, 38–49.
[2] Mohammad Arjomand, Mahmut T Kandemir, Anand Sivasubramaniam, and Chita R Das. 2016. Boosting access parallelism to PCM-based main memory. In *Proceedings of ISCA*. 695–706.
[3] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.
[4] Sangyeun Cho and Hyunjin Lee. 2009. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In *Proceedings of MICRO*. IEEE, 347–357.
[5] John L Henning. 2006. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News* 34, 4 (2006), 1–17.
[6] Adam N Jacobvitz, Robert Calderbank, and Daniel J Sorin. 2013. Coset coding to extend the lifetime of memory. In *Proceedings of HPCA*. IEEE, 222–233.
[7] Majid Jalili and Hamid Sarbazi-Azad. 2016. Captopril: Reducing the pressure of bit flips on hot locations in non-volatile main memories. In *Proceedings of DATE*. IEEE, 1116–1119.
[8] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting Phase Change Memory As a Scalable Dram Alternative. In *Proceedings of ISCA*.
[9] Rakan Maddah, Seyed Mohammad Seyedzadeh, and Rami Melhem. 2015. CAFO: Cost aware flip optimization for asymmetric memories. In *Proceedings of HPCA*. IEEE, 320–330.
[10] H. Noguchi et al. 2016. 7.2 4Mb STT-MRAM-based cache with memory-access-aware power optimization and write-verify-write / read-modify-write scheme. In *Proceedings of ISSCC*.
[11] Poovaiah M Palangappa and Kartik Mohanram. 2015. Flip-Mirror-Rotate: An architecture for bit-write reduction and wear leveling in non-volatile memories. In *Proceedings of GLSVLSI*. ACM, 221–224.
[12] Poovaiah M. Palangappa and Kartik Mohanram. 2017. CompEx++: Compression-Expansion Coding for Energy, Latency, and Lifetime Improvements in MLC/TLC NVMs. *ACM Trans. Archit. Code Optim.* 14, 1 (April 2017).
[13] Matthew Poremba, Tao Zhang, and Yuan Xie. 2015. Nvmain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems. *IEEE Computer Architecture Letters* 14, 2 (2015), 140–143.
[14] Moinuddin K Qureshi, John Karidis, Michele Franceschini, Vijayalakshmi Srinivasan, Luis Lastras, and Bulent Abali. 2009. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proceedings of MICRO*. ACM, 14–23.
[15] Nak Hee Seong, Dong Hyuk Woo, and Hsien-Hsin S. Lee. 2010. Security Refresh: Prevent Malicious Wear-out and Increase Durability for Phase-change Memory with Dynamically Randomized Address Mapping. In *Proceedings of ISCA (ISCA '10)*. 383–394.
[16] Nak Hee Seong, Dong Hyuk Woo, Vijayalakshmi Srinivasan, Jude A. Rivers, and Hsien-Hsin S. Lee. 2010. SAFER: Stuck-At-Fault Error Recovery for Memories. In *Proceedings of MICRO*. 115–124.
[17] S. Seyedzadeh, A. Jones, and R. Melhem. 2018. Enabling Fine-Grain Restricted Coset Coding Through Word-Level Compression for PCM. In *Proceedings of HPCA*. 350–361.
[18] Seyed Mohammad Seyedzadeh, Rakan Maddah, Alex Jones, and Rami Melhem. 2015. PRES: Pseudo-random Encoding Scheme to Increase the Bit Flip Reduction in the Memory. In *Proceedings of DAC*. 23:1–23:6.
[19] Wen Wen, Youtao Zhang, and Jun Yang. 2018. Wear Leveling for Crossbar Resistive Memory. In *Proceedings of DAC (DAC '18)*. 58:1–58:6.
[20] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie. 2015. Overcoming the challenges of crossbar resistive memory architectures. In *Proceedings of HPCA*. 476–488.
[21] J. Xu, D. Feng, Y. Hua, W. Tong, J. Liu, and C. Li. 2018. Extending the lifetime of NVMs with compression. In *Proceedings of DATE*. 1604–1609.
[22] J. Xu, D. Feng, Y. Hua, W. Tong, J. Liu, C. Li, and Z. Li. 2018. An efficient PCM-based main memory system via exploiting fine-grained dirtiness of cachelines. In *Proceedings of DATE*. 1616–1621.
[23] Byung-Do Yang, Jae-Eun Lee, Jang-Su Kim, Junghyun Cho, Seung-Yun Lee, and Byoung-Gon Yu. 2007. A low power phase-change random access memory using a data-comparison write scheme. In *Proceedings of ISCAS*. IEEE, 3014–3017.
[24] Vinson Young, Prashant J. Nair, and Moinuddin K. Qureshi. 2015. DEUCE: Write-Efficient Encryption for Non-Volatile Memories. In *Proceedings of ASPLOS (ASPLOS '15)*.
[25] L. Zhang, B. Neely, D. Franklin, D. Strukov, Y. Xie, and F. T. Chong. 2016. Mellow Writes: Extending Lifetime in Resistive Memories through Selective Slow Write Backs. In *Proceedings of ISCA*. 519–531.
[26] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. 2009. A durable and energy efficient main memory using phase change memory technology. In *Proceedings of ISCA*. ACM, 14–23.