ROBIN: Incremental Oblique Interleaved ECC for Reliability Improvement in STT-MRAM Caches

Elham Cheshmikhani, Hamed Farbeh, and Hossein Asadi
Sharif University of Technology
Tehran, Iran

Abstract— *Spin Transfer Torque-Magnetic RAM* (STT-MRAM) is a promising alternative for SRAMs in on-chip cache memories. Besides all its advantages, high error rate in STT-MRAM is a major limiting factor for on-chip cache memories. In this paper, we first present a comprehensive analysis that reveals that the conventional *Error-Correcting Codes* (ECCs) lose their efficiency due to data-dependent error patterns, and then propose a near-optimal ECC configuration, so-called *ROBIN*, to maximize the correction capability. The evaluations show that the inefficiency of conventional ECC increases the cache error rate by an average of 151.7% while ROBIN reduces this value by more than 28.6x.

I. Introduction

Spin Transfer Torque-Magnetic RAM (STT-MRAM) has attracted considerable research interests and efforts in recent years. Non-volatility, near-zero leakage power, high density, and immunity to radiation-induced particle strike persuade the designers to replace conventional SRAM technology with STT-MRAM in *Last-Level Caches* (LLCs) [1, 2]. Beside all of STT-MRAM technology advantages, it suffers from three error types: *write failure* (i.e., unsuccessful cell flip during a write operation), *read disturbance* (i.e., unintentional cell flip during a read operation), and *retention failure* (i.e., stochastic cell flip during cell idle interval) [3, 4]. To make STT-MRAM technology commercialized in on-chip caches, these reliability challenges should be carefully addressed by designers.

Considering the three mentioned error types, the rate of write failure is significantly higher than the two other errors in nanoscale technology feature size. Due to short lifetime of data blocks in LLCs, retention failure is an extremely rare event in STT-MRAM cells with 10 years retention time. Meanwhile, the read disturbance rate is negligible compared to write failure rate since the read current is much lower than the write current [5, 6]. Therefore, write failure is the major reliability challenge in STT-MRAM caches.

Employing *Error-Correcting Codes* (ECCs) is the most conventional scheme to overcome write failure in STT-MRAM LLCs [7, 8, 9, 10, 11]. Conventionally, each N-bit cache block is divided into several k-bit datawords, each of which is protected by an r-bit ECC. There are two common configurations for partitioning cache block cells to construct $(k+r)$-bit codewords. In *per-word* ECC configuration, a codeword is generated by k consecutive bit positions and in *interleaved ECC* configuration, bit positions with distance of N/k contribute in generating each codeword.

Since write failure is likely to occur *only* in cache cells that need to flip in a write operation, the write failure rate is proportional to the number of transitions required. When partitioning a block into multiple codewords, each capable of correcting single bit error, the probability of a correct write operation is dominated by a codeword with the maximum number of transitions. This number should be minimized for maximizing the ECC efficiency, which can be achieved by uniformly distributing the total number of transitions between the codewords. This uniformity is addressed in neither per-word nor interleaved ECCs. To improve the efficiency of ECCs and provide higher error correction capabilities, the ECC configuration should be redesigned based on STT-MRAM error patterns and customized according to its characteristics and requirements, which has been addressed in none of the previous studies.

In this paper, we first demonstrate the inefficiency of existing ECC configurations and then propose a near-optimal ECC configuration that effectively improve the reliability of STT-MRAM LLCs. In the first contribution, we conduct a deep investigation on the distribution of transitions between codewords of cache blocks in write operations and observe a large variations between the number of transitions in different codewords for both per-word and interleaved configurations, which cause a significant reliability degradation. In the second contribution, we propose *Incremental Oblique Interleaved* (ROBIN)

ECC configuration to uniformly distribute the transitions between the codewords. ROBIN selects the data bits of each codeword in such a way that all bytes of a cache block as well as all bit positions in all bytes equally contribute in all codewords. Meanwhile, the bit positions of the bytes in each word are shifted in such a way that bits of different byte positions in all words uniformly contribute in all codewords. The uniformity achieved by ROBIN significantly improves the ECCs efficiency and provides higher error correction capability.

We evaluate ROBIN using gem5 cycle-accurate simulator [12] and compare it with per-word ECC and interleaved ECC. The evaluations show that the error rate in per-word and interleaved ECCs is higher than the error rate in optimal ECC by 151.7% and 42.3%, respectively, which is reduced to 5.3% by ROBIN (equivalent to 28.6x and 8.0x improvements, respectively). These significant improvements are achieved without increasing the overheads and complexity of ECCs.

The rest of this paper is organized as follows. Section II describes the preliminaries of STT-MRAM memory and write failure. In Section III, the motivation and the proposed method is presented. In Section IV, the simulation framework and the results are given. Finally, we conclude the paper in Section V.

II. STT-MRAM Basics

STT-MRAM memories consist of an access transistor and a *Magnetic Tunnel Junction* (MTJ), which has three layers: *Reference layer*, which its magnetization direction is fix, *free layer*, which its magnetization direction can be changed and determines the stored data, and the middle layer is *oxide barrier layer*, which separates these two ferromagnetic layers. The magnetization of free layer due to spin-polarized current flow can be in the same or opposite direction as reference layer spin direction. This parallelism and anti-parallelism of two layers generates low and high resistance in MTJ that is interpreted as logic value '0' and '1' in the cell, respectively [13].

A data is written into a STT-MRAM cell by applying a write current (I_{write}) for a predetermined pulse width. Based on its direction, this current flow changes the magnetic field direction of the free layer and generates high or low resistance in the MTJ. Changing this direction from parallel to anti-parallel (or vice-versa) is a stochastic process. This means that by applying write current, it is probable that the cell remains unflipped. This unsuccessful write operation is named *write failure* [7, 9, 14]. The probability of a write failure in a STT-MRAM cell is according to (1):

$$P_{Write-Failure} = 1 - P_{write} = exp(-t_{write} \times \frac{2 \times \mu_\beta \times p \times (I_{write} - I_{C_0})}{c + \log_e(\pi^2 \times \Delta/4) \times (e \times m \times (1 + p^2))}) \quad (1)$$

where, P_{write} is the probability of cell transition, I_{write} is write current, c is Euler constant, e is electron charge, m is magnetic momentum of the free layer, p is tunneling spin polarization, μ_β is Bohr magneton, and t_{write} is write pulse width.

To increase the probability of bit transition and reduce the write failure rate, previous studies increase the amplitude and/or width of write pulse [15, 16, 17]. In addition to imposing high latency and energy overhead, these techniques increase the probability of oxide barrier breakdown in STT-MRAM cells. Another method is verifying each write operation by reading and comparing the written cell with incoming value and repeating the write operation on a mismatch [18]. Extra read operations increase the occurrence probability of read disturbance in this technique and higher dynamic energy is imposed due to extra write operations. Some studies try to reduce the number of bit transitions in write operations by encoding the incoming data or writing into a cache block with the minimum hamming distance [19, 20]. These techniques complicate the design and increase the read disturbance rate.

The most conventional and widely-used scheme to overcome write failures is to employ ECCs in cache blocks [8, 7, 9, 10, 11]. ECC have been used in SRAM caches for decades and are the predominated protection scheme in modern processors. This scheme is inherited by STT-MRAM caches and several studies utilized ECCs for overcoming write failure. For an ECC-protected STT-MRAM cache block, a write operation is successful as long as the number of erroneous cells due to write failure is not larger than the number of correctable errors.

III. Motivation and Proposed Method

An ECC-protected LLC block is conventionally partitioned into M codewords each consist of a k-bit logical dataword and an r-bit ECC generated by that dataword capable of correcting up to t bits errors. A codeword is written correctly if the number of write failures in its $k+r$ bits is less than or equal to t and the write operation of a cache block is successful *if and only if* all its codewords are written correctly. In this section, we formulate the probability of correct write operation for STT-MRAM caches protected by the existing ECC schemes and demonstrate their shortcomings in correcting write failure, based on our observations and investigations.

A. Problem Formulation

In today's processors, a cache block conventionally consists of eight 64-bit dataword each protected by an 8-bit *Single Error Correction-Double Error Detection* (SEC-DED (72, 64)) code. For the sake of simplicity, we limit our discussion to the mentioned structure hereafter. However, the formulations, discussions, configurations, and proposed method are generally valid and applicable and not limited to this structure.

There are two configurations for generating eight SEC-DED(72, 64) codes in a cache block with 512 bits data. In

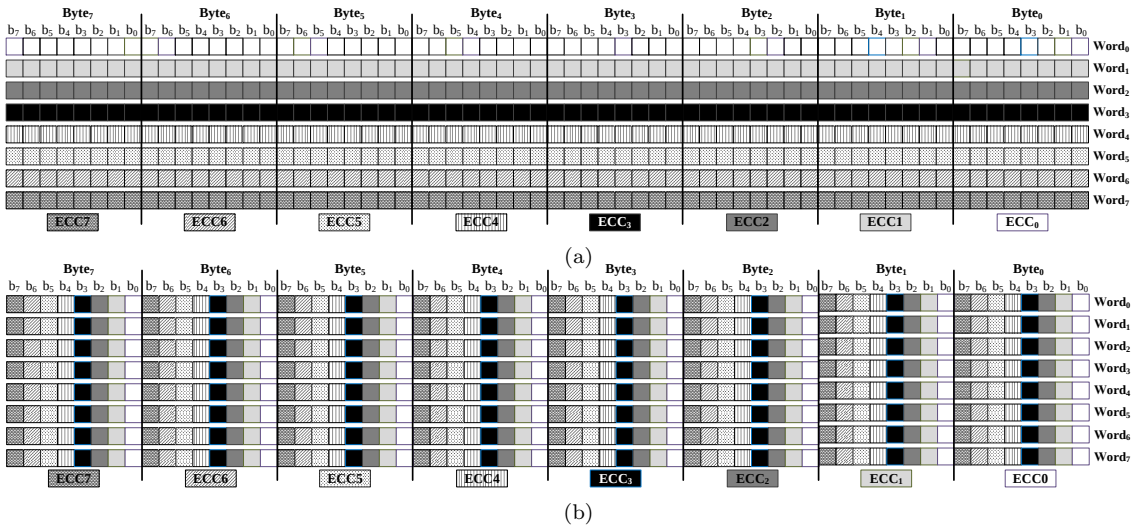

Fig. 1. ECC configuration schematic: (a) per-word configuration and (b) interleaved configuration.

the first configuration, known as *per-word ECC*, eight consecutive data bytes are grouped to generate a SEC-DED code. In the second configuration, known as *interleaved ECC*, bits with similar position in all 64 bytes contribute in generating each SEC-DED code. Interleaved ECC is capable of correcting *Multi-Bit Upsets* (MBUs) in SRAM caches by preventing the adjacent data bits to be grouped in the same logical dataword. The configuration of per-word and interleaved SEC-DED(72, 64) is illustrated in Fig. 1(a) and Fig. 1(b), respectively.

Both per-word and interleaved ECCs are applicable to STT-MRAM LLCs and none of the recent studies has differentiated between these two configurations in term of error correction capability. The advantage of interleaved ECC over per-word ECC in SRAM LLCs for correcting MBUs is not valid for STT-MRAM LLCs, because write failure in the cells is independent of the cells adjacency. A codeword is written correctly if write failure occurs in *none* of its bits or *only* in a single bit. Assuming 64-bit dataword protected by an 8-bit SEC-DED, the probability of a correct write operation is according to (2):

$$P_{word} = p_{write}^{k} + \binom{k}{1} \times p_{write}^{(k-1)}(1 - p_{write}) \qquad (2)$$

where, P_{write} is the probability of successful transition of single bit (calculated in (1)) and k is the number of bits in the codeword that must be flipped. All codewords must be written correctly in writing a cache block. The probability of a successful write operation for a block is according to (3):

$$\begin{aligned} P_{block} &= P_{w0} \times P_{w1} \times ... \times P_{w7} \quad = \prod_{i=1}^{7} P_{wi} \\ &= \prod_{i=1}^{7} [P_{write}^{k_i} + \binom{k_i}{1} \times P_{write}^{(k_i-1)}(1 - P_{write})] \end{aligned} \qquad (3)$$

where, w_i is codeword i in the block and k_i is the number of transitions in w_i.

P_{block} depends on the total number of transitions required for the target block as well as the distribution of these transitions between the codewords. The higher number of transitions and/or the larger variation in distribution of transitions, the lower probability of successful write operation is experienced. While the former depends on the data content and applications behavior, the latter depends on the partitioning the data bits between codewords, which is directly determined by the ECC configuration.

TABLE I
CONFIGURATION OF ON-CHIP CACHES

L1 Cache	**32KB, 4-way associative, 64B block size, write-back, SRAM**
L2 Cache	**1MB, 8-way associative, 64B block size, write-back, STT-MRAM**

When the total number of required transitions in a block ($K = \sum_{i=1}^{7} k_i$) is fixed, it can be easily proven that the maximum value of P_{block} in (3), which is the multiplication of probability of successful write of all codewords, is obtained when all $k_i s$ are the same and equal to $K/8$. Therefore, P_{block} increases by more uniform distribution of total transitions between the codewords and degrades by larger variance of that distribution.

B. Observation and Motivation

To investigate the capability of per-word and interleaved ECC configurations in evenly distributing the block transitions between their codewords, we conduct a set of simulations on SPEC CPU2006 benchmark suite using gem5 cycle-accurate simulator [12]. The details of simulation is described in Table I. Fig. 2 shows the total number of transitions for write operations in all bit positions of cache blocks during the workload execution. Considering the transitions of *bwaves* workload in Fig. 1(a), the number of transitions in 20% of upper part of each 64-bit word is by 4x larger than of lower part of each word. On the other hand, the transition pattern in all eight 64-bit words are almost similar. For workloads with this write pattern, which is the case for *floating-point* workloads, the distribution of transitions in per-word ECC can be more uniform than that in interleaved ECC.

Fig. 2(b) depicts another write pattern in which similar behavior is observed for every 32-bit data. In this pattern,

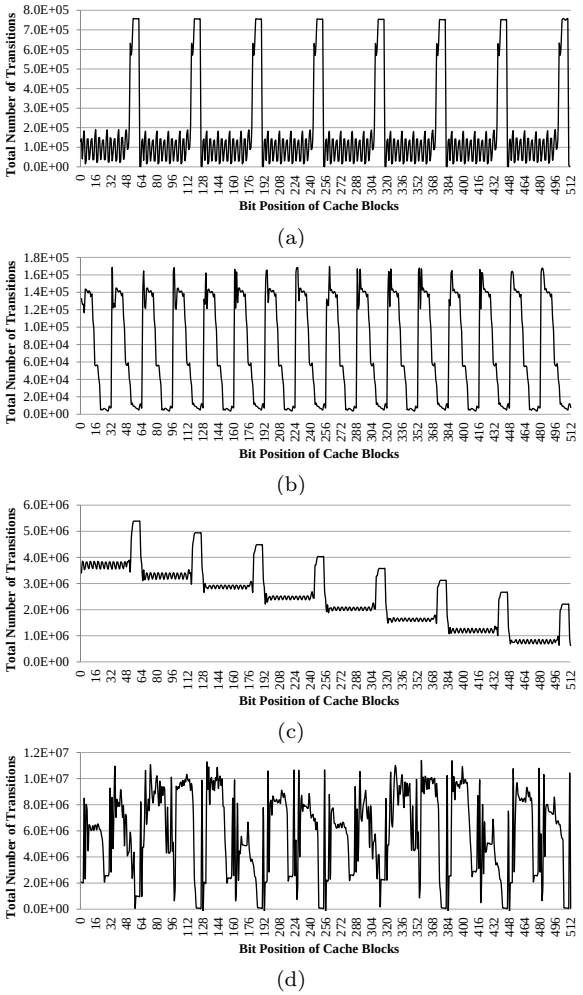

Fig. 2. Number of transitions in all bit positions of cache blocks: (a) bwaves, (b) calculix, (c) cactusADM, and (d) mcf workloads.

which corresponds to *calculix* as an *integer* workload, the number of transitions sharply drops in such a way that the low part of data is more active than the upper part by 14x. The write pattern of each 64-bit data in Fig. 2(c) is almost similar to that of Fig. 2(a), while the number of transitions in different words is largely different. In this workload, the probability of existing a valid data in a block decreases by increasing the position number of words and a fraction of the block containing invalid data. Considering *mcf* workload, an irregular behavior is observed in the write pattern depicted in Fig. 2(d). In this pattern, the number of transitions is almost randomly distributed. Even in this random pattern, a sharp drop can be observed is in bit positions near 32, 64, 96, and so on.

The capability of both per-word and interleaved ECC configuration is not easy to predict and strongly depends on per write pattern. An optimal ECC configuration, which provides the maximum probability of correct write operation, is the one that can evenly distribute the transitions between codewords in all write patterns. To demonstrate how the non-uniformity in transition distribution degrades the cache reliability, we calculate the cache error rate for per-word and interleaved configurations and compare the results with error rate of optimal configuration. Fig. 3 depicts the normalized error rate for three mentioned ECC configurations. The results show that the error rate in per-word and interleaved configurations is higher than that of optimal by 151.7% and 42.3%, respectively. This value for per-word ECC is as high as 544.0% in *cactusADM* and as low as 10.5% in *astar* workload. The worst-case error rate for interleaved ECC is 225.1%, which is observed in $h264ref$ workload.

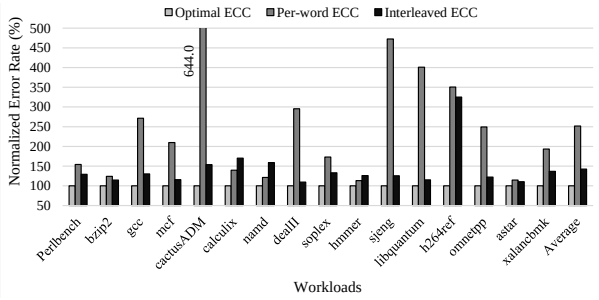

Fig. 3. Cache error rate in per-word and interleaved ECC normalized optimal ECC.

C. Proposed Method

Write patterns vary in different applications and even in different write accesses per application. We have already observed that this variation leads to non-uniform distribution of bit transitions and inefficiency of both per-word and interleaved ECCs in correcting errors. An optimal ECC, which evenly distribute the total number of transitions between codewords in all write accesses minimizes the cache error rate and provides the maximum reliability.

As observed in Fig. 3, for the majority of workloads, the error rate in per-word and interleaved configurations is significantly higher than that of optimal configuration. An efficient ECC should provide a near-optimal error rate for all workloads and all write accesses within a workload. Our proposed ECC configuration, so-called *Incremental Oblique Interleaved* (ROBIN) ECC, generates the codewords in such a way that variation in workloads behaviors has minimum impact on the ECC efficiency and its error rate is not more than 10% higher than that of the optimal ECC.

Prior to explain the architectural details of ROBIN, here we elaborate how various workload behaviors cause different transition distributions. This discussion clarifies the key idea behind ROBIN and highlights its capability in uniformly distributing the transitions in various write patterns. The transition patterns of write operations are data-dependent and beside the content, different data types cause different patterns. As an example, in *floating-point* applications, the number of transitions in exponent part is significantly larger than that in the mantissa part. Our observations in Fig. 2(a) and Fig. 2(c) for *bwaves* and *cactusADM* workloads confirm this intuition.

Considering *integer* applications, cache blocks contain multiple 32-bit data words in which the transition probability in lower order bits is larger than that in higher order bits. Majority of integer data are *narrow-width val-*

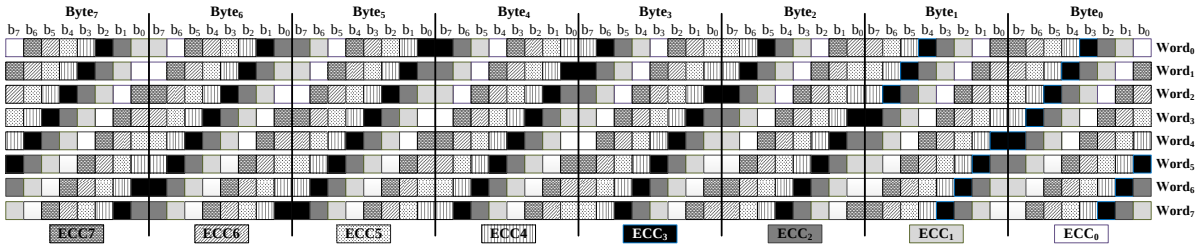

Fig. 4. ECC configuration schematic for ROBIN ECC.

ues [21], which confirm our intuition on larger number of transitions in lower order bits. We have already observed this behavior in Fig. 2(b) for *calculix* workload. Applications with large fraction of string data type have their unique write pattern, which strongly depends on the character coding standards. Besides the type of data, the validity of cache block words affects the transition pattern. There can be blocks in which all their words contain valid data, e.g., *bwaves* workload in Fig. 2(a), and blocks with several invalid words, e.g., some written blocks for *cactusADM* as shown in Fig. 2(c).

An efficient ECC configuration should be capable of evenly distributing the total transitions between all codewords. To this aim, ROBIN partitions the data bits in such a way that all 64-bit words in a block as well as all eight bytes of these words contribute in generating all ECC words. In addition, all bit positions of all bytes within a word contribute similarly in each codeword.

Fig. 4 illustrates the ROBIN policy in constructing codewords and generating ECCs. Considering a 512-bit cache block consisting eight 64-bit data words, eight bits from each data word contribute in generating each ECC. From another aspect, all 64 bytes contribute in generating all ECCs (one bit from each byte for each ECC). In addition, eight bit positions in each eight bytes of the word are similarly distributed between the ECCs. For $byte_0$ in the words, all bit positions b_0, b_1, ..., b_7 each selected from $byte_0$ of a unique word are in the same group. To balance the variation between bit positions of different bytes inside a word, a unique bit position from each byte of a word is selected for generating each ECC. Mathematically speaking, each ECC word ECC_n (n $\in \{0, ,1, ...,7\}$) is a function of 64 data bits selected according to (4):

$$ECC_n = f(\{b_{i,j,(i+j+n)} |\ \forall i, j\{0, 1, ..., 7\}\}) \qquad (4)$$

where, $b_{i,j,(i+j+n)}$ is a data bit in position $(i+j+n)mod(8)$ in $byte_j$ of $word_i$. Using this policy in grouping data bits to generate ECCs, ROBIN is capable of distributing transition variations between data words, between bytes of each word, and between bit positions of each byte. For the sake of clarity, we focused on 512-bit cache blocks protected by eight 8-bit SEC-DED codes. However, ROBIN configuration is generally independent of data and ECC size as well as coding scheme. It is scalable in term of block size and is also applicable to other ECCs. This method has no effect on the complexity of ECC encoder/decoder logic and its overhead is almost the same as those of interleaved ECC.

IV. Simulation Setup and Results

To evaluate the efficiency of the proposed ROBIN ECC, we implement it in gem5 cycle-accurate simulator [12] and use SPEC CPU2006 benchmark suite as workloads. The first 100 million instructions are skipped as warm-up phase and the results are extracted for the next one billion instructions. The details of simulation was shown in Table I. ROBIN is compared with per-word and interleaved ECC in terms of transitions distribution and error rates. The results are normalized to optimal ECC.

Fig. 5 depicts the variation in the total number of transitions in cache blocks. For all write accesses in a workload, we sort the number of transitions in eight codewords in a block and the summation of these values are normalized to the average of transitions in codewords (which is the number of transitions in optimal ECC). In Fig. 5, the number of transitions in codewords with minimum and maximum transitions is depicted for per-word, interleaved, and ROBIN configurations. Normalizing to the number of transitions in a codeword for uniform distribution (optimal), the results show that the number of transitions in codewords of per-word ECC is from 41.1% to 208.8%, on average. This interval is smaller for interleaved ECC and the minimum and maximum transitions in codewords are 43.6% and 169.2%, respectively. ROBIN significantly reduces this gap by limiting the number of transitions in codewords between 73.7% to 128.4%, on average. The interesting achievement is that in the worst case, the minimum and maximum transitions of codeword in ROBIN is larger than 61.8% and lower than 146.1%, respectively. These values for per-word ECC are 7.4% and 390.4%, respectively, and for interleaved ECC are 17.8% and 222.2%, respectively. The wider gap between the number of transitions in codewords indicates larger non-uniformity, which result in higher cache error rate compared to uniform distribution (optimal ECC).

Fig. 6 depicts the cache error rate in three evaluated ECC configurations. The results illustrate the increase in cache error rate due to non-uniformity of transitions distributions compared to the optimal ECC. On average, per-word and interleaved ECC increases the error rate by 151.7% and 42.3%, respectively. This value is reduced to

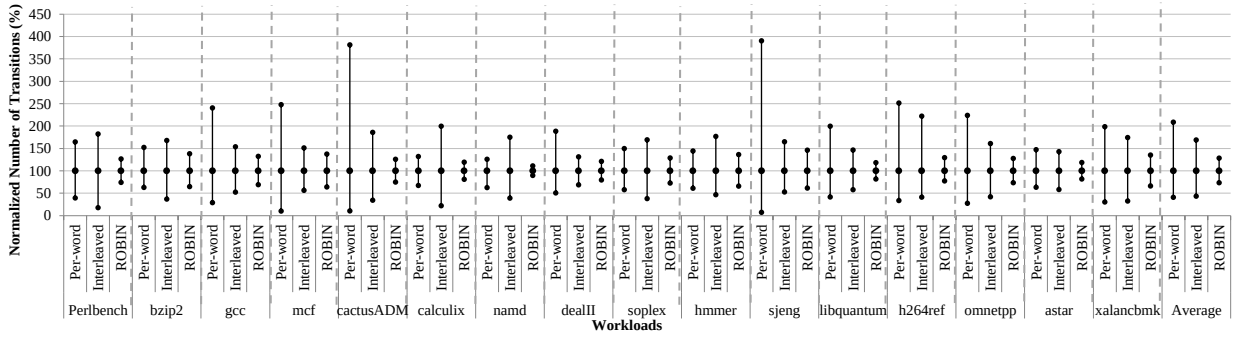

Fig. 5. Variation in number of transitions between codewords of cache blocks (minimum, average, and maximum).

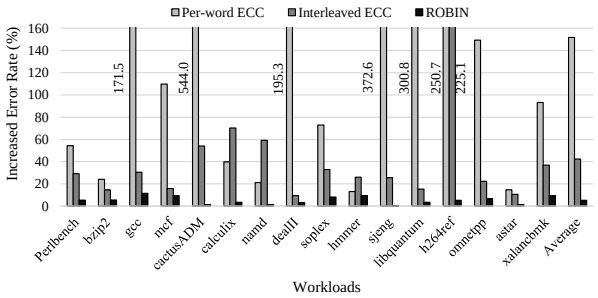

Fig. 6. Cache error rate increase for per-word, interleaved, and ROBIN ECC configurations normalized to the optimal ECC.

5.3% in ROBIN. Therefore, ROBIN reduces the error rate for per-word and interleaved ECC due to non-uniformity of transitions by 28.6x and 8.0x, respectively. For some workloads, error rate in per-word increases by even more than 300%, e.g., *cactusADM*, *sjeng*, and *libquantum*. In the worst-case, interleaved ECC increases the error rate by 225.1% in $h264ref$. The error rate increase in ROBIN is less than 10% for all workloads. This value is even less than 1% for some workloads, e.g., *cactusADM*, *namd*, *sjeng*, and *astar*, which is almost the same as optimal ECC.

It is worth noting that ROBIN provides this near-optimal reliability with the same cost as per-word and interleaved ECC. Its area and energy consumption is similar to those of conventional ECC configurations. In addition, it has no impact on a cache access time compared to existing configurations.

V. Summary and Conclusions

Write failure is the main reliability challenge in emerging STT-MRAM caches. Data dependency and variations in different codewords of a block significantly degrade the efficiency of conventional ECCs. This paper 1) investigated the commonly used per-word and interleaved ECCs and revealed that these schemes increase the cache error rate by 151.7% and 42.3%, respectively, and 2) proposed near-optimal ROBIN ECC to reduce the cache error rate to as low as 5.3%. This significant improvement in cache reliability is achieved with no increase in ECC overheads. ROBIN is a promising alternative for conventional per-word and interleaved configurations to maximize the efficiency of ECCs in emerging STT-MRAM caches.

References

[1] A. Chen, "A review of emerging non-volatile memory (NVM) technologies and applications," *Elsevier Solid-State Electron.*, vol. 125, pp. 25–38, 2016.

[2] E. I. Vatajelu *et al.*, "Challenges and solutions in emerging memory testing," *IEEE Trans. Emerging Topics Comput.*, 2017, in press.

[3] J. Choi and G.-H. Park, "Nvm way allocation scheme to reduce nvm writes for hybrid cache architecture in chip-multiprocessors," *IEEE Trans. Paral. Dist. Syst.*, vol. 28, no. 10, pp. 2896–2910, 2017.

[4] S. Mittal, "A survey of soft-error mitigation techniques for non-volatile memories," *Computers*, vol. 6, no. 1, p. 8, 2017.

[5] H. Naeimi *et al.*, "STT-MRAM scaling and retention failure," *Intel Tech. J.*, vol. 17, no. 1, pp. 54–75, 2013.

[6] A. Chintaluri *et al.*, "Analysis of defects and variations in embedded spin transfer torque (stt) mram arrays," *IEEE J. Emerg. Sel. Top. Circ. Syst.*, vol. 6, no. 3, pp. 319–329, 2016.

[7] Z. Azad *et al.*, "An efficient protection technique for last level STT-RAM caches in multi-core processors," *IEEE Trans. Paral. Dist. Syst.*, vol. 28, no. 6, pp. 1564–1577, 2017.

[8] H. Farbeh *et al.*, "Floating-ecc: Dynamic repositioning of error correcting code bits for extending the lifetime of stt-rams," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3661–3675, 2016.

[9] Z. Azad *et al.*, "AWARE: adaptive way allocation for reconfigurable ECCs to protect write errors in STT-RAM caches," *IEEE Trans. Emerging Topics Comput.*, 2017, in press.

[10] X. Guo *et al.*, "Sanitizer: Mitigating the impact of expensive ecc checks on stt-mram based main memories," *IEEE Trans. Comput.*, 2017, in press.

[11] W. Wen *et al.*, "Cd-ecc: Content-dependent error correction codes for combating asymmetric nonvolatile memory operation errors," in *Proc. Int. Conf. Comput.-Aided Des.*, 2013.

[12] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Comput. Arch. News*, vol. 39, no. 2, pp. 1–7, 2011.

[13] S. Asadi *et al.*, "Wipe: Wearout informed pattern elimination to improve the endurance of nvm-based caches," in *Proc. IEEE Asia South Pacific Des. Automat. Conf.*, 2017.

[14] E. Eken *et al.*, "A novel self-reference technique for stt-ram read and write reliability enhancement," *IEEE Trans. Mag.*, vol. 50, no. 11, pp. 1–4, 2014.

[15] H. Sun *et al.*, "Architectural exploration to enable sufficient mtj device write margin for stt-ram based cache," *IEEE Trans. Mag.*, vol. 48, no. 8, pp. 2346–2351, 2012.

[16] Y. Emre *et al.*, "Enhancing the reliability of stt-ram through circuit and system level techniques," in *Proc. IEEE Workshop Signal Proc. Syst.*, 2012.

[17] Y. Lakys *et al.*, "Self-enabled error-free switching circuit for spin transfer torque mram and logic," *IEEE Trans. Mag.*, vol. 48, no. 9, pp. 2403–2406, 2012.

[18] C. Yang, "Improving reliability of non-volatile memory technologies through circuit level techniques and error control coding," *EURASIP J.Adv. Sig. Proc.*, pp. 1–24, 2012.

[19] W. Wen, *et al.*, "CD-ECC: Content-Dependent Error Correction Codes for combating asymmetric nonvolatile memory operation errors," in *Proc. Int. Conf. Comput.-Aided Des.*, 2013.

[20] A. M. H. Monazzah, H. Farbeh, and S. G. Miremadi, "Ler: Least error rate replacement algorithm for emerging STT-RAM caches," *IEEE Trans. Device Mater. Rel.*, vol. 16, no. 2, pp. 220–226, 2016.

[21] M. Imani, A. Rahimi, Y. Kim, and T. Rosing, "A low-power hybrid magnetic cache architecture exploiting narrow-width values," in *Proc IEEE Non-Vol. Mem. Syst. App. Symp.*, 2016.