Compression with Multi-ECC: Enhanced Error Resiliency for Magnetic Memories

*Irina Alam, *Saptadeep Pal and Puneet Gupta
Department of Electrical and Computer Engineering, University of California, Los Angeles,
{irina1, saptadeep and puneetg}@ucla.edu

***Abstract*—Emerging non-volatile magnetic memories such as the spin-torque-transfer random access memories (STT-RAMs) provide superior density and energy benefits compared to conventional DRAM or Flash based memories. However, these technologies often suffer from reliability issues and thus strong conventional reliability schemes are required. These schemes have large overhead of storage which in turn can potentially eclipse the density and energy benefits these technologies promise. Moreover, the read and write operations in STT-RAMs show asymmetric behaviour i.e., bit-flip probability of 1→0 is significantly higher than 0→1. However, conventional ECC schemes treat both 0 and 1 flips similarly and thus results in unbalanced reliability of these two types of errors. In this work, we propose a new ECC protection scheme for STT-RAM based main memories, compression with multi-ECC (CME). First we try to compress every cache line to reduce the size of the cache line and then based on the amount of compression possible, we use the saved additional bits to increase the protection using stronger ECC codes if possible. Compression itself reduces the hamming weight of the cache lines, thus reducing the probability of 1→0 bit-flips. Opportunistically using stronger ECC codes further helps tolerate multiple bit-flips in a cache line. Our results show that for STT-RAM based main memories, CME can reduce the block failure probability by up to 81.6% (average 50.7%) and 78.4% (average 51.2%) over using a (72,64)SECDED for each cache line word, when maximum of 4 reads and 2 reads respectively are allowed to a cache line before a write-back/restore operation is done.**

***Index Terms*—STT-RAM, Cache line Compression, Multi-ECC**

I. Introduction

Emerging non-volatile memory (NVM) technologies are being considered as potential replacements for DRAM and Flash technologies, both of which are nearing their scaling limits. Most of these new non-volatile technologies (Phase Change Memory[PCM], STT-RAM, Resistive RAM[ReRAM], etc.) promise better scaling, higher density and reduced cost-per-bit [1]. However, they come with their own set of challenges. The biggest problem that these emerging technologies face is the high stochastic bit error rate. In fact, the reliability challenges of NVMs can offset the density and energy advantages that they offer. Increase in demand for memory capacity requires aggressive scaling of area-per-bit of storage. At higher density, these non-volatile emerging memory technologies tend to be more susceptible to stochastic bit errors [2]. Due to the random nature of the bit errors, these memory technologies require stronger in-field error-correcting codes (ECC) [3].

The stochastic nature of failures in NVMs is similar to the radiation induced soft errors in DRAM and SRAM and occur without any warning. In order to ensure integrity of the data, error detection mechanism, followed by correction of the error(s) needs to be incorporated in a system. In conventional systems, ECC schemes are deployed to recover from memory errors which requires adding redundancy information alongside the original data (or message). For DRAM based memory, the most commonly used ECC schemes to recover from bit error or faulty chip error are SECDED (Single-Error Correcting, Double-Error Detecting) scheme [4] and Chipkill-Correct scheme [5].

In the emerging NVM technologies, the stochastic bit error rate, however, is much higher than the single-bit soft error rate in DRAM. For example, in PCM, a two-bit cell may have a 10^6-times higher error rate than DRAM and require much stronger ECC [3], [6]. As a result, the conventional ECC schemes used in DRAM based memory need to be extended for these new memory technologies to provide multi-bit protection to maintain acceptable limits of yield and performance. However, the cost and complexity of stronger error detection and correction circuitry increases exponentially. Also, stronger ECC requires larger number of ECC bits. This has overhead not just in terms of storage but also power and performance.

Out of the various magnetic NVMs that have been proposed, Spin-Transfer Torque Random Access Memory (STT-RAM) is one of the most promising non-volatile technologies and has already been introduced in to commercial products [7], [8]. However, it also suffers from a very high Bit Error Rate (BER) [9], [10]. As the NVM technology scales to below 45nm, read disturbance error, retention error due to thermal instability alongside write error rates are growing, leading to unacceptably high bit error rates (BER). Several circuit level and bit-cell design solutions have been proposed to lower the error rates [11]. Also, a few recent efforts have been made to either reduce the error rate or to provide stronger error resiliency with the least possible overhead [3], [12], [13]. Most of these solutions however result in very high power and area overhead.

In this paper, we propose CME (Compression with Multi-ECC), a novel scheme to provide strong error correction in Magnetic RAM (MRAMs) based main memory subsystems. Though the proposed technique would be useful for many different types of MRAMs, for our evaluations, we consider the characteristics (error-rates) of STT-RAM. We use a slightly modified version of the compression scheme suggested in [14] to compress each cache line. Once compressed, the available bits are opportunistically utilized to provide strong error protection. In case the cache line can be compressed to a size of less than half, the

*I. Alam and S. Pal contributed equally to this work

cache line is replicated, so that consecutive reads from the same cache line can be done from different copies of the compressed data. This helps reduce errors due to read disturbance errors, which is a major source of error. Compression also allows employing stronger protection without incurring additional storage overheads of redundancy required for stronger ECC schemes.

The rest of the paper is organized as follows. Section II provides a brief background of STT-RAMs, their fault models, previous work on STT-RAM reliability and cache line compression. Section III then details our proposed scheme. Section IV highlights our experimental methodology and Section V includes the results and a brief discussion on the overhead of our scheme. Section VI finally concludes our work.

II. Background

A. STT-RAM Basics

In an STT-RAM cell, data is stored in a magnetic tunneling junction (MTJ). As current is passed through a mono-domain ferromagnet, the angular momentum of the electrons flips the direction of magnetization in the ferromagnet. The basic structure of a STT-RAM cell is given in Figure 1. The MTJ consists of a tunneling oxide (MgO) separating two ferromagnetic layers. One layer (reference layer) has fixed magnetization and the other is a free layer whose direction of magnetization flips depending on the direction of current of sufficient density. The relative alignment of the two layers results either in a high resistance path (when opposite and usually represents 1) or a low resistance path (when parallel and usually represents 0).

Errors in STT-RAM can be broadly classified under three categories: read errors, write errors and retention errors due to thermal instability.

1) Read Errors: The read operation in STT-RAMs is unidirectional. As technology scales below 45nm, read current doesn't reduce beyond 20μA while the write current reduces to around 30μA [10]. Thus, read current is getting closer to the write current such that the read operation now has the potential to alter the stored value. Such an error is called read disturbance error. The data that is read is correct but the stored value becomes erroneous and subsequent reads from this location may contain multiple bit-flips. Since the read current is unidirectional, the unintentional bit flip during read is asymmetric and happens only in one direction (1$\rightarrow$0 when reading a '1'). Thus, reducing the number of 1's (or Hamming distance) in a cache line will help to reduce the read disturbance errors (RDEs) considerably.

2) Write Errors: In STT-RAM, a write failure happens if the switching current is removed before the MTJ switching completes. The time required for flipping the cell content varies due to the stochastic switching characteristics of the MTJ. However, this failure is also asymmetric. Since, 0$\rightarrow$1 flipping requires larger switching current than 1$\rightarrow$0 flipping due to the lower spin-transfer efficiency, the chances of write error happening are much higher during a 0$\rightarrow$1 flip than a 1$\rightarrow$0 transition. As mentioned in [12], the bit error rate of 0$\rightarrow$1 flipping is $P_{ER,0\rightarrow1} \sim= 5\times10^{-3}$ while that of 1$\rightarrow$0 flipping is $P_{ER,1\rightarrow0} \sim= 10^{-7}$. They have also analyzed and concluded that the reliability of a word in a cache line

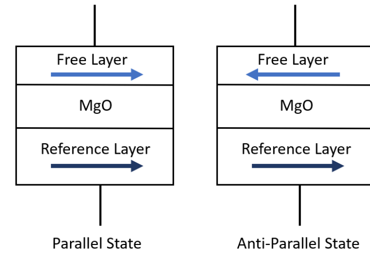

Fig. 1: Schematic of STT-RAM showing the anti-parallel and parallel states

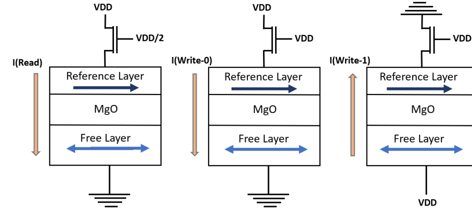

Fig. 2: Read and write mechanisms for STT-RAM is shown here

decreases exponentially with increase in Hamming Weight. Thus, just like RDEs, Write Error Rate (WER) can also be reduced by reducing the Hamming Weight of a cache line.

3) Retention Errors: In STT-RAM, the third major source of error is retention error where the data stored in the STT-RAM cell flips after a certain period of time. This false switching of data during standby state is due to the inherent thermal instability of STT-RAMs. The critical current or the write current is proportional to the thermal stability of the cell. Higher thermal stability requires a higher write current and/or a longer write pulse. Thus there is a fundamental trade-off between write-ability (write time and/or power) and retention time.

B. Previous Work On STT-RAM Reliability

To reduce errors due to read disturbance, restore operation can be used which writes back the data every time there is a read operation [10]. However, restore operations have a huge overhead in terms of latency, energy and complexity. One recent work [15] suggests the use of data compression to enable replication of bits in the memory. If cache lines are replicated, then a restore operation would be needed only after all the copies have been read. This can potentially decrease the number of restore operations required after every read to deal with read error disturbances.

To deal with write errors, [12] suggests reducing the Hamming weight of each cache line. If the number of 1's is reduced in each line, it would considerably reduce the probability of having write errors since a 0$\rightarrow$1 flip requires longer time and larger current and is thus, more prone to write errors. To reduce the Hamming weight of the cache line, [12] suggests using static/dynamic XOR between words of each cache line exploiting the value locality of stored data. However, few recent works [3], [16] suggest improvements at the circuit level to improve BER of magnetic memories. But most of these techniques either target mitigation from one type of error (write or read error) or have very large overheads in terms of circuit complexity, area or power.

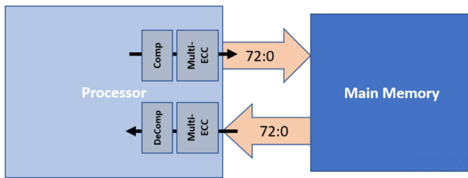

Fig. 3: Processor Memory system architecture with CME

C. Previous Work On Cache Compression

Cache line compression techniques are being widely proposed to satisfy the rising demand for memory storage capacity and memory bandwidth [17], [18], [19]. These techniques exploit spatial and value locality of the data in typical applications. Common usage of arrays and typical data-structures results in regular data access patterns, while value locality helps in representing an entire cache line using small magnitude delta values. Moreover, often values used in a program are low magnitude, however, these values get represented inefficiently, for e.g., 4-byte integer type used to represent values that usually need only 1-byte.

The Base-Delta-Immediate (BΔI) compression scheme, as proposed in [20], is frequently used. As the name suggests, the main idea behind this compression scheme is to represent the cache line in a compact form using a common base value plus an array of relative differences (deltas), whose combined size is much smaller than the original cache line. The base could be a single value or there can be multiple bases. Another very recent work on cache compression [14] claims to have better compression ratio than BΔI. The details of this compression scheme (Bit-Plane-Compression) is provided in Section III since we have used roughly the same scheme in our work.

III. Our Scheme - Compression with Multi-ECC (CME)

In this section, we will discuss the details of Compression with Multi-ECC scheme. Cache line compression is used for two reasons. Firstly, it helps in reducing the Hamming weight of each cache line. Secondly, it enables either data replication (when the compressed cache line is less than half its uncompressed size) or allows to use the available bits to provide stronger error protection. The selection of the stronger ECC scheme depends on the final size of the compressed cache line such that the overall size of each cache line with the redundant bits remains uniform across all cache lines.

A. Overall Architecture

As shown in Figure 3, every time a cache line is to be stored in the memory, it is a two-step approach. It first goes through the compression engine and then through the Multi-ECC encoder. In case of a load operation, it first goes through the ECC Decoder and then the de-compression engine. As mentioned before in Section II-C, we used Bit-Plane Compression (BPC) scheme proposed in [14] and is explained in detail in the following subsection. In case of minimal or no compression, BPC scheme might result in an increase in the original cache line size. Hence, once compression is done, it is checked if the size of the compressed cache line is less than the original size of 512 bits. If not, then the raw unmodified cache line is used and a (72,64)SECDED code is applied on each 64-bit block. However, if compression reduces the size of the cache line, then based on the final size of the cache line, we opportunistically add stronger ECC/protection scheme.

B. Cache Line Compression Scheme

Bit Plane Compression (BPC) as described in [14] is a two-step process. The first step is cache line manipulation and transformation (Delta-BitPlane-XOR [DBX]) to improve compressibility of data and thus reduce the compression hardware complexity. DBX itself is a three step process as shown in Figure 4. The first step is similar to Base-Delta-Immediate where the first 32-bit word of each cache line is kept intact and the subsequent words go through pairwise delta operations. The result of each delta operation is a 33-bit word (1 additional sign bit for subtraction). Thus, the overall cache line size now increases from 512 bits (32-bits x 16 words) to 527 bits (32-bit base-word + 33-bits x 15 delta-outputs). The following step is a series of bit-plane (a set of bits corresponding to the same bit position within each cache line word) rotation operations that the newly formed 15 33-bit delta outputs go through. The base remains intact and the 15 delta words now rotate and form 33 15-bit DBP symbol. The final step in this data transformation is the XOR operation between neighboring DBP symbols. The first DBP symbol remains intact (acts as a second base) and the remaining 32 symbols transform into their respective DBX symbols.

The next step after data transformation is the compression of the transformed data. BPC combines run-length encoding (RLE) with a type of frequent pattern encoding (FPE) to compress the transformed data. The work in [14] used word-size of 64 bits in a 128-byte cache line, while for our evaluations we use 32-bit words and 64-byte cache line. Hence, our symbol encoding is slightly different from theirs and is shown in table I. The base (first original) symbol is compressed separately by original symbol encoder as {3b000}, {3b001, 4-bit data}, {3b010, 8-bit data}, or {3b011, 16-bit data} if its value is 0 or fits into 4/8/16-bit signed integer, respectively. Otherwise, the base symbol is encoded as {1‘b1, 32-bit data}.

TABLE I: Frequent Patterns for BPC and DBP/DBX symbol encoding

DBP/DBX Pattern	Length	Code (binary)
0 (run-length 2∼33)	7-bit	{2’b01, (RunLength-2)[4:0]}
0 (run-length 1)	3-bit	{3’b001}
All 1’s	5-bit	{5’b00000}
DBX!=0 and DBP=0	5-bit	{5’b00001}
Consecutive two 1’s	9-bit	{5’b00010, StartingOnePosition[3:0]}
Single 1’s	9-bit	{5’b00011, OnePosition[3:0]}
Uncompressed	16-bit	{1’b1, UncompressedData[14:0]}

C. Multi-ECC on Compressed Cache Line

Compression helps to reduce the size of the cache line in most cases. Once the reduction is done, the final size of the cache line determines the ECC scheme to be used as shown in Table II.

The main idea is to be able to break up the cache line into 8 words and restrict the size of each word (original word + ECC redundancy) in the cache line to 72 bits. This is to make

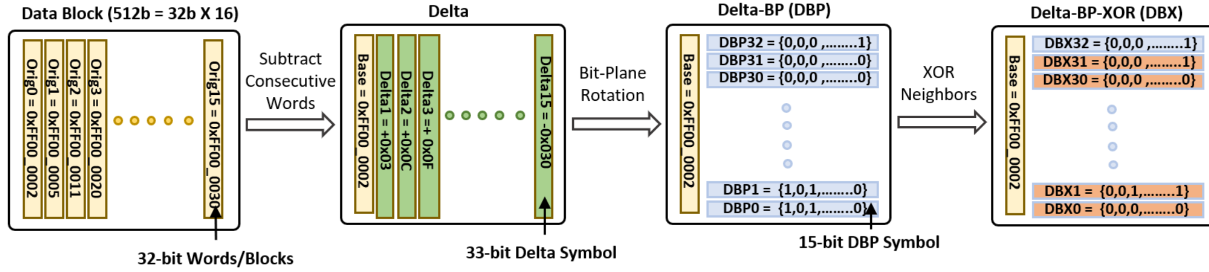

Fig. 4: An overview of the Bit-Plane Transformation scheme

TABLE II: ECC scheme to be used depending on the compressed cache line size

Length of compressed cache line (in bits)	ECC scheme to be used
>512	No compression (Use Raw Cache line + (72,64)SECDED)
≤512 and >464	(72,64)SECDED on each 64-bit cache line word
≤464 and >416	(72,58)DECTED on each 58-bit cache line word
≤416 and >256	(72,52)3EC4ED on each 52-bit cache line word
≤256 and >128	Replicate the cache line (2 copies) and (72,64)SECDED on each word
≤128	Replicate the cache line (4 copies) and (72,64)SECDED on each word

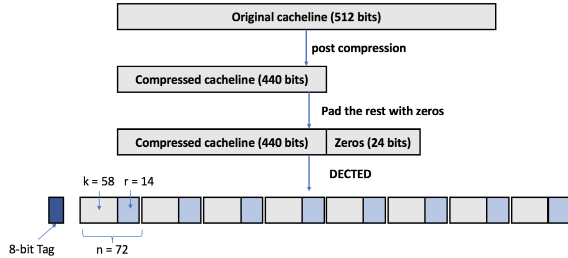

Fig. 5: An example of CME scheme where the compressed cache line size is 440 bits

sure that the STT-RAM based memory subsystem adheres to the standard DDR protocol. Figure 5 shows an example of a cache line of size 440 bits post compression. The best possible ECC scheme for this cache line is DECTED (Double Error Correction, Triple Error Detection). For DECTED, with codeword length n=72, minimum number of redundancy bits required is r=14. Therefore, every message or block needs to have a maximum length of k=58. This means that cache line, post compression, needs to be broken down into 8 words, each of size 58 bits. In order to be able to do so on a 440-bit cache line, it needs to be padded with 24 0s to increase its total size to 464 bits. After that, the cache line is divided into 8 equal sized words and each word is encoded using a DECTED encoder to a final codeword of length 72-bits.

It can be noted from Table II that, no stronger than 3EC4ED (3 error correcting, 4 error detecting) code has been used even for cases where stronger protection would be possible (for eg. 4EC5ED can be used in cases where the compressed cache line size is less than 368 bits). This is because stronger ECC not only adds greater hardware complexity and overheads, the redundant bits added to each word in the cache line often increases the Hamming weight of the overall word considerably, thus increasing chances of read disturb/write errors. As seen in Figure 6, with stronger ECC, the block

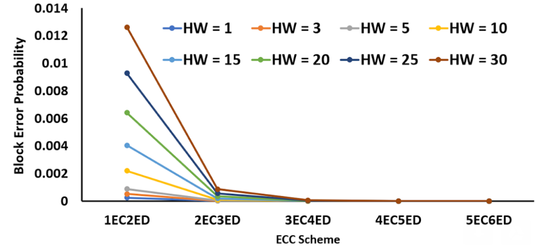

Fig. 6: Block error probability is shown for different Hamming weight (HW) blocks and ECC schemes. The probability of 1→0 bit-flip is considered to be 10^{-5}

error probability decreases rapidly till 3EC4ED, beyond which the benefit of stronger ECC saturates.

For cache lines that can be compressed to less than half the original size, we use the scheme presented in [15] where compressed cache line is replicated. The main benefit of replication is that it helps to provide protection against read disturbance errors. As mentioned in [15], the average number of read operation between two write operations for single and dual-core systems are 1.61 and 1.32 respectively. Clearly, between any 2 write operation, most cache blocks see less than 2 read operations. Thus, when the data is replicated once, the first copy is used during the first read operation and the second copy is used during the second read. If the first read operation causes an unwanted flip in the data, the error doesn't impact the second read operation as the replicated copy is used during the next read. Since the average number of read operations between any two writes is less than 2, replicating the data once is mostly good enough to prevent RDEs. Reduction in RDEs means restore operations are no longer needed after every read, which saves both time and energy. If the compressed size is less than $1/4^{th}$ of the original size, we propose a 4x replication for even stronger protection against RDEs.

D. Additional Tag Bits

Every cache line now needs additional tag bits to denote if the cache line is compressed and what protection scheme is used. As shown in Table III, we use 8 additional bits of tag to each cache line to denote the transformation operation that was done for that particular cache line.

- *Bit0:* Denotes if the stored cache line has been compressed or not. If compressed then the first bit of the tag is '1'; else '0'.
- *Bits1-3:* When the cache line is compressed, these three additional bits denote the ECC/replication scheme used

TABLE III: 8-bit Tag per Cache Line for CME

Tag Bits	When Compression is possible		When Raw Cache line is used
Bit-0	-	'1'	'0'
Bits1-3	BPC + (72,64)SECDED	'000'	'000'
	BPC + (72,58)DECTED	'001'	
	BPC + (72,58)DECTED	'001'	
	BPC + (72,52)3EC4ED	'010'	
	BPC + Replication (2copies) + (72,64)SECDED	'011'	
	BPC + Replication (4copies) + (72,64)SECDED	'100'	
Bits4-7	(8,4)SECDED redundancy for the first 4 Tag bits	-	'0000'

for that cache line as given in Table III, else the field is populated with '000'.

- *Bits4-7:* For a compressed cache line, these 4-bits are used to provide a SECDED protection on the first 4-bits of tag. In the case of uncompressed cache line, '0000' is used, which also happens to be the redundancy for SECDED encoding of an all-zero message.

Along with the tag bits, the memory controller keeps a track of the number of read operations per cache line in order to enable write-back or restore operations after a certain number of reads. This can be implemented using a similar mechanism proposed in [21] for selective DRAM refresh. Also for the replicated cache lines, the memory controller tracks the copy that was last read so that the next read is from the next copy.

IV. Evaluation Methodology

To evaluate our protection scheme for STT-RAMs, we first compiled a set of applications from the SPEC CPU2006 benchmark suite for 64-bit RISC-V (RV64G) instruction set v2.0 [22]. The list of applications from the benchmark suite have been listed in Table IV. These applications are a mix of integer and floating point benchmarks.

TABLE IV: Details about Evaluation Methodology

SPEC CPU2006 Benchmarks used	400.perlbench, 450.soplex, 473.astar, 401.bzip2, 403.gcc, 462.libquantum, 453.povray, 444.namd, 470.lbm, 447.dealII
Cache Line size	512-bits (64-Byte)
Write Error Rate (0→1) [11]	1×10^{-8}
Retention Error Rate/hour (0→1) [11]	1×10^{-5}
Read Disturb Rate [10]	1×10^{-5}

After compilation, the applications were run on Spike [23], a RISC-V simulator to extract the memory traces for both data and instructions. Next, each application was subjected to CME. The average block failure probability (average failure probability of each word in the cache line) was computed under a certain write, retention and read disturb error rates (given in Table IV) based on the final set of cache lines obtained after applying the CME scheme to the obtained memory traces.

The probability of a cache block/word of Hamming Weight W not failing under a certain write/read bit error rate P_{ER} protected by a t-error correction ECC is given by the following equation:

$$P_{block} = \sum_{i=0}^{t} \binom{W}{i} (1-P_{ER})^{W-i} (P_{ER})^{i} \quad (1)$$

For overall block error rate, we calculated the probability of failure using the knowledge obtained from the memory traces about the number of reads between two consecutive write instructions to a memory address. For examples, when there are two read operations, the failure probability of any word in a cache line replicated once or more is 0. When replication isn't possible and say, DECTED protection is used, the probability of no fault is calculated by considering all the following cases: (a) When two or less faults occur during the same operation (either during write, any of the two reads or because of retention error). (b) One fault occurs during one operation and the other fault occurs during another operation. Based on our memory trace statistics, we saw that most of the cachelines are read once. However, there are still 2-5% of cachelines that are read more than twice (some even more than 10 times). In such cases, restore/write-back operation is needed after a certain number of reads to avoid aggregation of bit flips due to read disturb errors. As a result we analyze two cases: a) cache line is written back after 2 reads when there are more than 2 consecutive reads, and b) cache line is written back after every 4 reads when there are more than 4 consecutive reads. For the case of write-back after 2 reads, 4x replication is not used as no cache line will be read consecutively more than twice.

V. Results and Discussion

In this section we demonstrate that CME provides considerable benefit in terms of block error reduction as compared to a normal (72,64)SECDED.

A. Reduction in Hamming Weight

We evaluate the hamming weight reduction when using BPC scheme and compare it against Dynamic-XOR scheme proposed in [12] where the goal was to solely minimize the weight of each cache line. From Figure 7 it can be seen that for all applications both the schemes reduce Hamming Weight of cache line as compared to the original weight. BPC reduces Hamming Weight by upto 35% compared to the original weight. For most applications, cache line after BPC compression ends up with a lower Hamming Weight than Dynamic XOR. On an average BPC has 6% lower weight than the Dynamic-XOR scheme. Thus, BPC not only has the advantage of reducing cache line size over Dynamic-XOR which, in turn, allows for stronger ECC, it also reduces Hamming Weight of the entire cache line, thus reducing chances of unwanted bit flips during write and read operations in STT-RAMs.

B. Reduction in Block Error Probability

To evaluate the reduction in block errors, probability of failure is computed per application for each word in all the cache lines undergoing load operation as retrieved from the memory traces for the following two cases:

- *Scheme-1:* Original cache line with only (72,64) SECDED protection
- *Scheme-2:* Our proposed CME protection scheme

The results shown in Figure 8 are plotted for the scenarios when there are maximum of two and four consecutive reads from a particular cache line before the restore operation (where the cache line is written back). It can be seen that just adding

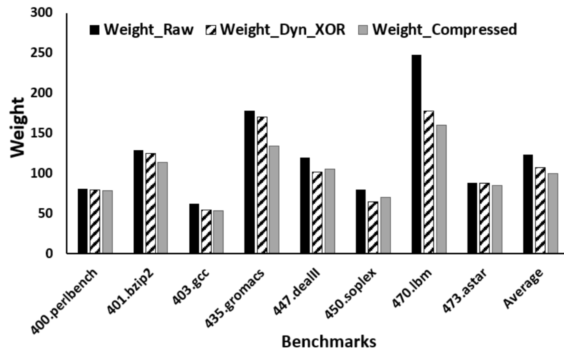

Fig. 7: Comparison of average Hamming weight of original cache line, BPC and DBX schemes

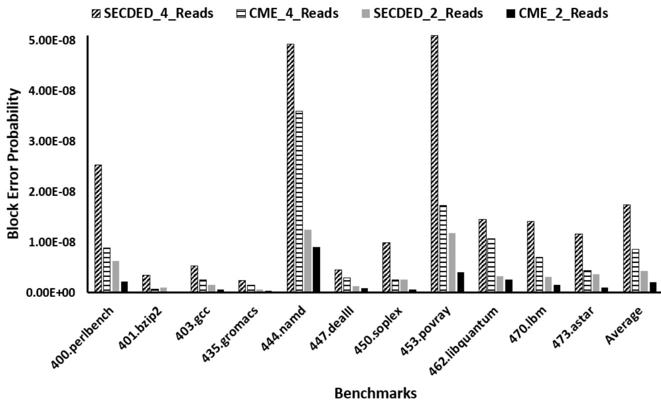

Fig. 8: Reduction in block error rate induced due to WER, RER and RDR is shown.

a (72,64) SECDED on the original cache line (Scheme 1) has much worse block failure probability than CME. This is because CME reduces Hamming Weight (reducing chances of 1→0 flip during read), reduces cache line size enabling stronger protection and enables replication in certain cases (for replication, read error probability = 0 if number of reads < number of copies). In fact, successful compression and therefore stronger protection was possible for 75.8% blocks across the benchmarks used. As a result CME provides up to 81.6% and 78.4% lower block failure probability than Scheme 1 respectively when maximum 4-reads and 2-reads are allowed before the next write-back to the cache line is performed. On an average, CME has about 50.7% and 51.2% lower block failure probability than Scheme 1. This massive reduction in error rate is achieved with only 8-bit Tag overhead as mentioned in previous section.

It is intuitive to understand that allowing smaller number of reads between multiple writes reduces read disturb error and retention errors and thus block failure probability reduces. However, for applications like *401.bzip2* and *450.soplex*, CME scheme provides lower block failure probability with 4 consecutive reads allowed than Scheme 1 with SECDED protection and maximum 2 consecutive reads allowed. This clearly shows that the restore or write-back operation can be well minimized when CME scheme is allowed.

Moreover, we argue that the opportunistic use of stronger ECC code is largely responsible for the large reduction in error probability. This is because, even for applications with low Hamming weight reduction (*perlbench, bzip2, gcc etc.*), decrease in the block error probability has been significant.

C. Overhead of Multi-ECC scheme

The CME scheme requires support for multiple ECC engines (SECDED, DECTED and 3EC4ED). Having multiple ECC encoders and decoders on a memory controller on chip can be costly in terms of both area as well as power. However, if asymmetric quantum BCH coding [24] is used, G and H matrices for a smaller ECC (for e.g., SECDED) can be composed out of sub-matrices of G and H matrices of a stronger ECC scheme (for eg. DECTED), therefore the same hardware can be reused. Since in our case, the total codeword length is same for all the cases (n=72 bits), eliminating rows from the bottom of the H-matrix of a stronger code (for e.g., 3EC4ED) would generate the H-matrix of a weaker code (for e.g., DECTED) with the same codeword length. However, for encoding using G-matrix, the rows (=k) of the G-matrix decrease as we move towards a stronger ECC code for a given constant codeword length (=n). Therefore, SECDED would require the largest encoding hardware while 3EC4ED would require the largest hardware for parity checking (H-matrix). Therefore, the only area overhead would come from the larger H-matrix compared to SECDED. Synthesizing the parity check engine using an industrial 45nm library results in about only about 3800 μm^2 of additional area overhead compared to a only SECDED implementation as in Scheme 1. Moreover, since only one word is decoded at a time, reuse isn't expected to have any performance degradation. The additional decoding energy and latency overhead of CME during read operation is much lesser than the overhead of having restore operations after every read that would have been otherwise required to achieve the same error rate with just a simple (72,64) SECDED ECC protection.

VI. Conclusion

In this work, we proposed a new ECC protection scheme for STT-RAM based main memories, compression with multi-ECC (CME). First we try to compress every cache line to reduce the size of the cache line and then based on the amount of compression possible, we use the saved additional bits to increase the protection using stronger ECC codes if possible. Compression itself reduces the hamming weight of the cache lines, thus reducing the probability of 1→0 bit-flips. Opportunistically using stronger ECC codes further helps tolerate multiple bit-flips in a cache line. Our results show that for STT-RAM based main memories, CME can reduce the block failure probability by up to 81.6% (average 50.7%) and 78.4% (average 51.2%) over using a (72,64)SECDED for each cache line word, when maximum of 4 reads and 2 reads respectively are allowed to a cache line before a write-back/restore operation is done.

VII. Acknowledgement

The authors would like to thank Professor Lara Dolecek and Clayton Schoeny from UCLA and the anonymous reviewers for their feedback.

REFERENCES

[1] S. Mittal and J. S. Vetter, "A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 1537–1550, May 2016.

[2] S. Sills, S. Yasuda, A. Calderoni, C. Cardon, J. Strand, K. Aratani, and N. Ramaswamy, "Challenges for High-Density 16Gb ReRAM with 27nm Technology," in *Symposium on VLSI Circuits (VLSI Circuits)*, pp. T106–T107, June 2015.

[3] Y. Emre, C. Yang, K. Sutaria, Y. Cao, and C. Chakrabarti, "Enhancing the Reliability of STT-RAM through Circuit and System Level Techniques," in *2012 IEEE Workshop on Signal Processing Systems*, pp. 125–130, Oct 2012.

[4] M. Y. Hsiao, "A Class of Optimal Minimum Odd-weight-column SEC-DED Codes," *IBM Journal of Research and Development*, vol. 14, pp. 395–401, July 1970.

[5] S. Kaneda and E. Fujiwara, "Single Byte Error Correcting Double Byte Error Detecting Codes for Memory Systems," *IEEE Transactions on Computers*, vol. 31, no. 7, pp. 596–602, 1982.

[6] N. H. Seong, S. Yeo, and H.-H. S. Lee, "Tri-level-cell Phase Change Memory: Toward an Efficient and Reliable Memory System," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*, pp. 440–451, 2013.

[7] H. Noguchi, K. Kushida, K. Ikegami, K. Abe, E. Kitagawa, S. Kashiwada, C. Kamata, A. Kawasumi, H. Hara, and S. Fujita, "A 250-MHz 256b-I/O 1-Mb STT-MRAM with advanced perpendicular MTJ based dual cell for nonvolatile magnetic caches to reduce active power of processors," in *2013 Symposium on VLSI Technology*, pp. C108–C109, June 2013.

[8] D. Shum, D. Houssameddine, S. T. Woo, Y. S. You, J. Wong, K. W. Wong, C. C. Wang, K. H. Lee, K. Yamane, V. B. Naik, C. S. Seet, T. Tahmasebi, C. Hai, H. W. Yang, N. Thiyagarajah, R. Chao, J. W. Ting, N. L. Chung, T. Ling, T. H. Chan, S. Y. Siah, R. Nair, S. Deshpande, R. Whig, K. Nagel, S. Aggarwal, M. DeHerrera, J. Janesky, M. Lin, H. J. Chia, M. Hossain, H. Lu, S. Ikegawa, F. B. Mancoff, G. Shimon, J. M. Slaughter, J. J. Sun, M. Tran, S. M. Alam, and T. Andre, "CMOS-embedded STT-MRAM arrays in 2x nm nodes for GP-MCU applications," in *2017 Symposium on VLSI Technology*, pp. T208–T209, June 2017.

[9] Y. Zhang, X. Wang, Y. Li, A. K. Jones, and Y. Chen, "Asymmetry of MTJ switching and its implication to STT-RAM designs," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1313–1318, March 2012.

[10] R. Takemura, T. Kawahara, K. Ono, K. Miura, H. Matsuoka, and H. Ohno, "Highly-scalable disruptive reading scheme for Gb-scale SPRAM and beyond," in *2010 IEEE International Memory Workshop*, pp. 1–2, May 2010.

[11] S. Wang, H. C. Hu, H. Zheng, and P. Gupta, "MEMRES: A Fast Memory System Reliability Simulator," *IEEE Transactions on Reliability*, vol. 65, pp. 1783–1797, Dec 2016.

[12] W. Wen, M. Mao, X. Zhu, S. H. Kang, D. Wang, and Y. Chen, "CD-ECC: Content-dependent error correction codes for combating asymmetric nonvolatile memory operation errors," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, Nov 2013.

[13] N. Kim and K. Choi, "A design guideline for volatile stt-ram with ecc and scrubbing," in *2015 International SoC Design Conference (ISOCC)*, pp. 29–30, Nov 2015.

[14] J. Kim, M. Sullivan, E. Choukse, and M. Erez, "Bit-Plane Compression: Transforming Data for Better Compression in Many-Core Architectures," in *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 329–340, June 2016.

[15] S. Mittal, J. S. Vetter, and L. Jiang, "Addressing Read-disturbance Issue in STT-RAM by Data Compression and Selective Duplication," *IEEE Computer Architecture Letters*, vol. PP, no. 99, pp. 1–1, 2017.

[16] S. Wang, A. Pan, C. O. Chui, and P. Gupta, "Tunneling negative differential resistance-assisted stt-ram for efficient read and write operations," *IEEE Transactions on Electron Devices*, vol. 64, pp. 121–129, Jan 2017.

[17] V. Sathish, M. J. Schulte, and N. S. Kim, "Lossless and Lossy Memory I/O Link Compression for Improving Performance of GPGPU Workloads," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, PACT '12, (New York, NY, USA), pp. 325–334, ACM, 2012.

[18] M. Thuresson, L. Spracklen, and P. Stenstrom, "Memory-link compression schemes: A value locality perspective," *IEEE Transactions on Computers*, vol. 57, pp. 916–927, July 2008.

[19] A. R. Alameldeen and D. A. Wood, "Adaptive cache compression for high-performance processors," in *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, pp. 212–223, June 2004.

[20] G. Pekhimenko, V. Seshadri, O. Mutlu, M. A. Kozuch, P. B. Gibbons, and T. C. Mowry, "Base-delta-immediate compression: Practical data compression for on-chip caches," in *21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 377–388, Sept 2012.

[21] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "Raidr: Retention-aware intelligent dram refresh," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, (Washington, DC, USA), pp. 1–12, IEEE Computer Society, 2012.

[22] A. Waterman, Y. Lee, D. Patterson, and K. Asanovic, "The RISC-V Instruction Set Manual Volume I: User-Level ISA Version 2.0," 2014.

[23] A. Waterman and Y. Lee, "Spike, a RISC-V ISA Simulator – git commit 3bfc00e."

[24] S. Aly Ahmed, "Asymmetric and Symmetric Subsystem BCH Codes and Beyond," 04 2008.