Enhancing Reliability of STT-MRAM Caches by Eliminating Read Disturbance Accumulation

Elham Cheshmikhani
dept. of Computer Engineering
Sharif University of Technology
Tehran, Iran
elham.cheshmikhani@sharif.edu

Hamed Farbeh
dept. of Computer Engineering
Amirkabir University of Technology
Tehran, Iran
farbeh@aut.ac.ir

Hossein Asadi
dept. of Computer Engineering
Sharif University of Technology
Tehran, Iran
asadi@sharif.edu

***Abstract*—*Spin-Transfer Torque Magnetic RAM* (STT-MRAM) as one of the most promising replacements for SRAMs in on-chip cache memories benefits from higher density and scalability, near-zero leakage power, and non-volatility, but its reliability is threatened by high read disturbance error rate. *Error-Correcting Codes* (ECCs) are conventionally suggested to overcome the read disturbance errors in STT-MRAM caches. By employing aggressive ECCs and checking out a cache block on every read access, a high level of cache reliability is achieved. However, to minimize the cache access time in modern processors, all blocks in the target cache set are simultaneously read in parallel for tags comparison operation and only the requested block is sent out, if any, after checking its ECC. These extra cache block reads without checking their ECCs until requesting the blocks by the processor cause the accumulation of read disturbance error, which significantly degrade the cache reliability. In this paper, we first introduce and formulate the read disturbance accumulation phenomenon and reveal that this accumulation due to conventional parallel accesses of cache blocks significantly increases the cache error rate. Then, we propose a simple yet effective scheme, so-called *Read Error Accumulation Preventer cache* (REAP-cache) to completely eliminate the accumulation of read disturbances without compromising the cache performance. Our evaluations show that the proposed REAP-cache extends the cache *Mean Time To Failure* (MTTF) by 171x, while increases the cache area by less than 1% and energy consumption by *only* 2.7%.**

***Index Terms*—Cache, STT-MRAM, Read Disturbance, Error-Correcting Code (ECC), Error Rate.**

I. Introduction

Static Random Access Memories (SRAMs) have been the prevalent memory technology in on-chip caches. SRAMs, however face several challenges, e.g, high leakage power and cell instability, by technology downscaling [1]–[3]. Recent development in Non-Volatile Memory (NVM) technology has made *Spin-Transfer Torque Magnetic RAM* (STT-MRAM) as the most promising alternative for SRAMs in on-chip caches. Near-zero leakage power, immunity to radiation-induced errors, higher density, better scalability, and non-volatility are the main advantages of STT-MRAM caches [4]. Beside the mentioned advantages, STT-MRAM caches are highly error-prone in read operations. When a read current is applied to cache cells during a read operation, it is probable that the content of the cells unintentionally flips. This error, known as *read disturbance*, is originated from the stochastic switching behavior of STT-MRAM cells [5]. Read disturbance error is a main source of unreliability in STT-MRAM caches.

Cache blocks are conventionally protected using *Error-Correcting Codes* (ECCs) [6], [7]. On a request to a cache block, the ECC decoder logic checks the block for a possible correctable error and sends out the corrected data. When the number of erroneous bits in a block is larger than the ECC correction capability, however, the ECC decoder fails to correct the data block. The read operation of the caches in modern processors is optimized for access time reduction [8]–[10], which significantly increases the occurrence probability of uncorrectable errors caused by read disturbances.

On a read request to a k-way set associative cache, in parallel to tag comparison operation and prior to determining the requested data block, all blocks in the target set is read simultaneously [8], [10], [11] and the requested block is sent out after checking its ECC. The other $k-1$ blocks are discarded in this case. These extra $k-1$ out of k speculative reads from blocks that are not being requested in each read access increases the read disturbance probability in these blocks. We call these extra imposed reads *concealed reads*. Reading multiple cache blocks for several times without checking their ECC causes the accumulation of read disturbance errors and occurrence of uncorrectable errors. To the best of our knowledge, none of the previous studies addressed this reliability challenge.

In this paper, 1) we first examine and formulate the adverse effect of concealed reads on the STT-MRAM cache reliability and analytically illustrate that the read disturbance accumulation severely increases the cache error rate. Then, 2) by conducting comprehensive experiments we show that the reliability of STT-MRAM cache is significantly degraded due to read disturbance accumulation. Next 3), we propose an effective scheme, so-called *Read Error Accumulation Preventer cache* (REAP-cache), to prevent the accumulation of read disturbance in cache blocks and completely eliminate the adverse effect of concealed reads on the cache reliability. REAP-cache scheme makes a minor modification in the cache read path to guarantee performing ECC checking not only for the requested block, but also for all other blocks within the cache set that have been read concurrently.

We evaluate REAP-cache using gem5 simulator [12] run-

ning a set of workloads from SPEC CPU2006 benchmark suite [13]. The results show that REAP-cache increases the $Mean\ Time\ To\ Failure$ (MTTF) of STT-MRAM caches by 171x, on average. The proposed scheme has no impact on the cache performance, increases the cache area by less than 1%, and its energy consumption overhead is $only$ 2.7%.

The rest of this paper is organized as follows. Section II describes the basics of STT-MRAM and its reliability challenges. In Section III, the observations and motivations for this work are discussed. The details of the proposed REAP-cache are presented in Section IV. Section V gives the simulation setup and evaluation results. Finally, we conclude the paper in Section VI.

II. STT-MRAM Basics

STT-MRAM cell consists of an access element and a storage element, as shown in Fig. 1(a). The access element is an NMOS transistor used to connect and disconnect the cell to the array. The other element is a magnetic storage element, named $Magnetic\ Tunnel\ Junction$ (MTJ), that uses magnetic charge to store data. MTJ consists of three layers, two ferromagnetic layers, and a thin oxide barrier layer, which separates those two ferromagnetic layers. The oxide barrier layer, which is made up of crystallized MgO, is sandwiched between $reference\ layer$ with a fixed magnetic field direction and $free\ layer$ that its field direction can be changed through an applied current [4]. The magnetization direction of the free layer can be in the same or opposite direction as the reference layer spin direction. If the spin-polarized current flow changes the orientation of the free layer spin to be parallel/anti-parallelism with reference layer spin, a low/high resistance is formed that is interpreted as storing logic value '0'/'1' in the cell.

Writing/reading to/from a STT-MRAM cell is done by applying a write/read current (I_{write}/I_{read}) for a predetermined pulse width. To write '0' in the STT-MRAM cell, I_{write} flows from bit line to source line to turn the electrons spin in the free layer in the same direction as that of in the reference layer. Writing '1' in the cell needs a current flow from source line to bit line and causes the electron charge to flow from the free layer to the reference layer. Reference layer with strong magnetic field reflects the electrons with the opposite direction to the free layer and these reflected electrons antiparallelize the magnetic field direction of the free layer and leads to write '1'.

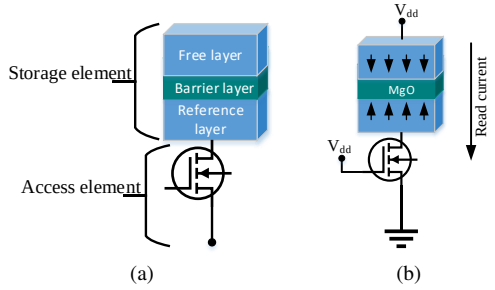

Fig. 1: (a) STT-MRAM cell structure with a MTJ unit and an access transistor and (b) Reading from STT-MRAM cell by applying read current.

To read a data from a cell, I_{read} should flow either from bit line to source line or from source line to bit line to measure the MTJ resistance using a sense amplifier. It means that, read operation is a unidirectional operation and is in the same direction as writing either '1' or '0'. Fig. 1(b) depicts the read operation from a STT-MRAM cell when the read current is adjusted to flow in the same direction as writing '0'. In this way, during a read operation, the content of the cell is probable to unintentionally switch from '1' to '0' while I_{read} flows through the cell. This unexpected bit-flip that changes the cell content erroneously is named $read\ disturbance$. The probability of read disturbance occurrence in a STT-MRAM cell is calculated by (1) [14].

$$P_{Read-Disturbance} = 1 - exp(\frac{-t_{read}}{\tau} \times exp(\frac{-\Delta(I_{read} - I_{C_0})}{I_{C_0}})) \quad (1)$$

Where, τ is attempt period and assumed to be 1ns, I_{read} is read current, I_{C0} is the current needed to write in 0°K, t_{read} is the read pulse width, and Δ is thermal stability factor.

There are some research studies that try to mitigate the read disturbance for improving STT-MRAM cells reliability. The common method in architecture-level studies is $disruptive\ reading\ and\ restoring$ [15], [16]. In these studies, a restore operation corrects the faulty bits after each read operation. Therefore, disruptive reading and restoring method increases both the cache access time and the $write\ failure$ probability as another reliability challenge. One of the studies that is categorized as a circuit-level method, try to leverage the read disturbance rate by minimizing the access transistor size [17]. This resizing decreases the read and write current and decrease the read disturbance rate, while sensing this low current causes another reliability challenge known as $false\ read$. In [18], [19] the sense amplifiers are redesigned to tackle the inability of accurate sensing, while they increase the cell area and complicate the memory design.

III. Observation and Motivation

In this section, we demonstrate that the conventional cache configuration degrades the cache reliability by accumulating read disturbance occurrence in cache blocks in read accesses. Then, we formulate the effect of read disturbance accumulation on the correctness of data blocks. Finally, we evaluate the behavior of this accumulation in cache blocks to illustrate the severity of this problem and necessity for mitigating it.

A. Read Disturbance Accumulation

Considering a conventional ECC-protected set-associative cache architecture, a sequence of operations is performed to commit a read request. First, the cache set is determined using the requested address and the tag part of the address is compared with tags of all blocks in the set to find a match. On a cache hit, the requested block is read and is sent out after checking its ECC and on a cache miss, the procedure to handle the misses is conducted. To minimize the cache access time, the logically sequential operations of tag comparison and reading from data block are conventionally performed in parallel.

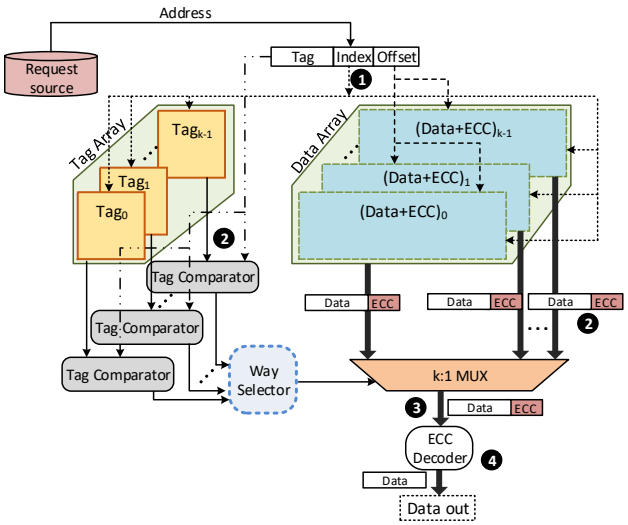

Fig. 2: Cache structure in conventional parallel (fast) access mode.

In this configuration, known as *cache parallel access* of *cache fast access* mode [8], [9], [20], [21], all data blocks have been read and ready for being sent out by the completion of the tag comparison operation. On a cache hit, the output of the tag comparison selects between the available data blocks and the other data blocks are discarded. This parallel cache access eliminates the time of accessing and reading from data block in the cost of higher energy consumption due to extra reads from data blocks. For each access, k data blocks are read in response to a request for a single block.

Fig. 2 illustrates a conventional k-way set-associative cache and the sequence of operations performed in cache fast access mode. In step ❶, the index part of the input address is extracted to determine the target cache set. In step ❷, all tags in the selected set is read and compared with the tag part of the input address. At the same time, all cache lines in the selected set are speculatively read and delivered to the inputs of the MUX unit whose control signals are the output of the tag comparison unit. In step ❸, on a cache hit, the MUX unit delivers one out of k data blocks to the ECC decoder unit and the other $k-1$ data blocks are discarded, ehereas on a cache miss all k data blocks are discarded. The selected data block on a cache hit is checked by ECC decoder unit for a possible error and data is sent out at step ❹.

As described, all k cache lines in a set are read in response to a request for reading a single line, while only the data of requested line is checked for possible error. The other lines are read and under read disturbance error without being checked by ECC decoder. These extra reads, named *concealed read* in this study, accumulate the read disturbance errors in the lines until the line is being requested.

A cache line can be under several hundreds or even thousands number of concealed reads and accumulation of read disturbance errors between two conservative results and checking its ECC. If the number of erroneous bits due to accumulation of read disturbance errors in a cache line exceeds the error correction capability of ECC, the data cannot be recovered. The next section formulates the occurrence probability of uncorrectable errors in cache lines.

B. Problem Formulation

An ECC-protected cache is conventionally capable of correcting single bit error in cache lines. We assume that unidirectional read disturbance only affects STT-MRAM cells containing logic value '1'. The probability of correct data block delivery when reading from a cache line is according to (2).

$$P_{corr-blk} = (1-P_{RD-cell})^n \\ +n \times P_{RD-cell} \times (1-P_{RD-cell})^{n-1} \quad (2)$$

Where, $P_{RD-cell}$ is the probability of read disturbance occurrence in a cell on a read access (given in (1)) and n is the number of '1' in the cache line. Equation (2) is a binomial experiment in which $P_{RD-cell}$ is the probability of failure in each Bernoulli trial, n is the number of trials and the probability of correct data block delivery ($P_{corr_{blk}}$) is the binomial probability of at most one failure.

Equation (2) depicts that the correct data is delivered if all STT-MRAM cells in the line are error-free or only one out of n cells is erroneous. In other words, ECC decoder unit is capable of delivering the correct data $if\ and\ only\ if$ at most one data cell is erroneous. As mentioned, several concealed reads are probable for a cache line before requesting it and checking its ECC.

The read disturbance probability in each concealed read is accumulated in the line and degrade the probability of correct data delivery, which is calculate according to (3).

$$P_{corr-blk-acc} = (1-P_{RD-cell})^{N\times n} \\ +N \times n \times P_{RD-cell} \times (1-P_{RD-cell})^{N\times n-1} \quad (3)$$

Where, N is the number of concealed reads occurred between two consecutive accesses to the line plus one (to count the last read access). Interpreting as a binomial experiment, by comparing (2) and (3), it is evident that concealed reads increase the number of trials from n to $N\times n$ while the number of failures still must be at most one in binomial probability.

We further clarify the effects of concealed reads on the occurrence probability of uncorrectable reads by a numerical example. Assume that 100 bits out of 512 bits of a cache line contain logic value '1' and $P_{RD-cell} = 10^{-7}$. The probability of uncorrectable error in this line after a read access without any concealed read is according to (4):

$$P_{err} = 1 - P_{corr-blk} = 1 - ((1-10^{-8})^{100} + \\ 100 \times 10^{-8} \times (1-10^{-8})^{99}) = 5.0^{-13} \quad (4)$$

and probability of uncorrectable error considering 50 concealed reads is according to (5):

$$P\prime_{err-acc} = 1 - P_{corr-blk-acc} = 1 - ((1-10^{-8})^{100\times 50} + \\ 50 \times 100 \times 10^{-8} \times (1-10^{-8})^{100\times 50-1}) = 1.3 \times 10^{-9} \quad (5)$$

The above example illustrates that only 50 concealed read increases the probability of inability to deliver a correct data block by more than 3 orders of magnitude.

The above analysis is supported by a set of experiments for an 8-way set-associative L2 cache evaluated in gem5 simulator [12] and using workloads from SPEC CPU2006 benchmark suite [13]. The evaluations show that the number of concealed reads in cache lines can be even higher than 10^5

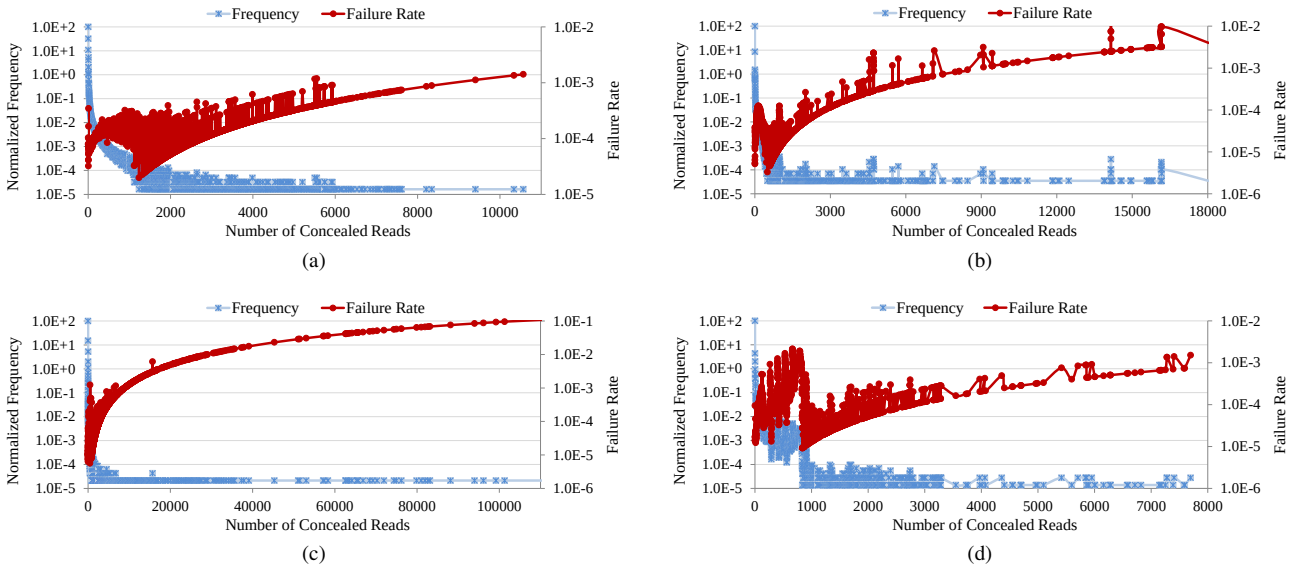

Fig. 3: Frequency of each number of concealed reads for cache access during the workload execution and the contribution of each frequency in total cache failure rate (a) perlbench workload, (b) calculix workload, (c) h264ref workload, and (d) dealII workload.

in some workloads, which extremely increases the occurrence probability of read disturbance error in more than one bit.

Fig. 3 shows the frequency of the number of concealed reads for all read accesses during the workload execution as well as their contribution in total cache failure probability, for four exemplary workloads. More precisely, x-axis is the number of concealed reads and the primary y-axis is the number of read requests to all lines with each specific number of concealed reads, which is normalized to the number of read accesses from lines with no concealed read. For example, point (35, 3) means that 3 lines are read with 35 concealed reads during the workload execution, if the number of reads from lines without any concealed read is scaled to 100. The secondary y-axis depicts the failure probability considering both the number of concealed reads and frequency of each concealed reads. This probability has direct relation with both frequency and the number of concealed reads. Both primary and secondary y-axes are in logarithmic scale.

The results show that the frequency of reading from lines with a each concealed read decreases for higher number of concealed reads, while the contribution of lines with high number of concealed reads in total cache failure rate is significantly higher. For example, in $perlbench$ workload depicted in Fig. 3(a), the frequency of reading from lines with 8000 concealed reads is by six orders of magnitude lower than that of lines with less than five concealed reads, while the cache failure probability due to the former is higher than that of the latter by two orders of magnitude.

The same trend is observed in $calculix$ depicted in Fig. 3(b), in which the cache failure rate due to line with 10^4 concealed reads is by more than 100x higher than that of lines with less than 20 concealed reads, despite the extremely lower frequency of the former. The number of concealed read in $h264ref$ workload exceeds 10^5 for some cache lines, which makes them the main contributor in total cache failure rate, according to Fig. 3(c). The cache lines with higher number of concealed reads shown in Fig. 3(d) also are the dominant contributor in cache failure rate, for $dealII$ workload.

The above observations demonstrate that concealed reads in conventional cache configuration significantly degrade the STT-MRAM reliability in the sake of higher cache performance. In the next section, we propose a scheme to eliminate the impact of concealed reads on the reliability without affecting the cache performance.

IV. Proposed Scheme

In a conventional k-way set-associative cache structure, each read from a line in a set imposes $k-1$ extra reads from other lines in that set. These concealed reads are discarded without checking their ECCs. This causes read disturbance error accumulation and decreases the probability of correct read operation. To confront this challenge and eliminate the adverse effect of concealed reads, two approaches can be imagined: 1) reading a data line after completion of tag comparison operation, while removes the performance benefit of cache parallel access and significantly increases the cache access time, 2) performing ECC checking for all data lines on a read access instead of only checking the requested line.

Our proposed scheme is based on the second approach and guarantees to prevent read disturbance accumulation without affecting the cache performance. Taking into account the cache operation on the read path in Fig. 2, we propose the low-cost *Read Error Accumulation Preventer-cache* (REAP-cache) scheme, which makes a minor modification in the cache structure. This modification provides the opportunity of simultaneously checking ECC of all data lines accessed during concealed read in addition to checking the requested line.

Fig. 4 depicts the cache structure in REAP-cache scheme. Referring to the conventional cache structure in Fig. 2, in REAP-cache scheme, we swap the location of ECC decoder

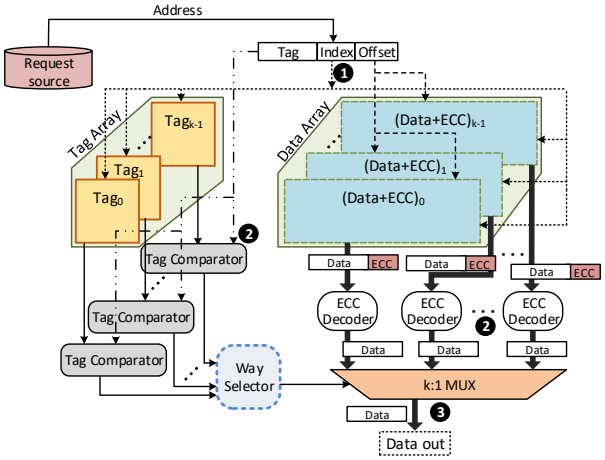

Fig. 4: Cache structure in proposed REAP-cache.

unit and the MUX unit in the read path. To make it possible to simultaneously check all data blocks, we replicate the ECC decoder unit equal to number of lines accessed in parallel. By this modification in the read path, we guarantee to perform ECC checking for all cache line after each read operation regardless of the type of read, i.e., concealed or real read. It can be theoretically proven that the cache access time is not increased by this swapping of MUX and ECC decoder unit.

The sequence of cache operations in a read request is depicted in Fig. 4. Same as in the conventional caches, the target set is determined by index part of the input address in step ❶. In step ❷, all tags in the selected set are read and compared with the input tag and all data lines are read simultaneously. All k data blocks are sent to k ECC decoding units and the corrected data are delivered to MUX unit by the end of this step. The output of tag comparison unit selects one of the blocks in step ❸ and the data is sent out by the cache.

REAP cache scheme eliminates the accumulation of read disturbance errors by transforming the checking a cache line after N concealed reads into N times checking the line each after a read operation. In this case, the probability of uncorrectable error in the line will be according to (6):

$$P\prime_{corr-blk_R EAP-cache} = ((1 - P_{RD-cell})^n + n \times P_{RD-cell} \times (1 - P_{RD-cell})^{n-1})^N \quad (6)$$

Considering our previous example in Section III for a line with 100 bits of '1' and 50 concealed reads, the probability of uncorrectable error is 2.6×10^{-11}, which is 50x lower than that of conventional cache.

V. Simulation Setup and Results

To evaluate the proposed REAP-cache scheme, we modeled a processor containing two levels of on-chip caches in gem5 full-system simulator [12]. L1 cache is comprised of dedicated data and instruction cache with SRAM technology and L2

TABLE I: Configuration of On-Chip Caches

L1 I-cache	32KB, 4-way set-associative, 64B block size, write-back, SRAM
L1 D-cache	32KB, 4-way set-associative, 64B block size, write-back, SRAM
L2 cache	1MB, 8-way set-associative, 64B block size, write-back, STT-MRAM

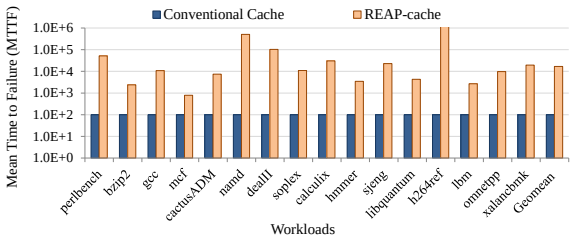

Fig. 5: Mean Time To Failure of cache in REAP-cache normalized to conventional cache.

cache is shared with STT-MRAM technology. The details of cache configuration is given in Table I. We use SPEC CPU2006 benchmark suite [13] as the workload and extract the result by running one billion instructions after skipping the initial 100 million warm-up instructions. REAP-cache scheme is compared with the conventional cache as the baseline in terms of reliability, energy consumption, area, and performance. As a reliability metric, we calculated the Mead Time to Failure (MTTF) of cache in REAP-cache and normalized it to MTTF in the baseline. The STT-MRAM cache is modeled in NVSim simulator [22] to extract the energy consumption, area, and access time parameters. In the following, we first investigate the cache reliability and then explore the overheads of REAP-cache scheme.

A. Reliability Evaluation

REAP-cache scheme increases the cache reliability by eliminating the accumulation of read disturbance errors. Read disturbance error is probable for each read access in a cache line and when the line is not check for several accesses, these errors are accumulated and the ECC decoder is not capable of delivering correct data if the number of errors exceeds the ECC correction capability. By performing ECC checking after each read access to a line in REAP-cache scheme, ECC decoder needs to only correct the errors occurred in a single read access, which results in a higher error correction capability and lower cache failure rate. Fig. 5 depicts the MTTF of the STT-MRAM cache in REAP-cache scheme normalized to the baseline for all workloads. The results show that REAP-cache improves the MTTF of the cache by 171x, on average. This value for some workloads is more than 1000x, e.g., $namd$, $dealII$, $h264ref$. In the worst case, REAP-cache increases the MTTF by 7.9x in mcf workload. Increasing the MTTF of the cache is equivalent to reducing its error rate by an average of 171x, which is achieved by providing higher error correction capability in REAP-cache.

B. Overhead

REAP-cache has no effect on the cache energy consumption in write operations as well as in tag unit operations. However, REAP-cache increases the cache energy consumption by performing larger number of ECC checking per read access. There is single ECC decoder unit in conventional cache structure for checking the requested cache line, whereas REAP-cache requires eight ECC decoder units for simultaneously checking all data blocks in a cache set. The contribution of ECC decoder

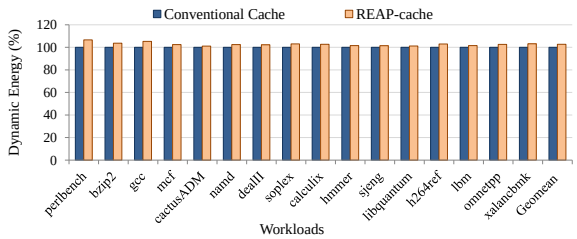

Fig. 6: Dynamic energy consumption in REAP-cache normalized to conventional cache.

unit in total energy consumption of the cache is less than 1% and the impact of the modification made by REAP-cache in cache structure on the energy consumption is not significant.

Fig. 6 shows the dynamic energy consumption of STT-MRAM cache in REAP-cache normalized to the baseline for all workloads. According the results, REAP-cache increases the energy consumption by an average of 2.7%. The worst case overhead of 6.5% is observed in cactusADM workload and in the best case, REAP-cache imposes 1.0% overhead in $xalancbmk$ workload. The variations in the energy consumption overheads is mainly due to different contributions of read accesses in total energy consumption. The fraction of read accesses in total cache accesses as well as fraction of dynamic energy in total energy consumption varies for different workloads, which leads to a small variation in the overhead of REAP-cache.

In term of area, the overhead of REAP-cache is only due to including ECC decoder units. Based on our evaluations using NVSim [22], the contribution of ECC decoder unit in total cache area is about 0.1%. Therefore, the overhead due to increasing the number of ECC decoder units from one to eight in an 8-way associative cache is less than 1%.

From performance point of view, REAP-cache has no adverse effect on the cache access time. By comparing the cache structure in Fig. 2 and Fig. 4, it is evident that read path access time in REAP-cache is less than or equal to that of the baseline. By swapping the MUX unit and ECC decoder unit in the read path, REAP-cache provides the opportunity to overlap not only the data lines accesses, but also the ECC decoding operation with the tag comparison operation. Hence, the cache access time can be even reduced in REAP-cache.

VI. Conclusion

Cache lines are accessed in parallel to tag comparison operation for minimizing the cache access time. This paper revealed that the consequence of this optimization is a previously unknown reliability problem, i.e., read disturbance accumulation, when replacing SRAMs with STT-MRAM technology in on-chip caches. Parallel accesses to all cache lines before determining the requested line significantly degrade the cache reliability by increasing the occurrence probability of uncorrectable errors. We proposed REAP-cache scheme to completely eliminate the accumulation of read disturbance errors. By making a minor modification in the cache structure, REAP-cache extends the MTTF of the STT-MRAM caches by an average of 171x. This significant reliability improvement is achieved with no performance degradation and imposing negligible area and energy consumption overhead.

Acknowledgment

This work has been partially supported by the Iran National Science Foundation under Grant No.: 96006071 and Iran National Elites Foundation.

References

[1] R. Salkhordeh and H. Asadi, "An operating system level data migration scheme in hybrid dram-nvm memory architecture," in *Proc. Des., Autom. & Test Euro. Conf. & Exh.*, 2016, pp. 936–941.

[2] M. Tarihi *et al.*, "A hybrid non-volatile cache design for solid-state drives using comprehensive i/o characterization," *IEEE Trans. Comput.*, vol. 65, no. 6, pp. 1678–1691, 2016.

[3] H. Farbeh *et al.*, "A-cache: Alternating cache allocation to conduct higher endurance in nvm-based caches," *IEEE Trans. Circuits Syst. II*, 2018.

[4] E. Cheshmikhani *et al.*, "Investigating the effects of process variations and system workloads on reliability of stt-ram caches," in *Proc. Euro. Depend. Comput. Conf.*, 2016.

[5] T. Na *et al.*, "Read disturbance reduction technique for offset-canceling dual-stage sensing circuits in deep submicrometer stt-ram," *IEEE Trans. Circuits Syst. II*, vol. 63, no. 6, pp. 578–582, 2016.

[6] E. Cheshmikhani, H. Farbeh, and H. Asadi, "Robin: Incremental oblique interleaved ecc for reliability improvement in stt-mram caches," in *Proc. IEEE Asia South Pacific Des. Automat. Conf.*, 2017.

[7] 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.

[8] H. Farbeh and S. G. Miremadi, "Psp-cache: A low-cost fault-tolerant cache memory architecture," in *Proc. Des., Autom. & Test Euro. Conf. & Exh.* European Design and Automation Association, 2014, p. 164.

[9] J. Dai, M. Guan, and L. Wang, "Exploiting early tag access for reducing l1 data cache energy in embedded processors," *IEEE Trans. VLSI*, vol. 22, no. 2, pp. 396–407, 2014.

[10] A. Sembrant, E. Hagersten, and D. Black-Shaffer, "Tlc: A tag-less cache for reducing dynamic first level cache energy," in *Int. Symp Microarchitect.* IEEE, 2013, pp. 49–61.

[11] S. Mittal, "A survey of architectural techniques for improving cache power efficiency," *Sustain. Comput.: Info. & Syst.*, vol. 4, no. 1, pp. 33–43, 2014.

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

[13] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Arch. News*, vol. 34, no. 4, pp. 1–17, 2006.

[14] Z. Pajouhi *et al.*, "Yeild, area, and energy optimization in STT-MRAMs using failure-aware ECC," *ACM J. Emerging Tech. Comput. Syst.*, vol. 13, no. 2, p. 20, 2017.

[15] R. Wang *et al.*, "Selective restore: An energy efficient read disturbance mitigation scheme for future stt-mram," in *Proc. IEEE Des. Automat Conf.*, 2015, pp. 1–6.

[16] R. Takemura *et al.*, "Highly-scalable disruptive reading scheme for gb-scale spram and beyond," in *Memory Workshop (IMW), 2010 IEEE International*. IEEE, 2010, pp. 1–2.

[17] W. Zhao *et al.*, "Design considerations and strategies for high-reliable stt-mram," *Microelectron. Rel.*, vol. 51, no. 9-11, pp. 1454–1458, 2011.

[18] T. Na *et al.*, "Read disturbance reduction technique for offset-canceling dual-stage sensing circuits in deep submicrometer stt-ram," *IEEE Trans. Circuits Syst. II*, vol. 63, no. 6, pp. 578–582, 2016.

[19] W. Zhao *et al.*, "High speed, high stability and low power sensing amplifier for mtj/cmos hybrid logic circuits," *IEEE Trans Magnet.*, vol. 45, no. 10, pp. 3784–3787, 2009.

[20] J. Dai and L. Wang, "An energy-efficient l2 cache architecture using way tag information under write-through policy," *IEEE trans. VLSI*, vol. 21, no. 1, pp. 102–112, 2013.

[21] H. Park, S. Yoo, and S. Lee, "A multistep tag comparison method for a low-power l2 cache," *IEEE Trans. Comput-Aided Des. Integrat. Circuits and Syst.*, vol. 31, no. 4, pp. 559–572, 2012.

[22] X. Dong *et al.*, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Des. Integ. Circuits & Syst.*, vol. 31, no. 7, pp. 994–1007, 2012.