# Opportunistic Write for Fast and Reliable STT-MRAM

Nour Sayed Mojtaba Ebrahimi Rajendra Bishnoi Mehdi B. Tahoori
Karlsruhe Institute of Technology (KIT) Karlsruhe, Germany
E-mail: {nour.sayed, mojtaba.ebrahimi, rajendra.bishnoi, mehdi.tahoori}@kit.edu

***Abstract*—Due to the stochastic switching behavior of the bit-cell in *Spin Transfer Torque Magnetic Random Access Memory* (STT-MRAM), an excessive write margin is required to guarantee an acceptable level of reliability and yield. This prevents the usage of STT-MRAM in fast memories such as L1 or L2 caches. The excessive write margin of STT-MRAM can be reduced to a large extent by an opportunistic write (i.e., terminating the write process before all bit switchings are completed) and by reducing thermal stability factor. The bits with unfinished writes have to be processed by robust *Error Correction Codes* (ECCs). However, such coding schemes have relatively large decoding latencies, which increases the overall read latency significantly. Moreover, thermally induced retention failures can limit the applicability of such schemes. In this paper, we exploit the fact that error detection is much faster than correction. Therefore, the errors can be detected quickly and all erroneous data can be reverted before they arrive critical parts of the system (e.g., commit stage or memory ports). We also provide an adaptive approach to manage temperature-dependent retention failures at runtime. Hence, our proposed approach enables the use of STT-MRAM technology for fast cache applications.**

## I. Introduction

The conventional memory technologies such as DRAM and SRAM suffer from high leakage power particularly in nanoscale technology nodes [1]. In order to address this challenge, the semiconductor industry is actively searching for alternative memory technologies, including magnetic memories which have close-to-zero leakage power. *Spin-Transfer Torque Magnetic RAM* (STT-MRAM) is a promising candidate to be a universal memory due to its various advantages such as non-volatility, scalability, fast read access, high density, CMOS-compatibility, and virtually unlimited endurance [2, 3].

Despite all these benefits, the long write latency and high write energy limit the application of STT-MRAM in high performance memories such as first level caches (e.g. L1) [4]. The write operation in STT-MRAM is of stochastic nature where the write latency follows a log-normal distribution with a long tail [5]. The common approach to address the write uncertainty is to consider a static timing margin. However, the amount of timing margin required to satisfy a reasonable *Write Error Rate* (WER) imposes considerable performance and energy penalty [6, 7].

Several circuit-level techniques [6–10] have been proposed to detect the actual switching time of an STT-MRAM bit cell and stop the unnecessary current flow immediately after the write completion. Although such techniques significantly reduce the write energy, the overall write latency remains the same. Recent studies show that *Error Correction Coding* (ECC) can be used to correct write errors in the long tail of the stochastic write distribution [11, 12]. This approach is able to improve both write latency and write energy by reducing the excessive write margin to a large extent for the same target WER [11]. The amount of reduction is maximized by exploiting more robust ECCs which can correct multiple errors in a single word [11–13]. Unfortunately, the decoding latency of robust ECCs is relatively high (i.e., multiple cycles) [14]. This significantly impairs the read latency, and hence, erodes the gain from write latency improvement. Therefore, the conventional multiple error correction approach is quite ineffective for fast caches because the relative impact on the read latency is considerable. At the device-level, the extreme write margin can be further addressed by reducing the thermal stability factor and trading off retention time, which speeds up the switching of MTJ cell [15]. However, high operating temperatures at runtime can further reduce the retention time of the memory significantly and cause unacceptably high retention failure rate.

In this paper, we propose a new opportunistic write scheme for STT-MRAM to improve its performance in order to be used for fast caches. The proposed technique terminates the write operation opportunistically by allowing unfinished switching of the bits with large write latency. Furthermore, we propose to lower the *thermal stability factor* ($\Delta$) to make the MTJ switching faster. As a result, the associated error rate of the proposed approach is relative high because of: i) the unfinished write, and ii) the higher retention failure rate due to the smaller $\Delta$. To address these challenges, we enable a fast yet robust ECC. This is achieved by decoupling error detection and correction steps, and performing speculative computation on the unchecked data. The error detection and correction phases are done in parallel to the speculative data processing. The fast error detection (within a cycle) guarantees error detection before widening the error sphere (e.g. committing erroneous data), to ensure error confinement and guarantee data integrity. Moreover, our adaptive approach, which is proposed to manage temperature-dependent retention failures, switches to a more conservative write scheme when operating temperature goes above the threshold. Consequently, our proposed technique enables fast, reliable and energy-efficient STT-MRAM design for high performance memories. Our simulation results show that this technique improves the performance of an STT-MRAM technology by 39.4% compared to the conventional ECC-based solutions, at the cost of negligible area overhead.

The rest of the paper is organized as follows: The basics of STT-MRAM and its reliability issues are presented in Section II. The details about our proposed technique along with the existing ECC-solutions are described in Sections III and IV, respectively. Finally, the conclusion is given in Section V.

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

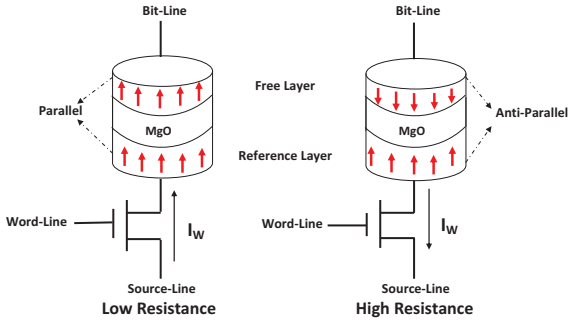



Fig. 1. Typical STT-MRAM bit-cell structure

## II. Background

### A. STT-MRAM Basics

As shown in Figure 1, a typical STT-MRAM bit-cell consists of a *Magnetic Tunnel Junction* (MTJ) device and an access transistor. The MTJ device is mainly composed of two ferromagnetic layers, namely *reference* and *free* layer. These two layers are separated by a very thin oxide barrier. The magnetic orientation of the reference layer is fixed, while that of the free layer is adjusted according to the direction of the current flowing through the MTJ. Each MTJ device is used to store one bit value depending on the relative magnetic orientation of the two ferromagnetic layers. When the magnetization of the two layers are parallel ('P'), the MTJ has a low resistance value. In contrast, the anti-parallel ('AP') magnetization leads to higher resistance of the MTJ. These two resistance states are utilized to represent logic '0' and '1'.

The read access in STT-MRAM requires a small current to sense the resistance value of the MTJ, whereas, writing into a bit-cell needs a high current to change the logic state of the cell. The direction of the write current determines the value to be written. Furthermore, the write operation is intrinsically stochastic due to the random thermal fluctuation in the MTJ, which causes significant variations in the switching time. As shown in Figure 2, the 'AP' switching delay has a wider distribution with very long tail compared to the 'P' switching delay. Therefore, the write latency of an STT-MRAM memory is determined with respect to its 'AP' switching delay distribution to guarantee a certain level of reliability. Thus, STT-MRAM write latency is pessimistic and leads to a significant performance penalty and a considerably high write energy.

### B. Failures In STT-MRAM

In despite of all STT-MRAM benefits, it suffers from various reliability threats including i) *read disturb failure* which occurs when the read current erroneously switches the bit-cell content, ii) *read decision failure* which occurs when the read circuitry delivers an incorrect logic value of the bit-cell, iii) *write error* as the desired data may fail to be stored

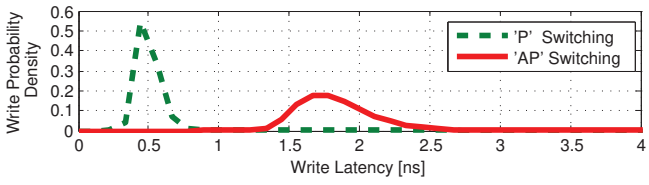



Fig. 2. Write latency distribution for 'AP' and 'P' switching delays for a 64-bit data. [$\Delta = 30$, $I_w/I_c = 3$, see Section IV-A for setup]

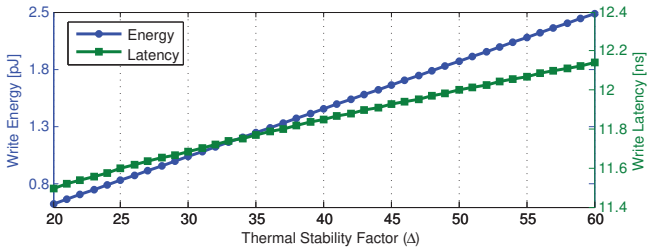



Fig. 3. Impact of $\Delta$ value on the energy and the latency of the switching of an MTJ cell [WER=$10^{-18}$]

correctly, and iv) *retention failure* which refers to the random flipping of the bit-cell content in zero current [16–18]. the required error rate of the read disturb and read decision failure, which is around $10^{-23}$ [19], can be easily maintained using low cost solutions during design-time [20]. Therefore, in this work, we will focus only on the write errors and the retention failures.

During the write operation, if the write current is terminated before the MTJ switching, the new data will fail to be stored correctly, leading to an error, and the *Write Error Rate* (WER) can be expressed as [21]:

$$WER_{bit}(t_w) = 1 - exp[\frac{-\pi^2 \cdot (I-1) \cdot \Delta}{4(I \cdot exp[C(I-1)t_w] - 1)}], \quad I = \frac{I_w}{I_c} \qquad (1)$$

Here $t_w$ is the write period and $C$ is a technology dependent parameter. In this equation, $I$ is the ratio of the write current ($I_w$) to the critical current ($I_c$) and $\Delta$ is the thermal stability factor. According to this equation, the increase in the write current can reduce the write period for a given error rate proportionally. However, this leads to a higher energy, and also there is a limit on the amount of the write current due to oxide breakdown. The alternative solution is to use a smaller $\Delta$ in which the barrier is lowered by reducing the volume of the free layer. This way, the switching can happen faster because of lowering the tail of the switching distribution. Moreover, it results in significant reduction in write energy, as described in Figure 3. The $\Delta$ value is calculated using the volume of the free layer (V), the effective field anisotropy ($H_k$), the saturation magnetization ($M_s$), and the temperature in kelvin (T), and modeled as [19]:

$$\Delta = \frac{V.H_K.M_S}{2.K_B.T} \qquad (2)$$

where $K_B$ is the Boltzmann constant. The retention time, which has to be evaluated based on the maximum possible time data is required to reside in the memory, depends mainly on $\Delta$ value, as described in [19]. Therefore, $\Delta$ has a significant impact on the retention failure probability ($P_{RF}$), which is modeled as Equation (3), and hence, on the overall reliability. Here $\tau$ is a constant value which is equal to 1 ns.

$$P_{RF} = 1 - exp[-\frac{t_R}{\tau \cdot exp[\Delta]}] \qquad (3)$$

### C. Temperature Effects on Thermal Stability of STT-MRAM

According to the Equation (2), the value of thermal stability factor ($\Delta$) is highly sensitive to the temperature. This means, it is important to consider the effects of temperature variations on the thermal reliability of STT-MRAM cell for a real application. According to Equations (2 and 3), as the temperature reduces, $\Delta$ reduces linearly, while retention failure increases exponentially. In other words, any minor change in

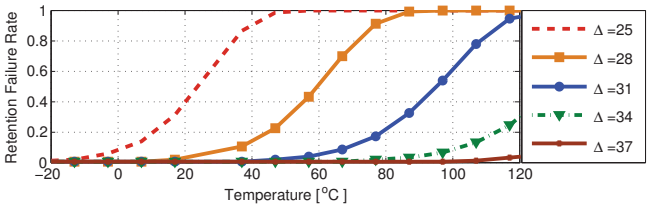



Fig. 4. Impact of the temperature on the retention failure rate of MTJ cell for various $\Delta$ values

the temperature significantly affects in the retention failure rate (see Figure 4). On the other hand, during the write operation, as the temperature increases, the high resistance of the MTJ decreases. This leads to decrease the $\Delta$ value, which results in the reduction of the critical current and hence, the switching current. This means, for the same write pulse set for target WER $> 10^{-14}$, WER improves at higher temperatures, as shown in Figure 5.

## III. Proposed Approach

### A. Motivation

In general, the excessive write margin of STT-MRAM in high-performance memories such as an L1 cache could be reduced to a large extent by exploiting small value of $\Delta$ (see Section II-B) and robust ECCs [11, 22]. The impact of such techniques could be analyzed by studying the impact of write errors at the word-level. At this level, a write operation is considered as a fail when at least one of the bits is not written in the given time margin. Hence, the write probability $WP_{word}(t_w)$ of an n-bit word in $t_w$ can be computed using Equation (4). However, in a word protected with ECC(e), e write errors can be tolerated. Hence, a binomial distribution is used to determine the write probability $WP^{e}_{word}(t_w)$, according to Equation(5).

$$WP_{word}(t_w) = WP_{bit}(t_w)^n = (1 - WER_{bit}(t_w))^n \quad (4)$$

$$WP^{e}_{word}(t_w) = \sum_{i=0}^{e} \binom{n}{i} \cdot (1 - WP_{bit}(t_w))^i \cdot WP_{bit}(t_w)^{n-i} \quad (5)$$

Here $WP_{bit}(t_w)$ is the write probability for a single bit. Among all the various ECC schemes, *Bose-Chaudhuri-Hocquenhgem* (BCH) code is most appropriate for STT-MRAM as it can correct multiple random errors [22].

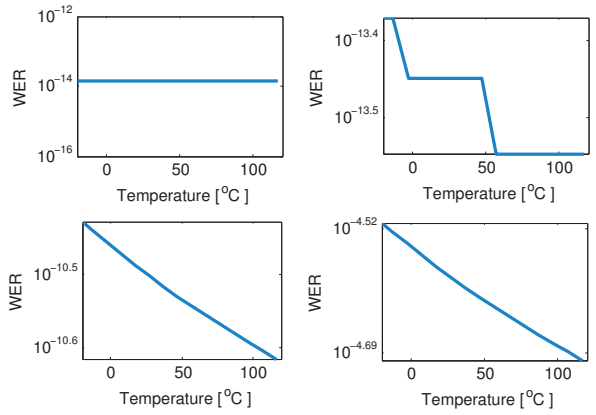



Fig. 5. Impact of the temperature on WER of MTJ cell for different write margins [$\Delta = 30$, $I_w/I_c = 3$]

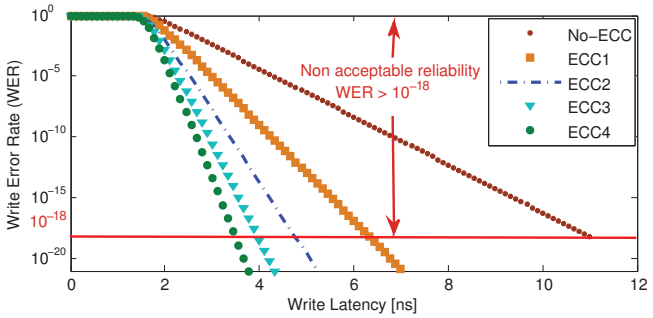



Fig. 6. WER versus write latency for 64-bit data with different ECCs [excluding ECC encoding, $\Delta = 30$, $I_w/I_c = 3$, see Section IV-A for setup]

Figure 6 illustrates the WER value for various write latencies in a 64-bit data-word with different ECC schemes that can correct 1–4 errors (denoted as ECC1 to ECC4). As shown in the figure, to achieve an acceptable WER value (i.e., $10^{-18}$ [19]), the write latency is significantly reduced (from 11.61 ns to 3.60 ns) by increasing the ECC robustness. However, robust ECC schemes have excessively large decoding latencies as the error correction (identifying the positions of all errors) is done in an iterative manner [14]. This means, the decoding overhead of robust ECCs negatively affects the read latency, and hence, the overall performance. Table I reports the breakdown of write/read latencies and detection/correction overheads for ECC1 to ECC4. The key to effectively address the long decoding latency of the conventional ECC-based approach is to eliminate the decoding latency from the read latency. In the conventional ECC-based approach, where data processing is done in a serial order, data is not available prior to the completion of the decoding process and only valid data (checked and corrected) can be propagated to other parts of the system. In contrast, in our proposed approach, non-checked data is simultaneously propagated to the decoder and to other parts of the system. This means that the data validation is not completed prior to data access and the propagated data might be erroneous. However, an erroneous data will be detected and reverted before it reaches the critical parts of the system. For this reason, this new ECC approach is referred to as *Lazy-ECC* approach.

### B. Proposed Opportunistic Write Operation Using Lazy-ECC

In Lazy-ECC, the erroneous data has to be detected before it reaches to the critical parts of the system to ensure error confinement in order to guarantee data integrity. For example, in pipeline stages of a high-performance processor, errors have to be detected before the data reaches the commit stage. In case there is no error in the propagated data, a significant reduction in read latency is achieved. Once an error is detected by the decoding circuitry, all speculative computations based

TABLE I. Read/write and detection/correction latencies of 64-bit STT-MRAM memory with 16 KB capacity [experimental setup in section IV-A]

| | | No-ECC | ECC1 SECDED | ECC2 BCH | ECC3 BCH | ECC4 BCH |
|---|---|---|---|---|---|---|
| Write [ns] | ECC Encoding | — | 0.40 | 0.52 | 0.53 | 0.55 |
| | Memory Write | 11.61 | 6.46 | 4.91 | 4.10 | 3.63 |
| | Overall | 11.61 | 6.86 | 5.43 | 4.63 | 4.17 |
| Read [ns] | ECC Decoding | — | 0.58 | 2.46 | 3.70 | 4.71 |
| | Memory Read | 0.90 | 0.90 | 0.90 | 0.90 | 0.90 |
| | Overall | 0.90 | 1.48 | 3.36 | 4.60 | 5.61 |
| Error Detection [ns] | | — | 0.37 | 0.40 | 0.44 | 0.44 |
| Error Detection and Correction [ns] | | — | 0.58 | 2.46 | 3.70 | 4.71 |

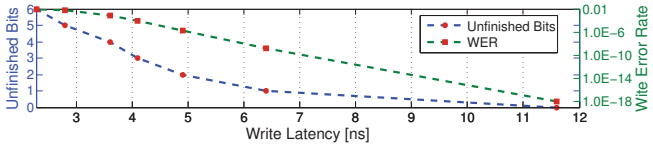



Fig. 7. Unfinished bits with corresponding probability vs. write latency for 64-bit data [$\Delta = 30$, $I_w/I_c = 3$, setup in Section IV-A]

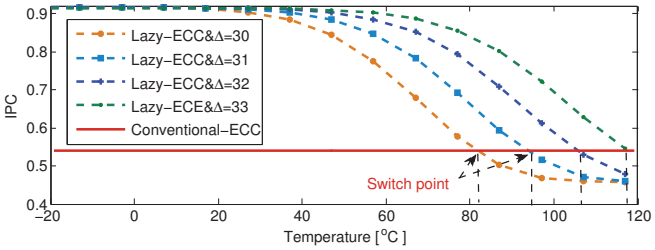



Fig. 8. *Instruction Per Cycle* (IPC) of Lazy-ECC for various $\Delta = 30$ values versus conventional-ECC under different operating temperature values

on the erroneous data are reverted and the computations are repeated using the corrected data. As shown in Table I, the detection latency is always smaller than the correction latency and the gap between these two increases significantly as ECC becomes more robust (e.g., $11\times$ for ECC4). To this end, for reverting all speculative computations based on erroneous data, only detecting the error is enough and the correction can be performed later.

A typical BCH decoder has no feedback structure, thus it can be efficiently broken up into two pipeline stages: 1) error detection which consists of syndrome vector generation and evaluation, and 2) error correction by generating the error locator polynomial and finding the error location numbers with subsequent error correction. In this regard, our proposed opportunistic write operation will be terminated faster than the conventional one with a particular probability of having unfinished bits. In case there are some unfinished bits, the Lazy-ECC scheme will take care of rewriting or correcting the unfinished bits. Figure 7 depicts the opportunistic write margin distribution at word-level. As shown in this figure, to have an acceptable WER (i.e., $10^{-18}$ [19]), the write latency has to be almost 12 ns. However, this margin can be significantly reduced at the expense of accepting higher WER, which can be translated into the expected number of erroneous bits in the word. This means, any reduction in the write margin defines a particular value of the corresponding error correction capability. For instance, the write latency can be reduced to below 4 ns (3X reduction) by expecting up to three erroneous bits per word.

In order to analyse the performance gain of our approach compared to the conventional one, we need to study the effects of the error rate. In the proposed scheme, we are dealing with errors due to the write operation and the retention period, since the write latency and retention time are reduced to meet the fast cache requirements. The sum of these error rates is named as *Total Error Rate* (TER). Let us assume that the decoding, encoding, read, and write latencies are $t_D$, $t_E$, $t_R$, and $t_W$, respectively. The expected latency of the conventional ECC-based approach in each read access is $T_{Conv} = t_R + t_D$. Whereas, in our approach, if there is no error, there is no penalty and the read latency would be equal to $t_R$. However, if there is an error, the read latency will be equal to $(t_R + t_D + t_E + t_W + t_R)$, because the corrected data has to be encoded, written to the memory by encoder, and then read again. Therefore, the overall expected read latency in our approach can be calculated as follows:

$$T_{Lazy} = TER \times (t_R + t_D + t_E + t_W + t_R) + (1 - TER) \times t_R. \quad (6)$$

Since the values of write error and retention failure rates are very small, the expected read latency of our approach can be estimated by: $\lim_{TER \to 0} T_{Lazy} = t_R$.

### C. Adaptive Approach to address Temperature-induced Retention Failures

As shown in Sec. II-B, the retention failure increases exponentially under elevated temperatures, while WER slightly reduces. This means, according to the Equation (6), as the latency of Lazy-ECC scheme depends mainly on the TER value, the overall performance of the Lazy-ECC scheme is highly affected by higher operational temperature due to the increase in the retention failure rate. Whereas, the performance of the conventional-ECC scheme is totally independent from the TER value and hence from the runtime temperature. Therefore, we propose a non-uniform ECC scheme for fast STT cache design.

There are two important parameters in the proposed hybrid scheme: i) the actual value of TER that can be determined based on the actual operating temperature, which can be obtained from on-chip thermal sensors, and ii) the threshold value ($TER_{th}$), which can be defined as a critical level of TER where we have to switch the adopted ECC approach. The optimal case can be achieved by adopting Lazy-ECC when TER is low enough (i.e., at low operating temperature). Afterwords, the performance degrades at higher temperatures due to TER increase. However, the performance of Lazy-ECC is still better than the conventional-ECC, since TER value is smaller than $TER_{th}$. If the operating temperature goes beyond the threshold and consequently TER becomes more than $TER_{th}$, the correction penalty at read access will not only erode the gains of Lazy-ECC scheme, but also the overall performance will dramatically reduce, while the performance of conventional-ECC scheme is fixed under different operating temperatures. Figure 8 illustrates the performance behaviors under the temperature variation extracted by applying Lazy- and conventional-ECC on L1 caches. It is obvious that a pure ECC scheme (either Lazy-ECC or conventional-ECC) can not provide an optimal performance, hence, the hybrid scheme is employed. In other words, for a certain adopted $\Delta$ value, Lazy-ECC scheme provides better performance compared to the conventional one even with higher temperature. However, as the temperature reaches to a particular value (e.g., $80°C$ for Lazy-ECC with $\Delta$=30, as shown in Figure 8), we have to switch from Lazy-ECC to conventional-ECC to gain more performance. Otherwise, the overall performance of the system is significantly impacted. The value of such threshold temperature is a function of the $\Delta$ value and the adopted write margin.

### D. Case Study

As a case study, the implementation of the Lazy-ECC approach for an instruction cache of a processor is presented. As shown in Table I, the error detection latency is very small, which in the worst case is less than 500 ps (i.e., for

a 2GHz processor, the detection is done in one clock cycle). Once the processor reads an instruction from the instruction cache, the instruction is propagated to both the ECC decoder and the first stage of the pipeline structure of the processor. While the instruction is being processed in the processor, the decoder circuitry checks it for possible errors. When no error is detected, the execution continues without any interruption. However, if errors are detected, an error signal will be activated by the decoder circuitry to revert the erroneous instruction from the pipeline before it propagates to the next stage. For the error correction, the ECC decoder corrects the data while NOP instructions are inserted in the pipeline until the corrected data is written back to the memory. Once the correction process is finished, the processor re-fetches the corrected data from the cache and continues its operation.

## IV. Simulation Results

### A. Simulation Setup

Figure 9 illustrates our evaluation flow. For the bit-cell characterization, we run a cell-level analysis in SPICE, based on TSMC 65 nm general transistor models and the perpendicular STT-MRAM model presented in [23]. For the bit-cell in L1 cache, we used $\Delta = 30$ and $I = 3$ to guarantee fast write operations. For the L2 cache bit-cell, we assumed $\Delta = 35$ and $I = 2$ to guarantee high retention time. In order to compute the write latency, we model the stochastic behavior distribution using Matlab according to the methodology described in Section II. The encoder and decoder circuitries were synthesized with the TSMC 65nm standard cell library using Synopsys Design Compiler. The synthesis constraints are adjusted to obtain the minimum delay for encoding and decoding circuitry. For system-level analysis, the proposed Lazy-ECC as well as the conventional-ECC approach are implemented on L1 caches (both instruction and data) of Alpha processor modeled in gem5 [24]. gem5 is a cycle accurate simulator which provides accurate processor models and configurable memory systems with all related parameters such as capacity, latency, and associativity. We modify the gem5 memory system to accept different read and write latencies for each cache. These latencies are adjusted based on the results obtained by Matlab and NVSim. Table II summarizes the setup for our experiments.

### B. Performance Analysis

The impact of our approach is determined in two steps. First, we have to evaluate its error correction penalty for a real processor. This means, we need to assess the cost of a propagated error in the processor, which leads to pipeline flushing. We conducted a fault injection campaign on the instruction cache and estimated the performance penalty for 3000 randomly injected errors in this unit. Our results show that the average performance penalty of pipeline flushing is less than 4 cycles, because of a potential cache miss due to the pipeline flushing. In the second step, the employed workloads are executed on the processor using proposed and conventional approaches. The comparison is done based on the *Instruction Per Cycle* (IPC) metric which shows the overall performance.

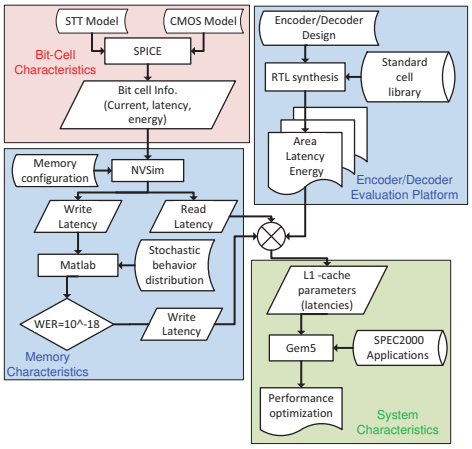



Fig. 9. Tool flow for evaluating results of proposed approach

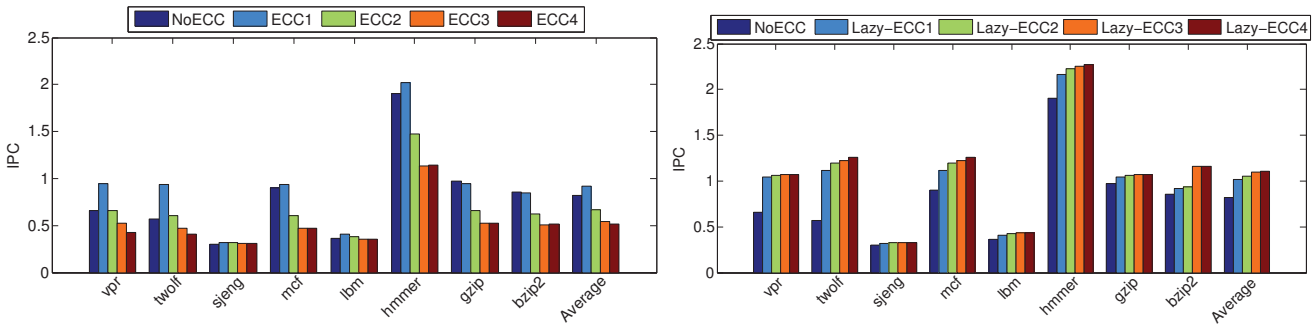
TABLE II. Simulation setup in gem5

| | |
|---|---|
| Processor | Single-core, 2 GHZ, Out-of-order, 4-issue |
| L1-cache (Data and Instruction) | 16 KB, 4-way set associative, 64B line size<br>STT-MRAM, $\Delta = 30$, $I = 3$<br>diff. read/write latencies obtained from Table I |
| L2-cache | 512 KB, 8-way set associative, 64B line size<br>STT-MRAM, $\Delta = 35$, $I = 2$<br>1.12ns/14.00ns read/write latencies |
| SPEC Applications | gzip, bzip2, mcf, twolf, vpr, lbm<br>hmmer, equake, sjeng |

Figure 10(a) illustrates the IPCs extracted by applying the conventional-ECC approach on the L1 caches of the processor. In the conventional approach, there is an enhancement of the overall system performance only for ECC1 which is on average 16%. However, by increasing ECC robustness, the IPC decreases compared to the case without ECC. This is mainly due to the impact of the long decoding latency. Results for Lazy-ECC are presented in Figure 10(b). As it is shown in the figure, the average IPC improvements of the error bit correction of 1,2,3 and 4 are around 28%, 33%, 38% and 39%, respectively. Unlike the conventional-ECC approach, in Lazy-ECC, IPC improves with the increase in the number of error correction bits. This is because of excluding the decoding latency from the read access. Not that the average performance gain of Lazy-ECC is highly dependent on the memory access rate. For instance, in an application with a low memory access rate (e.g., sjeng), there is no significant difference between the efficiency of Lazy-ECC and the conventional-ECC approach. In contrast, the high memory access rate in some other applications (e.g., vpr and twolf) results in a huge gain in performance compared to the conventional-ECC approach.

### C. Related Work

A First-In First-Out (FIFO) based technique is proposed in [25], which is based on the decoupling of error detection and error correction for conventional SRAM/DRAM memories. However, in this technique once an error is detected, the entire system is halted. This results in a considerable performance penalty. In STT-based memory technology, where we expect a high error rate due to the reduction of the write margin and the thermal stability factor, the performance penalty will be much higher. Furthermore in this technique, the data revived on the system inputs during error recovery process is stored in a FIFO structure to guarantee the correct functionality of the system. Consequently, this imposes significant area and power overheads to the real system, since the depth of the FIFO depends mainly on the number of the system inputs and on the number of cycles required for the error correction. In our experimental setup, correcting error and writing error-free data to the memory require 10 (i.e., 4.71 ns) and 9 cycles (i.e., 4.17 ns) (see Table I), respectively. This means that the depth of the FIFO at each input is at least 19 bits. Since




(a) Conventional-ECC

(b) Lazy-ECC

Fig. 10. Performance of the proposed Lazy-ECC and conventional-ECC approach for various workloads of SPEC benchmarks

the number of inputs in real processors is relatively large (e.g., 280 inputs in Alpha 7), the area overhead associated with this technique would be significantly higher than that of our proposed solutions (5320 flip-flops vs. few combinational gates). Moreover, the main shortcoming of FIFO-based technique is that it fails to capture all system inputs when errors are occurring in consecutive cycles. Basically, the FIFO could be fulfilled during the error correction phase, and hence, there should be some gap between two errors to be sure that the FIFO is empty again. Otherwise, some data appearing in processor inputs during the error correction time could be lost resulting in failure. Therefore, in applications with high error rate, our proposed technique provides significantly higher reliability than the FIFO-based technique. According to the results of our experiments, the probability of having write access in each cycle is 0.08. In addition, the probability of having a write error for the case of ECC4 which provides a significant performance improvement is $6.6\times10^{-4}$. Hence, the probability of having write error in each cycle is $5.3 \times 10^{-5}$. According to these, the probability of having the second write error in a timing window of 19 cycles after first write error could be computed according to exponential distribution to be $1 - e^{-\lambda \times cycle}$ where $\lambda$ is $5.3 \times 10^{-5}$ and $cycle$ is equal to 19. This probability is equal to 0.001 and this means that the probability of having two errors in the timing of 19 cycles is $5.3 \times 10^{-5} \times 0.001 = 5.3 \times 10^{-8}$ which is significantly larger than the target WER of $10^{-18}$. Although the failure rate for FIFO-based technique could be mitigated by increasing the depth of the FIFOs, this further increases the area overhead imposed by this technique.

## V. Conclusions

The extensive write margin due to the inherent stochastic write behavior limits the application of STT-MRAM in high performance memories such as low level caches. The write margin of STT-MRAM can be reduced to a large extent by exploiting robust ECCs with unfinished write, however, such techniques impair the read access due to high decoding latency. In this paper, we presented an opportunistic write technique equipped with Lazy-ECC approach to exclude the decoding latency from read access by performing speculative computation based on unchecked data. We have also provided a hybrid adaptation approach to address temperature-induced retention failures at runtime. The proposed approach was implemented on the L1-caches of a high-performance processor. The simulation results reveal that this approach significantly improves the performance by up to 39% compared to the conventional approach with negligible area overhead in the processor.

## VI. Acknowledgement

This work was partly supported by the European Commission under the Horizon-2020 Program as part of the GREAT project (http://www.great-research.eu/) and by ANR/DFG as part of the MASTA project.

## References

[1] M.-T. Chang, et al. Technology comparison for large last-level caches. In *HPCA*, pages 143–154, 2013.
[2] S. Mittal. A survey of architectural techniques for improving cache power efficiency. *Sustainable Computing: Informatics and Systems*, 4(1):33–43, 2014.
[3] S. A. Wolf, et al. The promise of nanomagnetics and spintronics for future logic and universal memory. *Proceedings of the IEEE*, 98(12):2155–2168, 2010.
[4] J. Ahn, et al. Dasca: Dead write prediction assisted STT-MRAM cache architecture. In *HPCA*, pages 25–36, 2014.
[5] K.-W. Kwon, et al. Aware (asymmetric write architecture with redundant blocks): A high write speed STT-MRAM cache architecture. *TVLSI*, 22(4):712–720, 2014.
[6] P. Zhou, et al. Energy reduction for STT-RAM using early write termination. In *ICCAD*, pages 264–268, 2009.
[7] R. Bishnoi, et al. Avoiding unnecessary write operations in STT-MRAM for low power implementation. In *ISQED*, pages 548–553, 2014.
[8] T. Zheng, et al. Variable-energy write STT-RAM architecture with bit-wise write-completion monitoring. In *ISLPED*, pages 229–234, 2013.
[9] D. Suzuki, et al. Cost-Efficient Self-Terminated Write Driver for Spin-Transfer-Torque RAM and Logic. *Magnetics*, 50(11):1–4, 2014.
[10] R. Bishnoi, et al. Self-Timed Read and Write Operations in STT-MRAM. *TVLSI*, 24(5):1783–1793, May 2016.
[11] B. Del Bel, et al. Improving STT-MRAM density through multibit error correction. In *DATE*, pages 1–6, March 2014.
[12] X. Bi, et al. Probabilistic design methodology to improve run-time stability and performance of stt-ram caches. In *ICCAD*, pages 88–94, 2012.
[13] Y. Emre, et al. Enhancing the reliability of STT-RAM through circuit and system level techniques. In *SiPS*, pages 125–130, 2012.
[14] D. Strukov. The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories. In *ACSSC*, pages 1183–1187, 2006.
[15] A. Driskill-Smith, et al. Latest advances and roadmap for in-plane and perpendicular STT-RAM. In *IMW*, 2011.
[16] X. Fong, et al. Bit-cell level optimization for non-volatile memories using MTJ and STT switching. *Nanotechnology*, 11(1):172–181, 2012.
[17] E. I. Vatajelu, et al. Power-aware voltage tuning for STT-MRAM reliability. In *ETS*, pages 1–6, 2015.
[18] W. Zhao, et al. Failure analysis in magnetic tunnel junction nanopillar with interfacial perpendicular magnetic anisotropy. *Materials*, 9(1):41, 2016.
[19] D.Apalkov, et al. Spin-transfer torque magnetic random access memory (STT-MRAM). *JETC*, 9(2):13, 2013.
[20] W. Kang, et al. Reconfigurable codesign of STT-MRAM under process variations in deeply scaled technology. *TED*, pages 1769–1777, 2015.
[21] K. Munira, et al. A Quasi-analytical model for energy-delay-reliability tradeoff studies during write operations in STT-RAM. *Electron Devices*, 59(8), 2012.
[22] W. Wen, et al. CD-ECC: Content-dependent error correction codes for combating asymmetric nonvolatile memory operation errors. In *ICCAD*, pages 1–8, Nov 2013.
[23] A. Mejdoubi, et al. A compact model of precessional spin-transfer switching for MTJ with a perpendicular polarizer. In *MIEL*, pages 225–228, 2012.
[24] N. Binkert, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.
[25] M. Nicolaidis, et al. Eliminating speed penalty in ECC protected memories. In *DATE*, pages 1–6, March 2011.