High-Performance Energy-Efficient and Reliable Design of Spin-Transfer Torque Magnetic Memory

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Nour Sayed

aus Edlib, Syrien

Tag der mündlichen Prüfung: 19.07.2019

Erster Gutachter: Prof. Dr.-Ing. Mehdi B. Tahoori
Karlsruher Instituts für Technologie (KIT); Germany

Zweiter Gutachter: Prof. Dr.-Ing. Lionel Torres
University of Montpellier; France

Dedication

To whom, who taught me my first letters...
To whom, who planted in me the love of learning and discovering...
To whom, who opened my eyes to the unknown world and guided me to discover...
To whom, God ordered me to acknowledge and respect them...
To my parents ... I love you too much

***

To whom, who change my life...
To whom, who taught me the patience...
To whom, who loved me, helped me, encouraged me and was always beside me...
To my husband ... I love you

***

To whom, who bring the happiness into my world...
To whom, who make me feel the responsibility...
To whom, who make me look into the future...
To my children ... I love you

***

To whom, who are always in my heart...
To whom, who I can always depend on...
To whom, who are part of my childhood and memories...
To the best brother and sisters ... I love you

***

Acknowledgment

My cordial gratitude is presented to my advisor Prof. Dr. Mehdi Tahoori, this great person from whom I learned a lot, who guided me through my research and give me the flexibility to try and discover, who unleashed my creativity to deal with the most challenging tasks ... I'm very thankful.

I'm also grateful to my co-advisor Prof. Dr. Lionel Torres for the nice and fruitful collaboration and for his constructive and encouraging advices, which highly motivated me especially during defense. Furthermore, my special thanks to Dr. Rajendra Bishnoi for the fruitful collaboration.

I would like to present my heartfelt thanks to all of my colleagues at the Chair of Dependable Nano Computing (CDNC) for the nice period I spent between them and the family atmosphere they have created at our chair and also for all their valuable feedback and discussions. In particular I want to thank: Sarath Mohanachandran Nair, Dr. Mojtaba Ebrahimi, Dr. Fabian Oboril, Dr. Saman Kiamehr, Dr. Mohammad Saber Golanbari, Dr. Anteneh Gebregiorgis, Dr. Arunkumar Vijayan, Dennis Gnad, Farhan Rasheed, Ahmet Turan Erozan, Dennis Weller, Christopher Münch and Jonas Krautter. I also want to thank Longfei Mao, whom I supervised in the scope of his Master thesis for his nice work. Finally, my thanks to all the secretaries and technicians at our institute in particular to Mrs. Iris Schrder-Piepka for all of her efforts in providing a good working environment.

List of First-Author Publications

[1] **Nour Sayed**, Longfei Mao, Rajendra Bishnoi and Mehdi B. Tahoori. Compiler-Assisted and Profiling-Based Analysis for Fast and Efficient STT- MRAM On-Chip Cache Design. ACM Transaction on Design Automation of Electronic Systems **(TODAES)**, 2019.

[2] **Nour Sayed**, Rajendra Bishnoi and Mehdi B. Tahoori. Fast and Reliable STT-MRAM Using Non-Uniform and Adaptive Error Detecting and Correcting Scheme. IEEE Transactions on Very Large Scale Integration **(VLSI)** Systems, pages 1-14, 2019.

[3] **Nour Sayed**, Rajendra Bishnoi, Fabian Oboril and Mehdi B Tahoori. A Cross-Layer Adaptive Approach for Performance and Power Optimization in STT-MRAM. In 2018 Design, Automation & Test in Europe Conference & Exhibition **(DATE)**, pages 791-796. IEEE, 2018.

[4] **Nour Sayed**, Sarath Mohanachandran Nair, Rajendra Bishnoi and Mehdi B Tahoori. Process Variation and Temperature Aware Adaptive Scrubbing for Retention Failures in STT-MRAM. In Proceedings of the 23rd Asia and South Pacific Design Automation Conference **(SAP-DAC)**, pages 203–208. IEEE Press, 2018.

[5] **Nour Sayed**, Fabian Oboril, Azadeh Shirvanian, Rajendra Bishnoi and Mehdi B. Tahoori. Exploiting STT-MRAM for Approximate Computing. In 2017 22nd IEEE European Test Symposium **(ETS)**, pages 1-6. IEEE, 2017.

[6] **Nour Sayed**, Mojtaba Ebrahimi, Rajendra Bishnoi and Mehdi B Tahoori. Opportunistic Write for Fast and Reliable STT-MRAM. In Proceedings of the Conference Design, Automation & Test in Europe Conference & Exhibition **(DATE)**, pages 554-559. European Design and Automation Association, 2017.

[7] **Nour Sayed**, Fabian Oboril, Rajendra Bishnoi and Mehdi B Tahoori. Leveraging Systematic Unidirectional Error Detecting Codes for Fast STT-MRAM Cache. In 2017 IEEE 35th VLSI Test Symposium **(VTS)**, pages 1-6. IEEE, 2017.

List of Co-Author Publications

[8] Mohammad Saber Golanbari, **Nour Sayed**, Mojtaba Ebrahimi, Mohammad Hadi Moshrefpour Esfahany, Saman Kiamehr and Mehdi B. Tahoori. Aging-aware Coding Scheme for Memory Arrays. In 2017 22nd IEEE European Test Symposium **(ETS)**, pages 1-6. IEEE, 2017.

[9] Mojtaba Ebrahimi, **Nour Sayed**, Maryam Rashvand and Mehdi B. Tahoori. Fault Injection Acceleration by Architectural Importance Sampling. 2015 International Conference on Hardware/Software Codesign and System Synthesis **(CODES+ISSS)**, Amsterdam, 2015, pp. 212-219.

Supervised Master Thesis that Partially Contributed to the Simulation

Longfei Mao: STT-MRAM Performance- and Energy-Efficient Design Based-on Dynamic Behavior Predictions, 2018.

Abstract

Nowadays, computing systems are indispensable in our daily life, because of their deployment in different application areas starting from small embedded electronic control units, like a small control processor in the automotive, health-care and smart devices, to high-performance supercomputers designed for advanced software applications like artificial intelligence, climate modeling, and cryptographic analysis, etc. This achievement has been done due to the continuous shrinking of the transistor dimensions guided by Moore's Law. However, chip power dissipation issue due to the dynamic and static power consumption has become a major design constraint for further technology down-scaling, in particular for modern integrated circuits designed for low-power applications in the emerging "Internet of Things" (IOT), automotive or personalized health-care devices. In these circuits, the most part of the energy consumption comes from on-chip memory, primarily due to the static power. Consequently, the traditional on-chip memory technologies, such as SRAM (Static Random Access Memory) and DRAM (Dynamic Random Access Memory), cannot target the strict energy budget, because of the high leakage. Semiconductor industry is actively searching for alternative memory technologies, including non-volatile memories with zero leakage power for the bit-cell. Moreover, the data can be retained in the non-volatile memories without any power supply. This in turn minimizes the static power sharply and paves the way towards normally-off/instant-on computing.

Among the emerging non-volatile memory technologies, *Spin Transfer Torque Magnetic Random Access Memory* (STT-MRAM) is one of the most promising candidates for future computing systems. As this technology is based on a magnetic device (called *Magnetic Tunnel Junction* or simply MTJ) that exploits the spin of electrons to store digital content as resistance states. STT-MRAM promises zero static power, fast access latencies (almost as fast as SRAM) and high integration density (as good as DRAM). It also provides high endurance, high density, CMOS compatibility and scalability. However, in order to employ STT-MRAM as the promising low-power on-chip memory solution, several issues related to this emerging technology have to be addressed. Excessive write operation costs and variety of reliability issues are the major roadblocks for adopting this technology. The write operation costs are because of the high write current required to flow for a long time. On the other hand, under reliability issues, there are read disturb (erroneous writing while reading), read and write failure, dielectric breakdown, and retention failure (digital content switching during the idle time). The aforementioned issues are further exacerbated due to the process variations and temperature effects.

In this dissertation new computing paradigms, architectures and design philosophy are proposed and evaluated for adopting the STT-MRAM technology as highly reliable, energy efficient and fast memory. For this purpose, a novel cross-layer framework from the cell-level all the way up to the system- and application-level has been developed. In these framework, the reliability issues are modeled accurately with appropriate fault models at different abstraction levels in order to analyze the overall failure rates of the entire memory and its *Mean Time To Failure* (MTTF) along with considering the temperature and process variation effects. Design-time, compile-time and run-time solutions have been provided to address the challenges associated with STT-MRAM. The effectiveness of the proposed solutions is demonstrated in extensive experiments that show significant improvements in comparison to state-of-the-art solutions, i.e. lower-power, higher-performance and more reliable STT-MRAM design.

- For energy-efficient STT-MRAM design:
    1. Multi-retention capabilities (i.e., standard and relaxed array regions) and multi-level write currents (i.e., high, medium and low) have been proposed to be adopted at memory architecture-level. The switching between regions or levels is guided according to the application behavior (i.e., performance and reliability needs). Static and dynamic prediction schemes of the application behavior are proposed with very high achieved accuracy.
    2. Relaxed retention and write parameters have been suggested at device and circuit levels, respectively. This is along with leveraging the approximate computing techniques for error tolerance at application-level.
- For high-performance STT-MRAM design:
    1. Fast write termination supported with an adaptive error correcting code is proposed, where the unfinished bits will be processed later at next access by an adaptive error correcting code (either Lazy-ECC or Conventional-ECC). The error correcting scheme is defined each access based on the system-level error probability, which is highly affected by the operating temperature variation.
    2. Adjusting system-performance according to the application-level requirements is achieved by on-the-fly write current scaling and controlling scheme.

- For reliable STT-MRAM design:

   1. Systematic Unidirectional Error-Detecting Code (SEDC) has been efficiently adopted instead of conventional error code. This is because of the SEDC code ability to leverage of the asymmetric nature of errors in STT-MRAM.
   2. Adaptive and multi-scrubbing level scheme is proposed to address the excessive performance and energy overheads of the conventional scrubbing scheme.

ZUSAMMENFASSUNG

Heutzutage sind Computersysteme in unserem täglichen Leben unverzichtbar, da sie in verschiedenen Anwendungsbereichen eingesetzt werden, angefangen bei kleinen eingebetteten elektronischen Steuergeräten wie Automobil, Gesundheitswesen und intelligenten Geräten bis hin zu Hochleistungs-Supercomputern, die für fortschrittliche Software wie künstliche Intelligenz, Klimamodellierung und kryptographische Analyse usw. entwickelt wurden. Diese Leistung wurde durch die kontinuierliche Schrumpfung der Transistorabmessungen nach dem Moore'schen Gesetz erreicht. Das Problem der Chip-Leistungsaufnahme aufgrund des dynamischen und statischen Stromverbrauchs ist jedoch zu einem wesentlichen Designeinschränkungsmerkmal für die weitere Verkleinerung der Technologie geworden, insbesondere für moderne integrierte Schaltungen, die für Anwendungen mit geringer Leistung im aufkommenden "Internet der Dinge" (IOT), im Automobil oder im personalisierten Gesundheitswesen entwickelt wurden. In diesen Schaltungen stammt der größte Teil des Energieverbrauchs aus dem On-Chip-Speicher, vor allem aufgrund der statischen Leistung. Folglich können die traditionellen On-Chip-Speichertechnologien wie SRAM (Static Random Access Memory) und DRAM (Dynamic Random Access Memory) aufgrund der hohen Leckage nicht auf das strenge Energiebudget ausgerichtet werden. Darüber hinaus können die Daten ohne Stromversorgung aufbewahrt werden. Dies wiederum minimiert die statische Leistung stark und ebnet den Weg zu normal-off/instant-on computing.

Unter den aufkommenden nichtflüchtigen Speichertechnologien ist der Spin Transfer Torque Magnetic Random Access Memory (STT-MRAM) einer der vielversprechendsten Kandidaten für zukünftige Computersysteme. Da diese Technologie auf einem magnetischen Gerät (Magnetic Tunnel Junction oder einfach MTJ genannt) basiert, das den Spin von Elektronen nutzt, um digitale Inhalte als Widerstandszustände zu speichern. STT-MRAM verspricht null statische Leistung, schnelle Zugriffslatenzen (fast so schnell wie SRAM) und hohe Integrationsdichte (so gut wie DRAM). Es bietet auch eine hohe Ausdauer, hohe Dichte, CMOS-Kompatibilität und Skalierbarkeit. Um STT-MRAM als vielversprechende Low-Power-On-Chip-Speicherlösung einzusetzen, müssen jedoch einige Probleme im Zusammenhang mit dieser neuen Technologie gelöst werden.

Die überhöhten Schreiboperationskosten und eine Vielzahl von Zuverlässigkeitsproblemen sind die wichtigsten Hindernisse für die Einführung dieser Technologie. Die Kosten für den Schreibbetrieb sind auf den hohen Schreibstrom zurückzuführen, der benötigt wird, um über einen längeren Zeitraum zu fließen. Unter Zuverlässigkeitsproblemen gibt es Lesestörungen (fehlerhaftes Schreiben beim Lesen), Lese- und Schreibfehler, dielektrischer Durchschlag und Rückhaltefehler (Umschalten digitaler Inhalte während der Leerlaufzeit). Die oben genannten Probleme werden durch die Prozessschwankungen und Temperatureinflüsse noch verschärft.

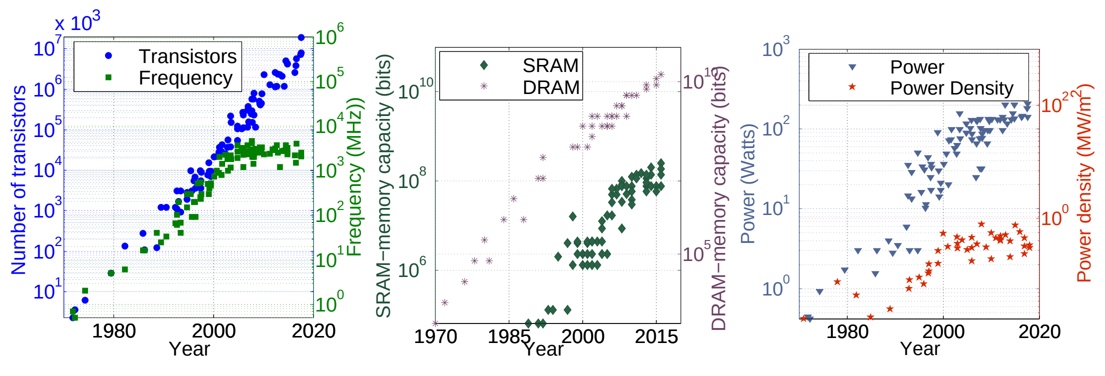
In dieser Dissertation werden neue Rechenparadigmen, Architekturen und Designphilosophien vorgeschlagen und bewertet, um die STT-MRAM-Technologie als hochzuverlässigen, energieeffizienten und schnellen Speicher einzusetzen. Zu diesem Zweck wurde ein neuartiges Cross-Layer-Framework von der Zellenebene bis hin zur System- und Anwendungsebene entwickelt. In diesem Rahmen werden die Zuverlässigkeitsprobleme mit geeigneten Fehlermodellen auf verschiedenen Abstraktionsebenen genau modelliert, um die Gesamtausfallraten des gesamten Speichers und seiner Mean Time To Failure (MTTF) unter Berücksichtigung der Auswirkungen von Temperatur- und Prozessschwankungen zu analysieren. Um den Herausforderungen im Zusammenhang mit STT-MRAM gerecht zu werden, wurden Design-, Compile- und Runtime-Lösungen bereitgestellt. Die Wirksamkeit der vorgeschlagenen Lösungen wird in umfangreichen Experimenten demonstriert, die signifikante Verbesserungen gegenüber modernen Lösungen zeigen, d.h. leistungsschwächeres, leistungsfähigeres und zuverlässigeres STT-MRAM-Design. Die Halbleiterindustrie sucht aktiv nach alternativen Speichertechnologien, einschließlich nichtflüchtiger Speicher mit leckagefreier Leistung für die Bitzelle.

Contents

Chapter 1

Introduction

The innovation in the computing system has been powered for the past five consecutive decades by Moore's law [1], which predicts an exponential growth in the MOSFET transistor density per unit area of each CMOS technology generation due to shrinking the geometry roughly by a half every 18 months. As a result, system functionality, memory size, storage capacity and the processor performance have been steadily optimized at the same rate [2–4] (Figure 1.1(a) and (b)). However, the energy gain is not in proportion with the dimension scaling factor of the technology generation, as for a given chip area the power dissipation remains constant. Furthermore, transistor lengths scaling increases the leakage currents exponentially, thereby making the static power a major design constraint (Figure 1.1(c)). Therefore, Dennard's law states to scale down both voltage and current of the transistor along with its physical dimension [5]. The combination of Moor's law and Dennard scaling leads to reduce the assumed factor of the performance gain significantly, which is expected to find a limit soon (Figure 1.1(a)).



(a) (b) (c)

Figure 1.1: Moore's law [6]: (a) Evolution of transistor count of Integrated Circuit (IC) chips ITRS scaling trends of IC's performance; (b) scaling trends of memory capacity and (c) scaling trends of power density.

1.1 Memory Wall Issue

The computing system of the conventional *von Neumann architecture* is composed mainly of a computational part (processor), a storage part (memory) and communication buses (data and address buses). The processor performance is tightly bound to the memory performance (memory bandwidth), as the data and instructions of the executed programs are stored in a limited memory. On the other hand, the exponential performance improvement in processor exceeds that in memory, because of the significant difference in the adopted semiconductor technologies in computational circuit and storage cells [7]. This gap of performance seen in Figure 1.2 has further increased due to technology down-scaling, leading to hit a wall in the performance improvement of the entire computing system, referred to as *"Memory Bandwidth"* or *"Memory Wall"* problem [8–13]. The *"On-Chip Memory"* is the only way to defend against *Memory Wall* problem because of improving the memory performance by reducing the average time (t_{avg}) required to access the memory [14], where t_c and t_c are on-chip and off-chip memory (cache and main memory) access time and p_{hit} is the cache hit probability.

$$t_{avg} = p_{hit} \times t_C + (1 - p_{hit}) \times t_M \tag{1.1}$$

By reducing t_C to match the CPU clock cycle and increasing the hit capacity in the cache, the memory speed scales with that of processor.

$$\lim_{\{t_C, p_{hit}\} \to \{1cycle, 1\}} t_{avg} = CPUSpeed \tag{1.2}$$

Single level of cache is not enough to bridge the gap between processor and memory, since the larger the cache, the larger its access time, and the smaller the cache, the smaller its hit probability. Therefore, *Multi-level cache hierarchy* (two or more levels) is the best potential solution to obtain the best system performance [8, 15, 16].

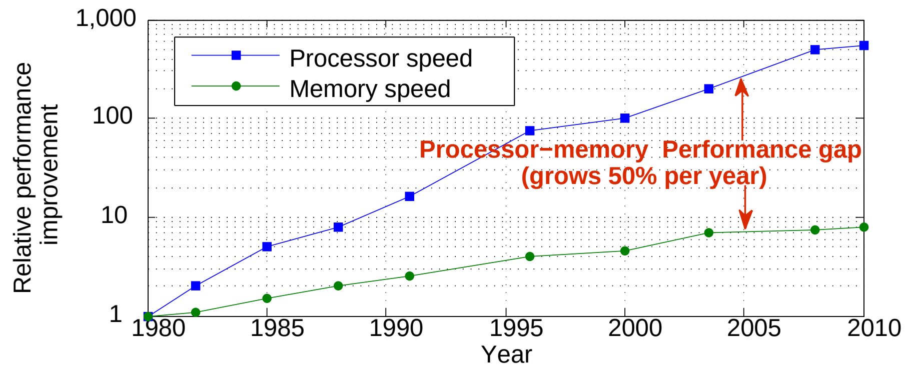


Figure 1.2: Performance gap between memory and processor due to Moore's law impact [8]

1.2 Power Wall Issue

Although the achieved performance improvement in the computing system that has been done due to the continuous shrinking of the transistor dimensions guided by Moor's Law, the power consumption, consisting of the dynamic and static power consumption, has become a major design constraint, referred to as *" Power Wall"* problem [17], as the power density per unit area stays constant. In fact, this problem is originally the result of the fact that the static power consumption of each transistor hardly becomes smaller with the technology down-scaling than the dynamic power [18]. The effects and consequences of the power wall problem are exacerbated in particular for modern integrated circuits designed for low-power applications in the emerging “Internet of Things” (IoT), automotive or personalized health-care. In such applications, the energy consumption, primarily due to the static power, limits their operation time [19, 20].

Power Wall has been partially addressed at device-level with voltage scaling at the cost of lower performance and reliability. However, at application-level, as the total energy is equal to the integral of the power dissipation over execution time, total energy consumption at lower voltage will be reduced only if the decrease in the power dissipation is lager than the increase in the execution time. Thus, *Dynamic Voltage and Frequency Scaling* (DVFS) policies have been explored to adjust the trade-off levels between performance and power with respect to the application requirements [21]. At system-level the total energy consumption is the sum of the dissipated energy in each active or non-active component, while the performance is tightly bound to only the bottleneck and active components. Therefore, for in-active components *Power Gating* PG scheme suppresses the dynamic and static power significantly [22].

1.3 Why STT-MRAM On-Chip/Off-Chip Memory?

Addressing the memory-wall problem in the recent computing system requires more memory capacity and higher cache hierarchy. This in turn makes the memory system the dominant part of the system. A typical memory hierarchy or storage pyramid consists of a multi-level of different technologies, which are accessed by different parts of the system [23]. At the top, the *register file*, which is accessed very fast and directly by the processor during the program execution, is used to store program parameters values. In the next level, the *on-chip cache hierarchy* is typically integrated directly into the processor chip as an invisible memory to the programmer to store the program instructions as a sub-set of the most frequently accessed data (memory elements). System performance is tightly bound to the cache performance [8, 24, 25], which is more important than the cache size. The next level is the *off-chip main memory*, which is not part of the processor. It contains the working set size (i.e., a full copy of the executed programs) and is accessed only in case of cache miss or replacement.

File register, cache levels and main memory, referred to as *primary memory* and used to store active items of data which are required for the current applications, are implemented based on SRAM and DRAM technologies (i.e., semiconductor technologies), which are volatile memories and retain data while powered. However, volatile memory consumes fairly large static power due to leakage current compared to its low access energy. PG is a promising way to address the leakage current issue, however it requires a power management, referred to as *Normally-Off Computing* [22], in order to guarantee the data integrity in the storage elements (memory) of the powered-off components with minimum overheads. The traditional volatile technology used in the on-chip memory is SRAM provides fast read and write performance despite its relatively large cell size compared to other technologies. One of the proposed solutions is to save the content of the SRAM cell in an external non-volatile memory for every power gating cycle. This minimizes the power gains obtained by PG. A full usage of PG requires *Non-volatile* technology for on-chip memory that provides the ability to retain the context locally, [26, 27].

On the other hand, SRAM technology provides around 8× lower capacity for a given area compared to DRAM. However, DRAM performance is not sufficient for processors, as it is limited by the charging and dis-charging processes of the storage device in DRAM cell (i.e., an electric capacitor). Furthermore, the charge of capacitor is leaky and requires a periodic refresh. Typically, every 1 ms, all cells contents must be read and written again to avoid data loss. This makes the energy consumption for the refreshing process dominant in DRAM technology. Moreover, the processor is not able to access DRAM array while it is being refreshed, which reduces system performance significantly.

The electric charge and current leakage phenomenon of DRAM and SRAM technologies, respectively, raise the need of replacing the current volatile on-chip and off-chip memories with a *Non-Volatile Memory* (NVM), which firstly retains data even while the device is in power-off mode, and secondly provides zero leakage during the active mode, while satisfying SRAM performance and DRAM density [26, 28–30].

New generation of NVM technologies, such as *Phace-Change Memory* (PCM) [31–33], *Resistive Random Access Memory* (RRAM) [34–36] and *Magnetic Random Access Memory* (MRAM) [37–39], which are byte-addressable, open the way toward a new memory system, which presents prominent opportunities in favor of low power design, as they are not leaky and don not require power to retain the data. Among these emerging NVM technologies, STT-MRAM is the most promising one to be adopted as a universal technology in the memory hierarchy [40, 41]. As this technology is based on a magnetic device (called *Magnetic Tunnel Junction* or simply MTJ that exploits not only the charge of electrons but also their spin to store a digital data as resistance states. Thus, this storage mechanism provides zero leakage. MTJ consists of two ferromagnetic layers separated by a thin oxide layer. The magnetic orientations of this device specifies the resistance state as low or high to represent logic ‘0’ or ‘1’, respectively. STT-MRAM has several beneficial features such as fast access latencies (comparable to SRAM), high density (as good as DRAM), CMOS compatible, high endurance and scalability.

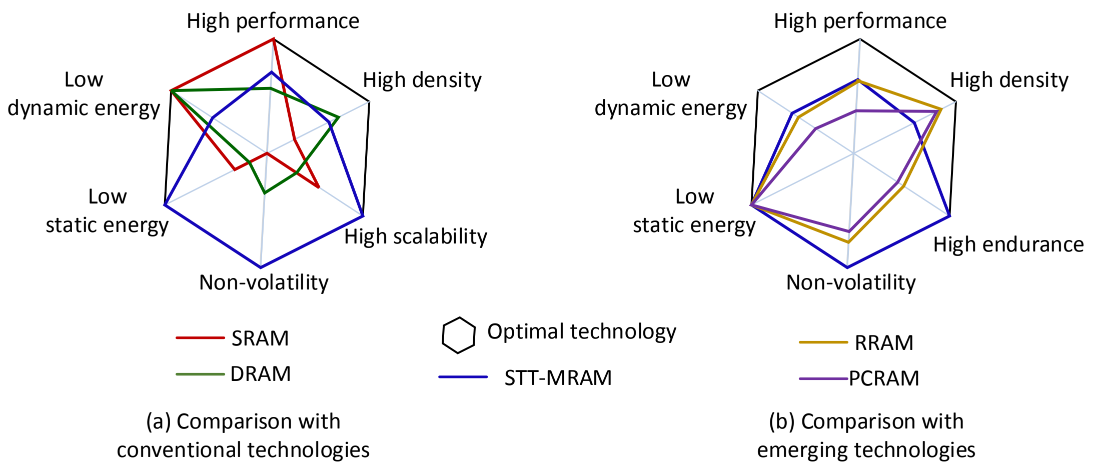


Figure 1.3: Comparison of STT-MRAM technology with conventional and emerging memory technologies

Next figure illustrates the comparison of STT-MRAM technology with respect to the convention volatile (Figure 1.3(a)) and other emerging non-volatile (Figure 1.3(b)) memory technologies [42–45]. However, adopting STT-MRAM as a promising replacement of DRAM and SRAM in the cache and memory system design suffers from several issues related to the reliability and write operation [40, 41, 46–48].

1.4 Key Challenges

In STT-MRAM, the switching behavior of the cell contents is inherently stochastic [49, 50]. Thereby, it can be classified into desired switching at write access and undesired or erroneous one at read access or during the retention time.

The desired switching of the write process is non-deterministic even under the same environmental and operating conditions [51, 52]. This phenomena is because of two main reasons: i) the spin-transfer efficiency factor of the free layer for switching its magnetic orientation to the same or the opposite spin direction of the reference layer, and ii) the impact of the voltage degradation of the access transistor, which reduces the current density. However, this stochasticity can be converted into a deterministic probability, which should by very low, by specifying high write current required to flow for a long write pulse [53, 54]. This in turn challenges the replacement of SRAM technology with STT-MRAM for on-chip memories, as STT-MRAM consumes higher write energy with slower write access than SRAM.

On the other hand, the random erroneous switching at read access or even during the retention time is due to unavoidable thermal fluctuations of the MTJ device, which is further exacerbated due to the process variations and temperature effects, resulting in a variety of reliability issues [55–58] such as read disturb (erroneous writing while reading), read and write failure, dielectric breakdown, and retention failure (digital content switching during the idle time).

The major challenges in employing the STT-MRAM technology as a universal technology in the memory hierarchy is to reduce the on-it write latency and energy required for on-chip memory along with satisfying reliability constraints [46].

Table 1.1: Scope of contributions

Contribution	Performance improvement	Energy reduction
Lazy-ECC scheme [59]	39%	29%
Non-uniform & Adaptive ECC scheme [60]	23.5%	24%
SEDC scheme with write through policy [61]	As SRAM	12%
Adaptive write current mechanism [62]	15.5%	1.7%
Approximate computing for STT-MRAM [63]	25%	70%
Compile-time data mapping approach [64]	9%	49.7%
Adaptive scrubbing mechanism [65]	Eliminate scrubbing overheads	

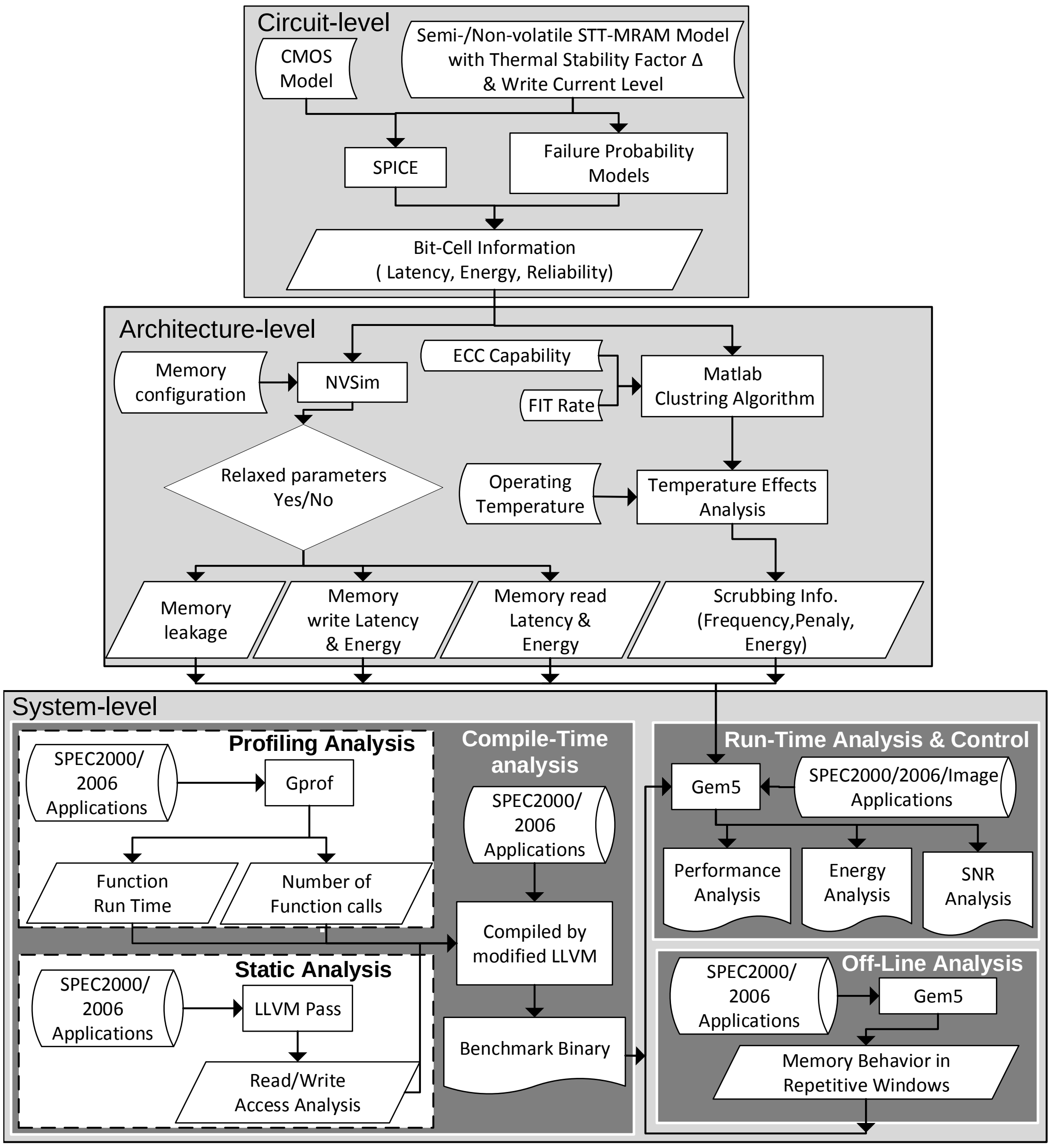

Circuit-level
CMOS Model
Semi-/Non-volatile STT-MRAM Model with Thermal Stability Factor Δ & Write Current Level
SPICE
Failure Probability Models
Bit-Cell Information (Latency, Energy, Reliability)
Architecture-level
Memory configuration
NVSim
ECC Capability
FIT Rate
Matlab Clustring Algorithm
Relaxed parameters Yes/No
Operating Temperature
Temperature Effects Analysis
Memory leakage
Memory write Latency & Energy
Memory read Latency & Energy
Scrubbing Info. (Frequency,Penaly, Energy)
System-level
Profiling Analysis
SPEC2000/ 2006 Applications
Gprof
Function Run Time
Number of Function calls
Static Analysis
LLVM Pass
Read/Write Access Analysis
Compile-Time analysis
Compiled by modified LLVM
Benchmark Binary
Run-Time Analysis & Control
Gem5
SPEC2000/2006/Image Applications
Performance Analysis
Energy Analysis
SNR Analysis
Off-Line Analysis
Memory Behavior in Repetitive Windows

Figure 1.4: A cross-layer design framework

1.5 Dissertation Contributions

This dissertation makes the following contributions, which can be classified into three main categories, in order to address the aforementioned challenges associated with STT-MRAM based memory system, specifically, for on-chip memories, 1) optimizing performance under reliability constraints, 2) minimizing energy under reliability constraints, and 3) minimizing overheads of satisfying target reliability constraints.

The contribution scope of this work, which is briefly illustrated in Table 1.1 with summarized results, have been achieved by developing a novel cross-layer framework from the cell-level all the way up to the system- and application- level, as demonstrated in Figure 1.4.

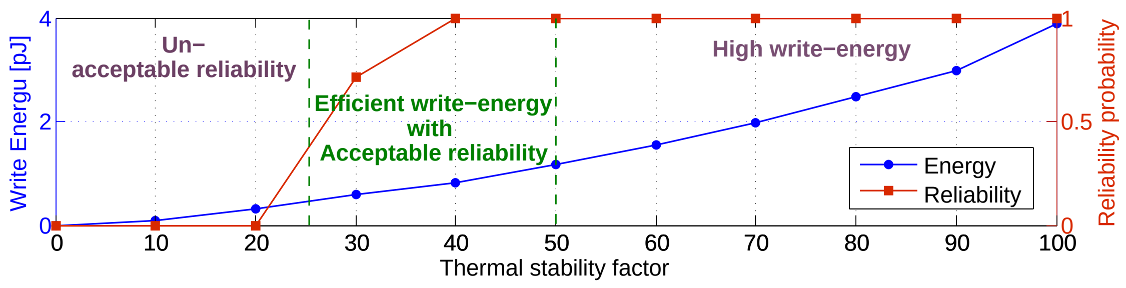


Figure 1.5: Energy and reliability trade-off of STT-MRAM design

1.5.1 Minimizing energy under reliability constraints

STT-MRAM adopts the *Spin-transfer torque switching* as a write mechanism in the MTJ device. Therefore, high write current is required in order to inject enough spin-polarized current in the free layer of the MTJ device. Consequently, write operation in STT-MRAM technology dissipates energy aggressively, which makes it not suitable for memories accessed with high write rate, like on-chip memories.

The minimum current required for the switching decreases as the STT-MRAM size shrinks. This results in reducing the thermal stability factor, and hence relaxing non-volatility condition. In fact the standard non-volatile STT-MRAM technology is designed with extremely long retention time (i.e., for more than 10 years), which meets the high-level memory requirements, while for on-chip memories, a small retention time (a few ms) is sufficient to retain the data [48]. Therefore, semi-volatile STT-MRAM technology with relaxed retention time can be used for such caches with much lower write energy and even shorter write pulse, as the data in on-chip caches requires shorter life-time. However, this approach increases the rate of the retention failures as well as the chances of the cell content being flipped during the read operation [66]. Figure 1.5 illustrates the reliability-energy trade-off along with thermal stability factor scaling.

A hybrid STT-MRAM on-chip memory design is very useful to prevent the data loss or distorted along with achieving low-power and fast write operation, however, to explore the advantage of the hybrid cache, data management policy has to be designed accurately based on the data behavior, which reflects the performance and reliability requirements of the application. The key idea is to create a cache hierarchy with two different regions: a write region implemented by semi-volatile STT-MRAM, which has low write energy and latency, and a read region with non-volatile STT-MRAM, which protects the data against read and retention errors. Static and dynamic data management approaches are introduced in this thesis.

In the static approach, data mapping decisions are made at compile-time, based on the application requirements (i.e., execution time and memory access rate). Then, program data layout is re-arranged at compilation time for achieving fast and energy efficient hybrid STT-MRAM on-chip memory design with no reliability degradation. In this approach, application requirements have been defined at function granularity based on profiling and static code analysis, which estimate the required retention time and memory access rate, respectively.

Adaptive or dynamic data management is important, as it considers the unexpected behavior of the software based on the current status of the hardware. Dynamic data

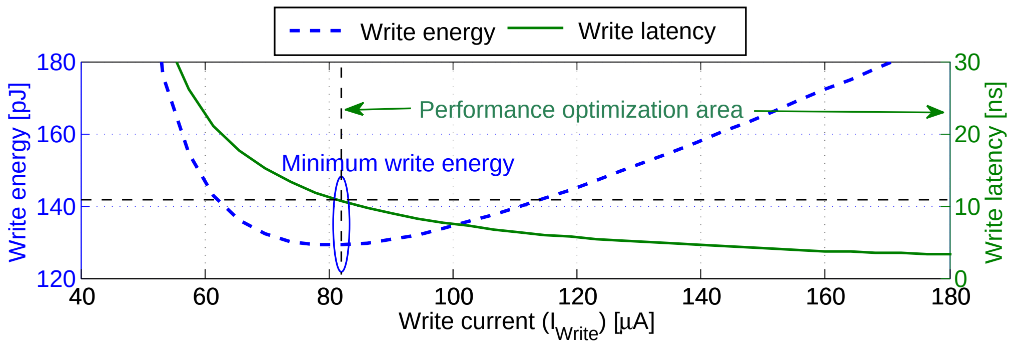


Figure 1.6: Performance and energy trade-off of the write operation in STT-MRAM technology

allocation is implemented based on access pattern prediction of read-intensive and write-intensive data. Three dynamic behavior prediction approaches are introduced with a prediction accuracy of 75%, on average.

The fundamental challenges (i.e., read and retention errors) related to employing semi-volatile STT-MRAM as a promising low-power solution can resolved in the scope of approximate computing, in which such errors can be tolerated at the application level. In general, approximate computing relaxes the bounds of accurate computing for the applications which are inherently error resilient and in return, it improves the efficiency of the system by other means such as performance and energy improvements. In this thesis a comprehensive trade-off study between energy and reliability models in semi-volatile STT-MRAM-based on-chip memory design is proposed.

1.5.2 Optimizing performance under reliability constraints

Typically, the write current required to switch STT-MRAM cell contents is set for the minimum writ energy point, which dramatically impacts the needed write latency, see Figure 1.6. On the other hand, at system-level, higher performance can achieve better energy efficiency, while boosting the performance required to meet application-level requirements, if the decrease in the execution time is lager than the increase in the induced power dissipation. Based on this observation, two approaches for optimizing the performance of STT-MRAM memory along with the total energy while satisfying reliability constraints are proposed in this dissertation.

Opportunistic write technique equipped with Lazy-ECC scheme is the first approach proposed to reduce the excessive write margin of the STT-MRAM technology to a large extent by terminating the write process before switching of the bits with large write latency are completed, which will be later processed by Lazy-ECC, which eliminates the costs of conventional ECC codes.

However, in some cases, the fast write termination could not achieve the application requirements. Therefore, an adaptive write current scaling technique is proposed, which adjusts the write current, and hence the write latency and energy based on the performance needs at run-time. Consequently, maximizing the overall system performance will be achieved along with minimizing the overall energy consumption under reliability constraints.

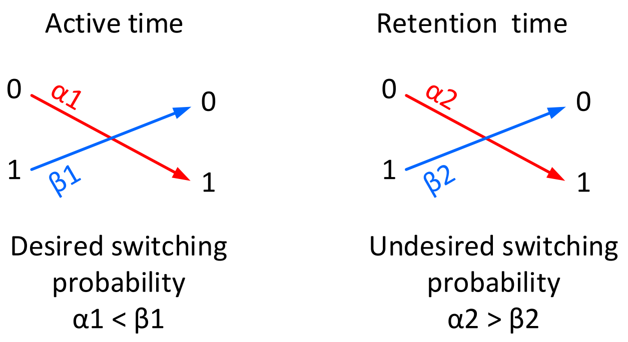

Active time
Retention time
0
α1
0
1
β1
1
0
α2
0
1
β2
1
Desired switching probability
α1 < β1
Undesired switching probability
α2 > β2

Figure 1.7: Asymmetric switching behavior of STT-MRAM bit-cell content

1.5.3 Minimizing overheads of satisfying target reliability

In order to have a low-power but also highly reliable STT-MRAM design, it is important to co-optimize reliability and energy efficiency simultaneously, as both constraints are tightly coupled via the employed thermal stability factor and read/write parameters (current and pulse). To achieve this goal, the fact that STT-MRAM technology poses asymmetric errors due to the nature of the asymmetric switching of the MTJ content during active and retention time (see Figure 1.7), has been exploited. Therefore, the efficient *Systematic Unidirectional Error-Detecting Code* SEDC is proposed to be adopted instead of a conventional ECC code. This SEDC scheme needs to be combined with a proper cache access mechanism (i.e., write through) in order to fetch correct data in a case of an error detection.

We also have verified the data integrity, which is retained in the standard or relaxed STT-MRAM memory by proposing a process variation and temperature aware scrubbing technique, where the cache lines have been clustered into different groups based on their retention times. Each of these groups uses different scrubbing intervals. In addition, the scrubbing interval is adjusted at run-time based on the operating temperature to guarantee target error rate.

1.6 Dissertation Outlines

This chapter presented the motivation and contributions of this thesis. The remainder of the thesis is primarily categorized into eight chapters:

- Chapter 2 provides the preliminaries on STT-MRAM technology and presents basics of cache architecture, as well as this chapter gives an overview about the coding theory. Summary of the state-of-the-art related to the STT-MRAM challenges as caches is discussed at the end of this section.
- Chapter 3 presents the details about hybrid SVM and NVM STT-MRAM cache array for energy-efficient design supported with statistic and dynamic data management policies.
- Chapter 4 presents the potential of the energy-efficient SVM STT-MRAM technology in the scope of approximate computing under the process variation and the operating temperature considerations.

- Chapter 5 proposes the first performance improvement approach. First, the *Opportunistic Write Policy* is introduced. Then, the proposed Lazy-ECC circuit is described, in which a fully combinational one stage decoding circuitry is broken up into two pipeline stages, detection and correction error stages. Finally, A non-uniform adaptive ECC approach for managing process variation and temperature-dependent errors at runtime has also been proposed.

- Chapter 6 describes the adaptive write approach for STT-MRAM performance optimization. First , the opportunity of scaling write current on-the-fly based on workload needs is presented with respect to energy-performance trade-off at device and system levels.

- Chapter 7 exploits the asymmetry in the MTJ switching behavior for writing and retaining '1' and '0' values by adopting asymmetric SEDC scheme instead of conventional symmetric ECC scheme. This leads to significant reliability enhancement in STT-MRAM design with negligible SEDC related overheads.

- Chapter 8 explains the proposed adaptive scrubbing mechanism which significantly reduces both performance and energy overheads associated with conventional scrubbing. In the proposed approach, STT-MRAM cache lines are clustered into different groups based on MTJ process variation effects, while scrubbing intervals of each group are adjusted dynamically according to the operating temperature.

- Chapter 9 concludes the thesis and discusses the potential future direction of the STT-MRAM memory.

Chapter 2

Background and State-of-The-Art

This chapter summarizes the background and related work concerning energy dissipation reduction and performance optimization under reliability constraints of an STT-MRAM-based on-chip memory design. In this chapter, a general background for cache architectures are firstly presented. Then, a short overview of the error detecting and error correcting codes is described. Afterwards, memory architectures using STT-MRAM technology are explained, followed by a comprehensive reliability study with respect to the operating temperature and process variations. Finally, state-of-the-art methodologies for adopting STT-MRAM technology for on-chip memories instead of SRAM are presented.

2.1 Cache Basics and Terminology

Cache memory in computing systems is small and fast memory, which is placed between processors and main memory. Caches are hardware-managed by cache controller and are organized as a power of 2 number of lines. Each line holds multiple bytes or words. Hence, data in cache is either byte-accessible or word-accessible. A cache line size is supposed to meet the size of main memory block. Cache is introduced in order to improve the memory response time and bridge the gap between high speed processors and slow main memories [8, 24, 25]. Therefore, it is implemented with faster technology rather than main memory, such as SRAM technology, stores a sub-set of the recent accessed memory contents to exploit temporal and spatial locality of the data [67], which means, once a memory location (block) is accessed, there is high probability to access it and its nearby locations in the near future. A typical cache is divided into instruction and data cache to hold the instructions and data of the programs fetched by the processor.

A cache array is divided into groups of columns and rows as illustrated in Figure 2.1(a).

- Index: to identify the cache line or group.
- Tag: to identify the data, as many blocks of memory may be mapped into the same cache line or group.
- Valid bit: to indicate weather the data in a cache array is valid (as “1” logic) or not (as “0” logic).

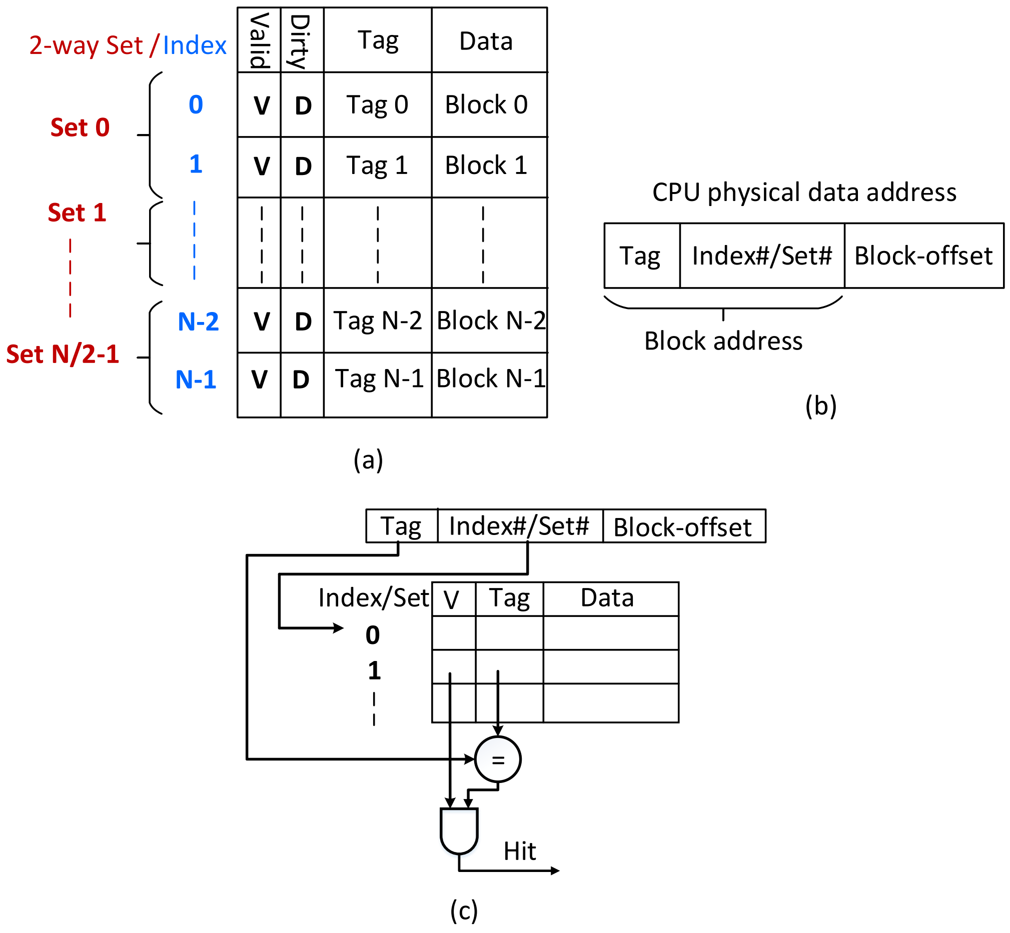


Figure 2.1: Cache structure: (a) Cache array; (b) CPU data/byte address; (c) Accessing data implementation

- Dirty bit: to indicate weather the fetched memory data in the cache array has been modified by a write request from the processor (as "1" logic) or not (as "0" logic).

Consequently, as shown in Figure 2.1(b), the CPU physical data address consists of tag bits, index bits, which are together referred to as "block address" in the memory, while block-offset bits identify a specific word requested by the processor from an accessed memory block.

Three methods are defined for mapping the memory data into cache lines [23, 68]:

1. Fully-associative: Memory address can be mapped into any cache line, resulting in long access time, as it is required to search for needed data in the entire cache.
2. Direct-mapped: Each memory address is translated directly into a specific single cache line, resulting in reducing the access time significantly at the cost of the data availability in the cache.
3. Set-associative: Memory address is translated into a limited number of cache lines. For I-way set-associative cache, a block of data from the main memory can be written only to one of I cache lines. Consequently, the time to search and access

data is reduced by a factor of N/I in a I-way set associative cache compared to a fully-associative one containing N number of lines.

Accessing data in caches requires a tag comparison logic (see Figure 2.1(c)), which fires either a hit or miss signal, if the requested data is found or not. The cache performance is tightly bound to the hit rate and hit latency. A hit/miss rate refers to the number of cache hits/misses divided by the total number of accesses, whereas hit latency is defined as the required time to transfer the requested available data to the processor. Consequently, a larger cache improves the hit rates at the cost of increasing the hit latency, while the hit rate for small caches will be reduced significantly. Addressing this contradictory issue can be solved by adopting a multi-level cache system [8, 15, 16], where fast and small low level caches (i.e., L1 and L2) are adopted to meet the performance requirements by reducing the hit latency, while a relative larger cache for higher level (i.e., L3) limits the impact of the miss rate.

Data consistency between cache levels and main memory is maintained with one of the two following write policies:

1. Write Through (WT): Data is updated in cache and main memory simultaneously. Thus, cache content is always consistent with memory. However, the overall performance is reduced significantly due to the unnecessary accesses to the slow main memory.
2. Write Back (WB): Data in main memory is not updated simultaneously with the modified data in the cache. This improves the performance of memory system at cost of higher design complexity to face the data inconsistency.

2.2 Error Detecting and Correcting Codes

Memory system is especially sensitive to variability (i.e., operating temperature and process variation), whose impacts are exacerbated due to the continuous voltage and technology down-scaling. Consequently, a correct store or access to the data in some memory cell may fail, resulting in hard and soft errors. Invented code-based techniques for error detection and correction partially accommodate such errors in an efficient way [69, 70], at cost of redundant information integrated with the original stored data in the memory. The more added redundant information, the higher are error detection and correction capability.

The stored information in memory is represented as a sequence of “0” and “1” (binary form). Hence, a finite field for detecting and correcting binary errors is defined as $GF(2^n) = (\{0,1\}^n)$, which has only two elements “0” and “1”, and consists of (n-bits) vectors, so called (n-tuples) codewords, representing polynomial of degree $\in \{0, 1, ..., n-1\}$ with arithmetic modulo of a prime generator polynomial of degree m over $GF(2)$; $m \leq n$. If the collection of n-tuples codewords is a set of all linear combinations of k independent (n-bits) vectors in a vector space $GF(2^n)$, this code C is called a *linear code*, and hence, any $k \times n$ matrix G of these vectors is defined as a *Generator Matrix*.

A code C is called *systematic*, if the G matrix has the form of $G = (I_k \mid P)$, where $I_k = (k \times k)$ is identity matrix and $P = (k \times (n-k))$ is redundant matrix. $H = (-P^t \mid I_{n-k})$ is a *Parity Check Matrix* of C:

$$(I_k \mid P)(-P^t \mid I_{n-k}) = -P + P = 0 \tag{2.1}$$

In a systematic code, the original message of m appears unscrambled at the beginning of a generated codeword ($c \in C = m \cdot G = m \in M \mid p \in P$). Encoder and decoder in linear code have a simple design, as encoding at worst case is a vector-matrix multiplication, and several steps in the decoding process are linear. Suppose that m, c, e and c', are the original, coded, corrupted, error vectors, respectively.

$$c' = c + e \longrightarrow e = c' - c \tag{2.2}$$

$$s = c' \cdot H^T = (c + e) \cdot H^T = c \cdot H^T + e \cdot H^T = e \cdot H^T \tag{2.3}$$

where, s is the syndrome calculated to find the possible errors patterns e in the coded message c'.

Hamming distance t between any codewords $ci, cj \in C$ is the number of coefficients in which they differ. A correcting code is considered as a perfect code if for some $t >=$ every codeword is within Hamming distance t of exactly one codeword. Consequently, the code bound (i.e., correcting capability) is $\lfloor \frac{1}{2}(t-1) \rfloor$.

Cyclic code is a linear code where a cyclic shift of any word from the finite field $GF(2^n)$ is a codeword of C, which is extracted from a right shift cyclic from another codeword.

$$(c_0, c_1, ..., c_{n-2}, c_{n-1}) \in C \Longrightarrow (c_{n-1}, c_0, c_1 ..., c_{n-2}) \in C \tag{2.4}$$

For an (n,k) cyclic code (C) over $GF(2)$, there is a monic and unique polynomial g(x) called as *Generator Polynomial* of C with degree of $(n-k)$, if and only if g(x) is a divisor of $\forall c \in C$. The g(x) is a divisor of $x^n - 1$ polynomial, and hence the *Parity-Check Polynomial* (h(x)) is calculated as follows:

$$h(x) = (x^n - 1)/g(x) \tag{2.5}$$

2.2.1 Reliability Modes of Binary Storage Device

1. Reliable storage device: It is an error-free storage device, where, the retained binary data is fetched exactly at the output without error (Figure 2.2(a)).

2. Symmetric un-reliable storage device: There are two possible outputs (correct and wrong output) corresponding to each input (i.e., retained data 0,1) with the same error probability (p) (Figure 2.2(b)).

3. Asymmetric un-reliable storage device: The error probabilities of (input, output) pairs are not similar 2.2(c).

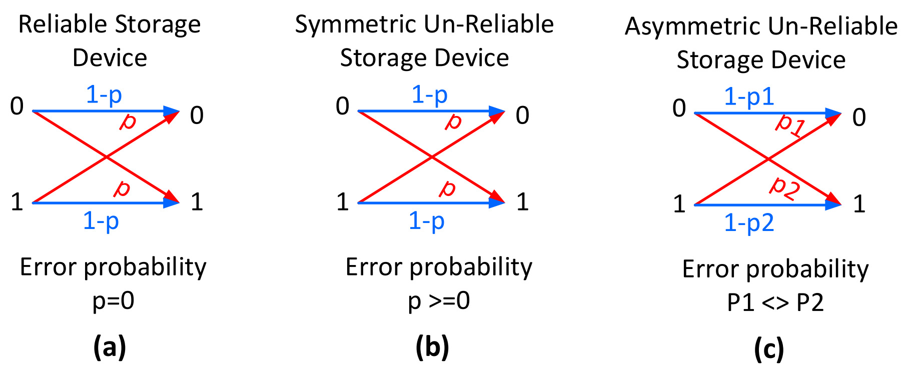


Figure 2.2: Reliability modes of binary storage device

2.2.2 Possible Error Codes

2.2.2.1 Single Error Correcting Hamming Code

Binary Hamming Code (n,m,e) [71, 72] is a code of length (n), $n = 2^r - 1 : \ r >= 2$, and has a *Parity Check Matrix* ($P = (k \times (n-k))$), whose vectors are of length r. Hamming code is a perfect error correcting code of distance t, dimension of m where: $n = m - r$, and with hamming bound of 1 (i.e., error correcting capability (e) of 1). This code supports a binary storage device with symmetric error probability.

$$(n, m, 1) = [(7, 4, 1), (12, 8, 1), (21, 16, 1), (38, 32, 1), (71, 64, 1)] \tag{2.6}$$

2.2.2.2 Bose-Chaudhuri-Hocquenhgem Code (BCH)

It is the most studied family of cyclic linear codes [73, 74]. The BCH code is derived from a single error correcting hamming code as a class of multiple error correcting codes. The generators polynomials (g(x)) are products of minimal polynomials f(x), over $GF(2)$ of vectors of $GF(2^n)$. The BCH code can correct e errors if the minimum hamming distance is $2e + 1$.

2.2.2.3 Systematic Unidirectional Error-Detecting Codes SEDC

The SEDC [75] is an optimal code for detecting asymmetric errors, as it is systematic and ideal to detect a fixed number of errors at run time. Furthermore, encoder/decoder circuits of SEDC are very simple with relatively low penalty, hardware and storage overheads. It is capable of detecting a large number of errors (e) with less number of required redundant bits r.

$$e = 5 \cdot 2^{(r4)} + r4 \tag{2.7}$$

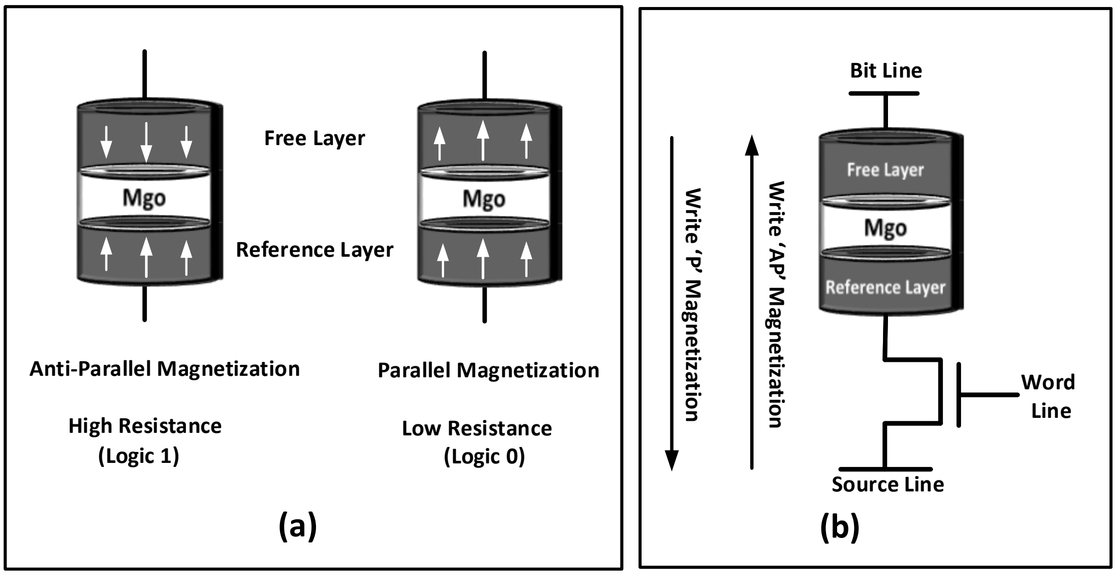


Figure 2.3: Spin Transfer Torque Magnetic RAM: a) Storage device and b) Bit-cell structure

2.3 Memory Architectures Using STT-MRAM Technology

2.3.1 STT-MRAM Basics

A typical STT-MRAM bit-cell consists of a *Magnetic Tunnel Junction* MTJ device and an access transistor, as shown in Figure 2.3 (b). The MTJ device is mainly composed of two ferromagnetic layers, namely *reference* and *free* layers. 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 magnetizations 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' (see Figure 2.3 (a)). The key parameter of the MTJ is the *Thermal Stability Factor* Δ, which specifies the stability of the magnetic orientation in the free layer against the thermal noise [48]. Hence, the adopted Δ defines the amount of current required to switch the magnetic orientation of the free layer during a write access. Alternatively, it determines the induced failure due to the erroneous switching of the magnetic orientation at idle time or during a read access [55]. The Δ value is highly influenced by the operating temperature and the process variation, i.e., the volume of the free layer of the MTJ device [48, 76].

2.3.2 Read and Write Accesses in STT-MRAM

The read access in STT-MRAM requires a small current (I_r) to sense the resistance value of the MTJ, whereas writing into the bit-cell needs a high write current (I_w), which is much higher than the critical current (I_c) (minimum current required to switch the magnetization of the MTJ for a given write period).

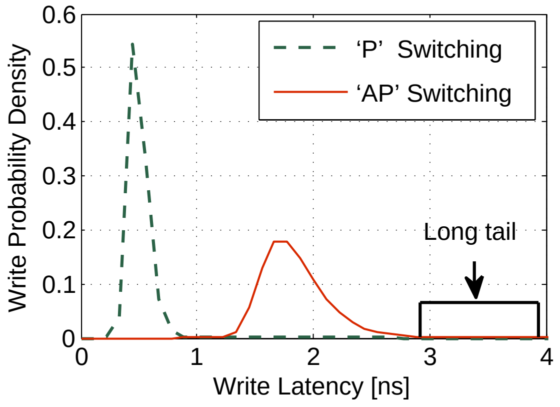


Figure 2.4: Write latency distribution for 'AP' and 'P' switching delays for a 64-bit data, [$\Delta = 30$, $I_w/I_c = 3$]

The write operation in STT-MRAM is intrinsically stochastic due to the random thermal fluctuations in the MTJ. Therefore, the switching time can significantly vary. The 'AP' and 'P' switching delay distributions for a 64-bit data are shown in Figure 2.4. As shown, the 'AP' switching delay has a wider distribution compared to the 'P' switching delay. Moreover, the 'AP' distribution has a very long tail. 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.

2.3.3 Retention Time and Critical Current in STT-MRAM

The data retention time indicates how long data can be retained in a non-volatile memory cell after being written. In STT-MRAM, the retention time depends on the *thermal stability factor* (Δ) value of the MTJ cell, which specifies the stability of the MTJ resistance state against the thermal noise. This means that the higher the Δ value, the longer the data can stay in the bit-cell. For a Δ value of 60, MTJ cell can retain its content for 10 years [48]. The Δ value can be modeled as:

$$\Delta = \frac{V \cdot H_k \cdot M_s}{2 \cdot K_B \cdot T} \tag{2.8}$$

where V is the volume of the free layer, M_s is the saturation magnetization, K_B is the Boltzmann constant, T is the temperature in Kelvin and H_k is the effective anisotropy field. As shown in Equation (2.8), the Δ value is directly proportional to the volume of the MTJ cell, which can be tuned by changing the MTJ size during fabrication. Another way to alter Δ is by adjusting M_s and H_k values at the material level during the MTJ stack development.

On the other hand, an MTJ cell with a high Δ value requires high switching latency and energy. This is due to the fact that the thermal barrier is higher for a high thermal stability, hence requires more current for a longer duration to perform the switching. The Δ relation with I_c can be modeled using the following equation as [77]:

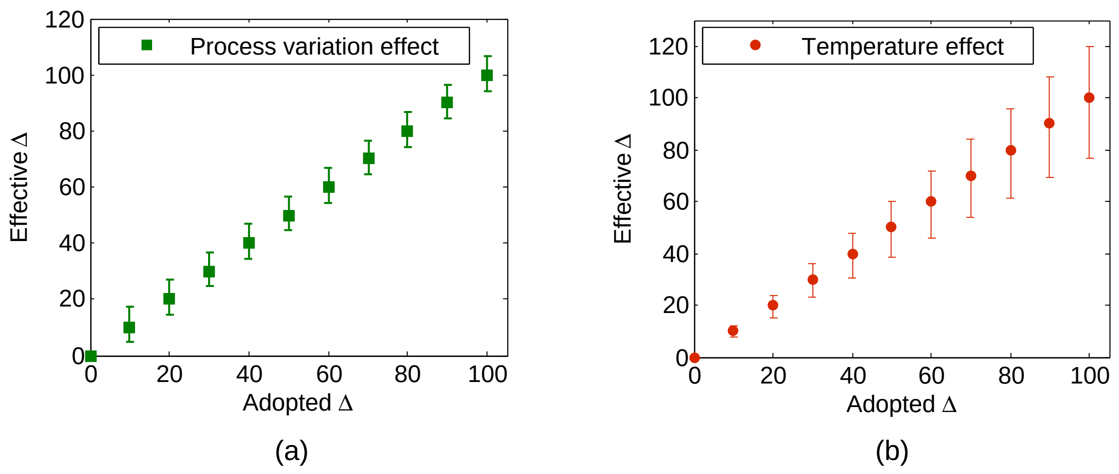


Figure 2.5: (a) Process variation; (b) Operating temperature effects on the thermal stability factor scaling

$$I_c = \frac{4 \cdot e \cdot K_B \cdot T}{h} \cdot \frac{\alpha}{\eta} \cdot \Delta \cdot (1 + \frac{4 \cdot \pi \cdot M_{eff}}{2 \cdot H_k}) \tag{2.9}$$

where h is the Planks constant, α is the damping constant, η is the STT-MRAM efficiency parameter and $4\pi M_{eff}$ is the effective demagnetization field.

As MTJ cell is mainly composed of several ultra-thin layers, any slight variation in the fabrication process of the oxide thickness results in an aggressive change in the magnetic and electrical STT-MRAM parameters including: 1) the thermal stability factor (Δ) for retention and read operation and 2) the critical current (I_c) for write operation. Therefore, process variations result in large variations in the switching probability during idle and active time. This in turn induces undesired bit switching, namely retention failure during idle time and at active read and write accesses so-called read disturb and write error rate, respectively. The MTJ switching parameters mainly depend on the volume of the free layer (V), which in turn depends on the radius (r) of the MTJ [76].

$$\{I_c, \Delta\} \propto V \propto r^2 \tag{2.10}$$

In this work, the MTJ oxide thickness is set to 2.0 nm, and an inter-die Process Variation PV of 5% according to the Pelgrom law is considered [78], as there is no published high volume wafer scale information available from foundries. Figure 2.5 illustrates the variations in the thermal stability factor based on the variations in the scaled r. For our analysis, we have assumed a normally distributed radius variation with a standard deviation of $\delta = 5\%$. This constructs the Δ variation, as it depends on the radius of the MTJ cell. However, the results are not dependent on particular PV assumptions and any particular distribution can be considered. Then, we select 4×10^6 and 0.5×10^6 samples for L2 (512 byte) and L1 (64 byte), respectively. Finally, the minimum delta is defined under the operating temperature variation for the data integrity guarantee.

2.3.4 STT-MRAM Reliability Challenges

STT-MRAM technology suffers from various reliability threats including *write failure*, *read disturb* and *retention failure* [79–82].

Read Disturb: Since both read and write currents share the same path in the STT-MRAM bit-cell, the read current can erroneously switch the bit-cell content. This phenomenon is known as read disturb. The read disturb probability is strongly dependent on the read current (I_r), the read duration (t_r), and the thermal stability factor (Δ). The read disturb probability (P_{RD}) is modeled as [55]:

$$P_{RD} = 1 - exp[-\frac{t_r}{\tau \cdot exp[\Delta(1 - \frac{I_r}{I_c})]}] \tag{2.11}$$

where τ is an intrinsic attempt time constant equal to 1 ns. I_c is the critical current which is defined as the minimum current required to flip the bit-cell state.

Retention Failure: In STT-MRAM, the thermal stability factor (Δ) determines the retention time of the bit-cell during the standby mode. The retention failure probability (P_{RF}) for a given time period (t) can be computed according to the read disturb failure model (Equation (2.11)) where $I_r = 0$ [48]:

$$P_{RF} = 1 - exp[-\frac{t}{\tau \cdot exp[\Delta]}] \tag{2.12}$$

Write Error: As mentioned earlier, the write operation in STT-MRAM is of stochastic nature, and the switching time can significantly vary from one access to another. In case the write current is terminated before the MTJ switching, the desired data will fail to be stored correctly, leading to a write failure. The Write Error Rate (WER) in a single bit-cell is a time dependent variable and is expressed as [83]:

$$WER_{bit}(t_w) = 1 - exp[\frac{-\pi^2 \cdot (I-1) \cdot \Delta}{4(I \cdot e^{[C(I-1)t_w]} - 1)}], I = \frac{I_w}{I_c} \tag{2.13}$$

where t_w is the write latency and C is a technology dependent parameter that is calculated from the Gilbert damping parameter and the gyro-magnetic ratio. In Equation (2.13), I is the ratio of the write current (I_w) to the critical current (I_c). At device-level, the critical current is extracted based on the technology parameters, which are independent of the write pulse width. This is illustrated in Equation 2.9, where the critical current value is driven from the value of the thermal stability factor (Δ), effective demagnetization field, etc. However, at circuit-level, the critical current, which is the minimum current required to switch the magnetization of the MTJ, is defined based on the switching time. For example, for the read disturb phenomena, Equation 2.11 presents the switching probability at read access. From this equation, it is clear that there is a considerable switching probability although the read current is around 5 to 8 times less than the critical current. The other parameter that is impacting such switching probability is the duration of the current flow. Therefore, for circuit-level analysis, the critical current has to be measured for a particular duration.

According to Equation (2.12), the retention failure can be controlled by exploiting a memory with a reasonable Δ according to the maximum possible time that data resides

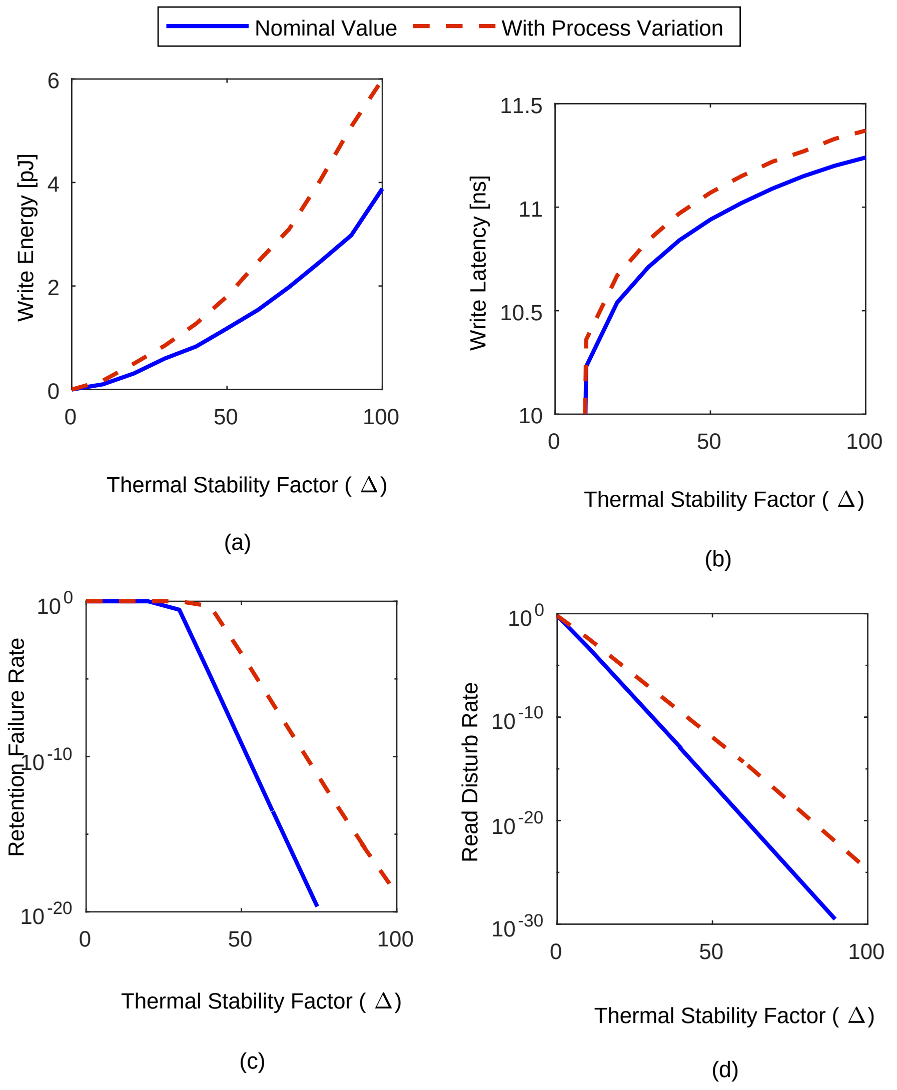
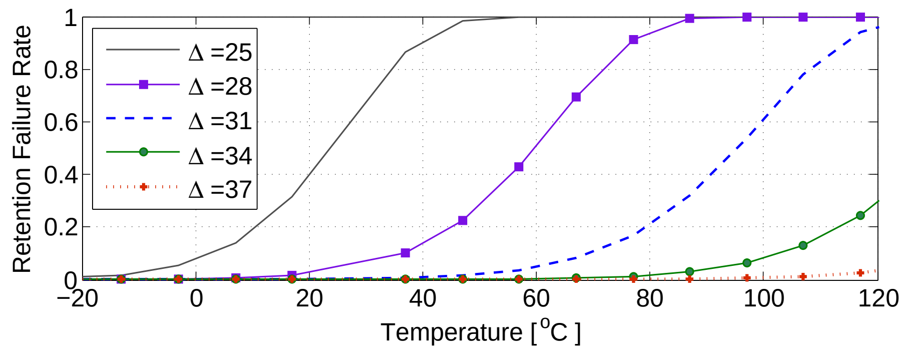


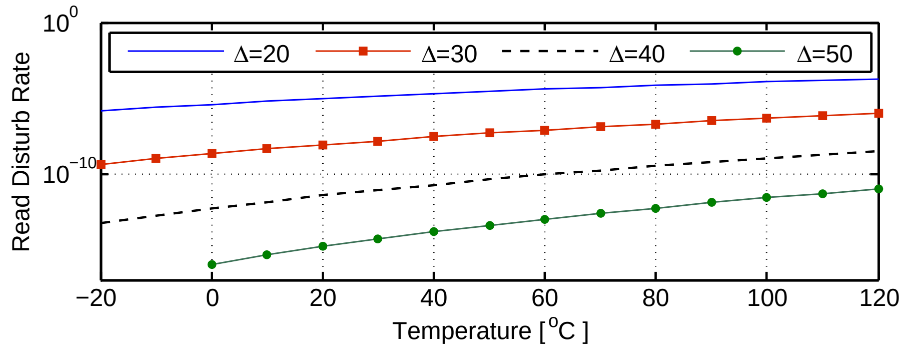
Figure 2.6: Thermal stability factor effects combined with process variation effects on (a) Write energy, (b) Write latency, (c) Retention failure and (d) Read disturb rates

in the memory. For example, $\Delta = 20$ provides a retention period of one second and is suitable for L1 caches, while L2 caches require a higher Δ value between 30 and 40, as the data may require to stay there for considerably longer periods [48]. However, reducing Δ not only affects the retention time, but also exponentially increases the read disturb probability based on Equation (2.11). On the other hand, based on Equation (2.13), reducing Δ has a positive impact on the write latency due to a lower thermal barrier [84].

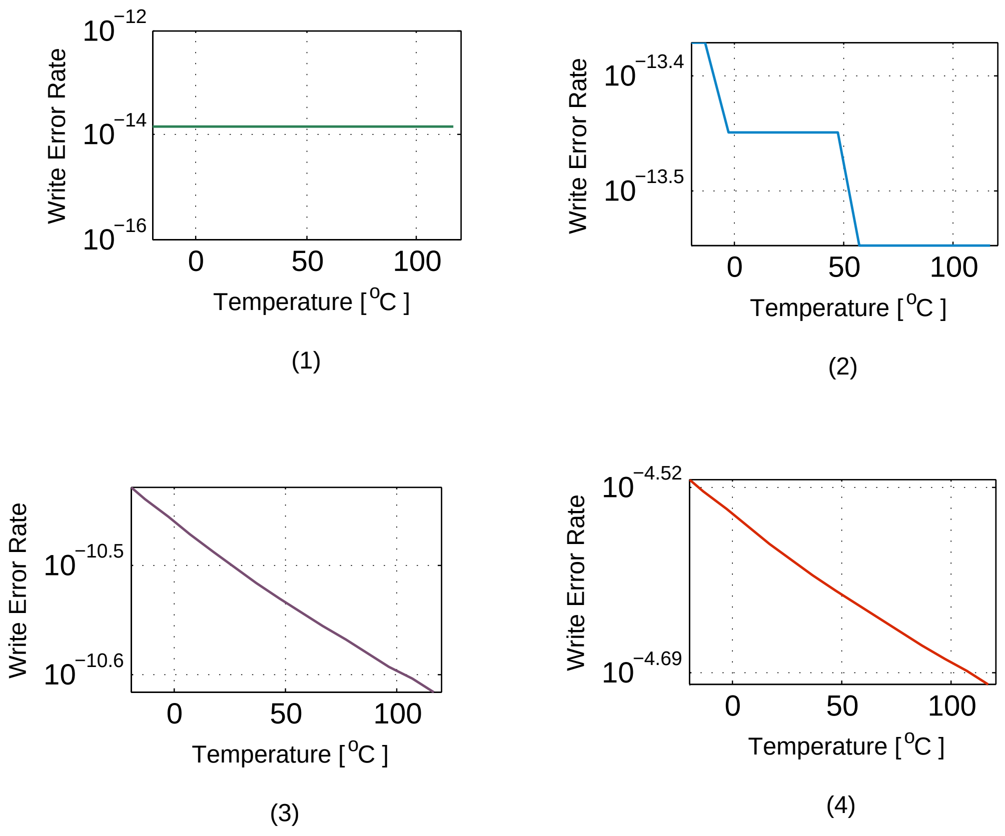
With continuous technology down-scaling, the thermal noise and the process variation impacts increase significantly, that negatively affect the data validity and the performance of STT-MRAM memory. We have studied the process variation effects by analyzing 0.5×10^6 samples of STT-MRAM cells for a 64 KB STT-MRAM cache with line size of 64 Byte. Based on Equation (2.11), (2.12) and (2.13), the variation on thermal stability results in a considerable variance in the write operation parameters as well as various failure rates, as shown in Figure 2.6. As a case study, Table 2.1 shows the combined effects of the process variation and the random thermal fluctuations of the MTJ cell on the reliability of the STT-MRAM cell for relaxing the thermal stability factor



(a) Impact of temperature on retention failure rate of MTJ cell for various Δ values



(b) Impact of temperature on read disturb probability of MTJ cell for various Δ values



(c) Impact of temperature on the write error rate of MTJ cell for different write margins ($\Delta = 30$, $I_w/I_c = 3$)

Figure 2.7: Operating temperature effects on STT-MRAM bit-cell reliability

Table 2.1: Impact of thermal stability factor scaling from 60 down to 30 on bit-cell reliability in the nominal case (without process variation) and in the presence of 5% process variation effects

Process variation	Nominal case	Process variation of 5%
Switching energy reduction	61%	65.5%
Switching latency reduction	3%	4.1%
Retention failure probability increase	10^{13}	10^{7}
Read disturb probability increase	10^{13}	10^{14}

value from 60 down to 30, which means retaining data, on average, from 10 years down to less than 10 seconds.

According to Equation (2.8), the value of thermal stability factor (Δ) is highly sensitive to the temperature. This means that it is important to consider the effects of the operating temperature variations on the thermal reliability (i.e., retention failure and read disturb) of the MTJ-cell for a real application.

Based on Equation (2.8), (2.11), and (2.12), as the operating temperature (T) increases, Δ reduces linearly, while retention failure and read disturb probabilities increase exponentially. Hence, any minor change in the temperature significantly affects the retention and read disturb rates, as illustrated in Figure 2.7(a) and (b), respectively. On the other hand, during the write operation, as the temperature increases, the high resistance of the MTJ decreases. This leads to a decrease in the Δ value, which results in the reduction of the critical current and hence, the switching current. This means that for the same write pulse set for target WER $> 10^{-14}$, the achieved WER improves at higher temperatures (see Figure 2.7(c)). As shown in Figure 2.7(c) (1, 2, 3 and 4), the reduction in the write error induced by the operating temperature increases at lower target WER.

According to the Δ variations, different cells of the STT-MRAM memory array have retention failures at different rates. Consequently, process variations can cause the thermal stability factor of the bit-cells of each line to vary significantly from the nominal value. Therefore, for retention failure fixing, the scrubbing interval has to be determined based on the cell with worst retention capability of the entire memory array. However, this introduces significant scrubbing overheads due to: i) stalling memory from accepting external read/write accesses and ii) energy consumption due to reading and checking all cache lines sequentially for errors.

In order to have an accurate estimation of the target retention time based on process variation and temperature effects, we have to start from the failure budget for the whole system, which is measured as Failure in Time (FIT). One FIT equals one failure in one billion hours of operation. The FIT rate per single cache line has to be bounded by FIT/K, where K refers to the number of cache lines. The failure probability of a cache line ($P_{fail-line}$) with no error correction capability can be computed as:

$$P_{fail-line} = 1 - (1 - P_{fail-bit})^n \tag{2.14}$$

where $P_{fail-bit}$ is a single bit failure probability and n is the the number of bits. With an ECC correcting e errors, $P_{fail-line}$ can be computed by solving the following binomial distribution model:

$$P_{fail-line} = 1 - \sum_{i=0}^{e} \binom{n}{i} \cdot (P_{fail-bit})^i \cdot (1 - P_{fail-bit})^{n-i} \tag{2.15}$$

2.4 State-of-the-art STT-MRAM Cache

Despite all aforementioned STT-MRAM technology benefits, the long write latency and high write energy limit the application of STT-MRAM in high performance memories such as first and second levels of caches (e.g. L1 and L2) [61, 62, 85].

The write operation in STT-MRAM is non-deterministic where the write latency follows a log-normal distribution with a long tail [86]. The amount of uncertainty in the write latency is even higher in advanced technology nodes due to severe process variation effects [76], which influences the magnetic properties of the cell such as switching characteristics, retention ability, etc. Furthermore, the magnetic properties of the STT-MRAM cell are significantly affected by the temperature variation. The common approach to address the write uncertainty is to consider a static timing margin based on the worst process and operational conditions. This implies that each write access which cannot be finished within the given timing margin leads to a *write error*. However, the amount of timing margin required to satisfy a required *Write Error Rate* (WER) based on the process variation effects is considerably larger than the average switching time [87]. This significant increase in the write latency of STT-MRAM not only imposes a considerable performance penalty, but also leads to significantly higher write energy, as the write current flows through the bit-cell even after the actual switching completion [88, 89]. Therefore, it is very important to reduce the excessive write margin and the associated write energy in a cost-effective manner.

The prior research efforts have explored several techniques to achieve energy-efficiency and high-performance under reliability constraints of an STT-MRAM design at various levels.

2.4.1 Device-Level Techniques

At the device-level, the extreme write margin can be further addressed by reducing the thermal stability factor (Δ) and trading off retention time, which speeds up the switching of the storage element in an STT-MRAM bit-cell (i.e., the *Magnetic Tunnel Junction* (MTJ)) [84, 90]. The retention time is relaxed in [48] by shrinking the MTJ planar area, while reducing the volume of MTJ free layer (V) by either decreasing its thickness or lowering the saturation magnetization (M_s) as adopted in [91].

However, this method is implemented without considering the workload requirements at application level, which may erode the device gains at system level. As the low Δ value can dramatically affect the retention time of the memory, this results in data integrity loss especially under temperature and process variations and hence, an expensive refresh operation is required as proposed in [48, 91]. Nevertheless, the related overheads in terms of performance, energy and area are relatively high, when such refresh schemes are used.

2.4.2 Circuit-Level Techniques

Several circuit-level techniques [87–89, 92, 93] have been proposed to detect the actual switching time of an STT-MRAM bit-cell and terminate 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 Correcting Code* (ECC) can be used to correct write errors in the long tail of the write distribution [94–96]. 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 [94]. The amount of reduction is maximized by exploiting more robust ECCs which can correct multiple errors in a single word [94, 96, 97].

Unfortunately, the decoding latency of such robust ECCs is relatively high (i.e., multiple cycles) [98]. This significantly impairs the read latency, and hence, erodes the gain from the 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.

Device-level and circuit-level techniques address the STT-MRAM challenges at design time regardless of the applications behavior which have to be considered at run-time, and may affect the required write parameters significantly.

2.4.3 Approximate Computing Techniques

Previous research on the approximate memory focuses on finding a trade-off between errors in the content of the saved or read data and the energy consumption. In the prior work of DRAM memory [99, 100], the energy consumption is optimized by relaxing the refresh rate which reduces the refresh power. However, the retention failure increases at the cost of this method. In [101], the energy reduction is achieved by reducing the number of pulses used to write *Phase Change Memories*. For STT-MRAM memory, the prior efforts are very limited.

In [102], the approximate computing is leveraged for STT-MRAM design improvement in which the MTJ cell is accessed with different levels of accuracy (i.e., different current levels) based on the application requirements. Hence the approximation is adopted in order to reduce the dynamic energy of read and write operations by reducing both sensing and switching currents at the cost of higher read and write error rates. The main limitation of this work is the effect of the process variation which is totally ignored.

Process variation in STT-MRAM affects device parameters of both MTJ cell and CMOS transistor which in turn impacts the main parameters of read and write operations (i.e., the switching current, the thermal stability factor and the output current of the transistor). Consequently, with different and tiny read-write access current levels, process variation results in considerable variations in the output accuracy at architecture-level. Exploring the approximation computing for STT-MRAM design to achieve substantial improvements in both energy consumption and performance for negligible loss in the application quality by lowering the thermal stability factor of MTJ cell should be based on the application requirements.

2.4.4 Hybrid Cache Techniques

Write-efficient but leaky, volatile and low density SRAM is integrated with non-leaky high density and non-volatile STT-MRAM to compensate for the effects of inefficient write operations of STT-MRAM [47, 103–106]. However, the main challenge of implementing the hybrid caches is the data placement. The write rate is considered as a run-time data placement criterion for maximally reducing the write activity in STT-MRAM array. The previously proposed solutions for the data placement based on the write rate are either too simple or too static, which leads to frequent data migrations between STT-MRAM and SRAM arrays. Consequently, the overall performance and energy consumption are relatively high. However, the aforementioned migration overheads are reduced in [107] by managing the data layout at compiler-time based on the read/write access rate.

A hybrid STT-MRAM cache with low and high retention STT-MRAM cells has been proposed in [90]. Nevertheless, the data placement policy erodes the system gains due to the data migration and control costs, as they depend on write counters saturation. The authors in [108, 109] have proposed data placement optimization using dynamic schemes based on run-time information. However, these schemes require a mechanism to measure the write intensity for each cache block, which makes the related hardware overheads considerably high for on-chip memories. [110] presents the principle of multi-retention STT-MRAM technology, which is limited to the scratchpad memory application, as it is software controlled. In [110], data arrangement is performed by compiler-based data placement algorithm based on the worst-case execution time of all functions, while neglecting the potential data corruption in the low retention STT-MRAM region due to the impact of the data read behavior.

2.4.5 Scrubbing Techniques

Retention failures have been addressed in many previous works for Flash, DRAM and also STT-MRAM technology.

For Flash technology, the first work on optimizing SSD by relaxing the retention of NAND flash was proposed in [111]. Here the retention time is characterized from a set of datacenter workloads. To fix the errors due to the relaxed retention time, the authors propose using variable ECC codes based on the retention requirements. The work in [112] proposes a retention time trimming approach which exploits the fine-tuned retention time requirement of each data to-be-programmed, for reducing the wear-out of flash memory. This work uses a retention time prediction scheme based on the past retention time and the past prediction of retention time. However, the aforementioned works are primarily focused on characterizing retention time based on workloads for storage systems. In our work, the focus is on on-chip memories and is based on well known device-level models targeting a certain reliability of the memory system.

Many proposals have been made to address the retention failure of DRAM technology. They mainly adopted refresh mechanisms with an interval of 64 ms based on shortest retention time determined by the worst case retention capability of DRAM bit-cell. This means that memory contents should be refreshed periodically every 64 ms for reliable operation. Various multi-rate refresh mechanisms [113–116] have also been proposed to exploit the non-uniformity in the retention capabilities of different bit-cells. In [114], the single refresh interval has been optimized by considering the worst cases as hard errors and supporting them with additional error correction codes. In [115] the non-uniformity of the retention capabilities of DRAM bit-cells in one row is addressed using different error correction capabilities based on the information of the fault map. In general, the proposed multi-rates refresh scheme for DRAM aims to isolate the rows with weak retention capabilities and apply higher refresh rate on them [113]. However, deploying DRAM-style refresh for STT-MRAM technology is challenging because of several reasons: i) simply writing back the memory contents periodically will not improve the reliability for STT-MRAM, since the retention errors in STT-MRAM are not decay-based and ii) the temperature dependency of STT-MRAM retention failures is completely different from and much more severe than that of DRAM.

For STT-MRAM, previous work of relaxing the thermal stability factor and estimating the related retention time is done either without process variation impacts or with no considerations of the operating temperature effects. In [48, 91, 117], simple DRAM-style refresh is used for retention failure. However, this method is not effective for STT-MRAM, since it just results in reading the corrupted bit-cell contents and writing them back. Therefore, error correction capability is required for retention failure prevention, and the write-back needs to be done only for erroneous data as in [78]. Furthermore, this work also proposes to optimize the scrubbing costs significantly with stronger ECCs. In the previous work [90], simple DRAM-style refresh is adopted with a dynamic interval based on the benchmark behavior (i.e., last access write) and without considering the impact of process variation. Therefore, the related overheads in terms of performance, energy and area are relatively high.

2.5 Summary

In this chapter, basic information regarding cache architecture and its accessing policies are provided. Afterwards, a preliminary explanation about the coding theory and the common error codes adopted in the memory system are explained. This is followed by the basics, parameters and reliability challenges of the STT-MRAM technology as on-chip memory. Finally, there is an overview about the existent STT-MRAM state-of-the-art solutions.

Chapter 3

Efficient STT-MRAM Based on Compiler-Assisted Analysis and Dynamic Behavior Predictions

In this chapter, we propose a combined device, architecture and application levels solution for a fast, energy-efficient and reliable STT-MRAM design. *At device-level*, two different STT-MRAM bit-cell parameters are considered, i) non-volatile STT-MRAM (NVM) bit-cell with a large retention ability for reliable magnetic switching, and ii) semi-volatile STT-RAM (SVM) bit-cell with a relaxed retention ability for fast and efficient magnetic switching. *At architecture-level*, a hybrid STT-MRAM memory array is proposed with two regions: i) SVM array blocks and ii) NVM array blocks. Each region is configured with specific read and write currents and latencies. The SVM array is proposed to attain a high write access efficiency (both energy and performance), while the NVM blocks are adjusted to maintain a high reliability against both retention failure and read disturb. Finally, *at application-level*, a data mapping in the hybrid cache blocks is proposed based on the application needs (performance and reliability). These needs are driven from the data behavior in the cache. The high write intensive data, which has low retention requirements, can be loaded into SVM blocks. On the contrary, the NVM blocks store low write intensive and long lifetime data. Data placement policy is constructed based on either compiler-assisted analysis Section 3.2 or dynamic behavior predictions Section 3.3.

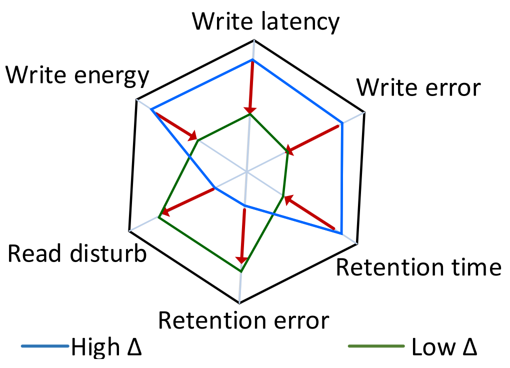
3.1 Hybrid Semi- and Non-Volatile Cache MOTIVATION

A promising approach to hide the write overheads of the standard STT-MRAM technology is to adopt a hybrid STT-MRAM cache architecture in order to combine the benefits of fast write and reliable read with different levels of data retention. In this hybrid architecture, SVM is used for write-intensive data, which means short data residency, while NVM is used for read-intensive and long data residency. The motivations and challenges of such approach at different abstraction levels are discussed next.

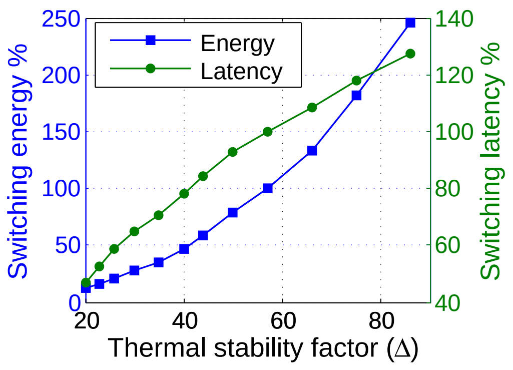
3.1.1 Device-level Motivation

NVM STT-MRAM supported with a high Δ value offers high ability to retain the data for a long time at a cost of a high switching current required to flow for a long duration. However, on-chip memories (e.g., L1 and L2 caches) often do not require very long data retention periods due to their frequent accesses in comparison to the off-chip memories. Moreover, the write operation in the on-chip memories is on the critical performance path. Therefore, STT-MRAM technology with relaxed retention ability (i.e., a lower Δ value), so called SVM STT-MRAM, can be leveraged for fast and low level caches. Figure 3.1(a) qualitatively illustrates the impact of lowering the thermal stability factor on six design criteria (i.e., write latency, write energy, write error, read disturb, retention error and retention time).

Figure 3.1(b) shows the induced energy and latency for bit-cell content switching of different Δ values normalized to the results of a standard $\Delta = 60$. In this figure, the switching latency and energy of STT-MRAM bit-cell can be optimized by 35.0% and 72.5%, respectively, by lowering Δ from 60 down to 30. These significant gains are achieved along with reducing the average retention time from 10 years down to few seconds, which is sufficient for low level caches. On the other hand, the read disturb probability due to lowering Δ from 60 down to 30 increases by $10^{13}\times$. This means that adopting only SVM STT-MRAM for low level caches may lead to a data corruption due to the read disturb, as the reduced retention time to few seconds is still sufficient to retain the data correctly. Consequently, considering a hybrid cache architecture of SVM and NVM STT-MRAM can achieve the reliability requirements along with a high performance and low energy consumption.



(a)



(b)

Figure 3.1: Δ Scaling impacts: (a) Δ effects on STT-MRAM design metrics, (b) Switching energy and latency of different Δ values normalized to $\Delta = 60$

3.1.2 Application-level Motivation

To explore the opportunity of adopting a hybrid STT-MRAM architecture for on-chip memories, the cache behavior of the memory intensive SPEC2000 applications has been analyzed based on the data residency time, read and write operations in L1 data cache. We run the benchmarks on gem5 simulator that is a cycle accurate simulator and provides an accurate cache model [118]. During the benchmark execution, the residency time of each memory address in the data cache is monitored. The residency time (lifetime) is defined as the interval between the time of loading the data into the cache from a lower memory level and the time of evicting this data to a lower memory level. As a result, we observed the following three motivations:

1. The residency time of all memory data in the data cache lines can be categorized into two patterns: short and long (Figure 3.2 (a)).
2. More than 90% of the data cache lines is mostly accessed by either read or write requests from the processor. Therefore, classifying the memory data into either read intensive or write intensive data patterns can be easily achieved (Figure 3.2 (b)).
3. The system performance can considerably be optimized by increasing the size of the data cache because of the hit rate improvements (Figure 3.3 (a), (b) and (c)).

Initially, in L1 cache, the data residency time is divided into a vulnerable and non-vulnerable time. *A vulnerable time* is the duration between loading data from the lower memory to L1 data cache and using it again through read requests from the processor side, or the duration between the last write request and the next read requests of the data in the cache. Consequently, the target retention time is directly driven from the vulnerable time, in which the validity of the data should be guaranteed. On the contrary, any failure (such as retention failure) during the non-vulnerable time does not affect the system-level output.

The required retention time of data in L1 cache is application behavior dependent, and hence, it can be estimated with the execution of the SPEC2000 benchmarks. The results shown in Figure3.2 (a) demonstrate that the retention time (i.e., the vulnerable time) of the data in L1 cache can be categorized into short and long periods (i.e. less or more than 10 ms). As the data cache size is multiple times smaller than the main memory, the replacement policy is applied for loading new data in the cache, and hence, the majority of the data life-times is short. Thus, according to our simulations conducted on SPEC2000, only 5% data required long retention time in the data cache on average.

Afterwards, by considering the read and write access patterns of the data in the cache, the overall reliability, performance and energy efficiency of the hybrid STT-MRAM cache can be optimized significantly. As the write intensive patterns require high energy and latency with low retention time, the data can be accommodated in the SVM STT-MRAM cache. Whereas, the NVM STT-MRAM cache is dedicated to the read intensive data. This is for reducing the error rate due to the read disturb, which increases exponentially with lowering the thermal stability factor based on the read disturb Equation (2.11), as explained in Section 2.3.4. Figure 3.2 (b) illustrates that the read and write intensities of the benchmarks are 60% and 16%, respectively, of the entire memory data.

Finally, as the SVM STT-MRAM with low thermal stability factor exhibits higher density compared to the non-volatile one, the overall performance is optimized because of the increased hit rate. Figure 3.3 (a), (b) and (c) show the cache density increase and the improvements regarding hit rate and system performance for 'gcc' benchmark as a case study, along with lowering the thermal stability factor for the same hardware area of 32 KB L1 data cache. As seen in Figure 3.3 (a), for lowering Δ from 60 down to 30, the size of data cache will increase by 4 KB (from original 32 KB) under the same area constraints.

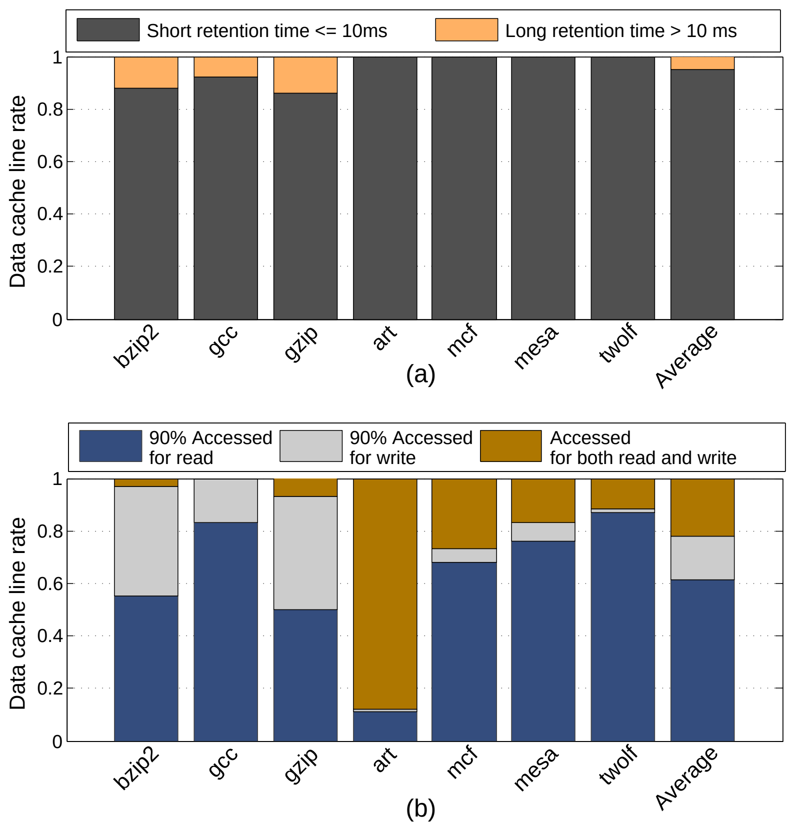


Figure 3.2: L1 cache line behavior for SPEC2000: (a) Needed Retention time for L1 cache based on the vulnerable time, and (b) Read and write rate distributions over L1 cache lines [See setup in Section 3.2.3.3.1]

Based on the previous three observed motivations, we propose a hybrid STT-MRAM architecture with N-ways NVM STT-MRAM array and N^*-ways semi-volatile STT-MRAM array, where $N^*/N = i$, as illustrated in Figure 3.4. This hybrid STT-MRAM architecture is mainly employed in order to achieve:

- Energy-efficient design, as the data with short residency time can be placed to the semi-volatile STT-MRAM array.
- Reliable design (by reducing read disturb and retention failure rate), as the read intensive data and the data with long residency time should be mapped to the non-volatile STT-MRAM blocks.

3.1.3 Motivation Example

This example is presented to illustrate the purposes of the proposed hybrid STT-MRAM architecture. Assume that an application consists of six main functions with a specific control flow graph. Each function has different features, as shown in Figure 3.5: run-time, data (i.e., heap and stack memories), number of calls, and the op-code instructions that illustrate the processor read and write requests. In Figure 3.5, the T and R values define the retention time and the read disturb rate, respectively, of the semi-volatile STT-MRAM. Table 3.1 shows the opportunity of adopting hybrid STT-MRAM cache against a pure non-volatile or semi-volatile one.

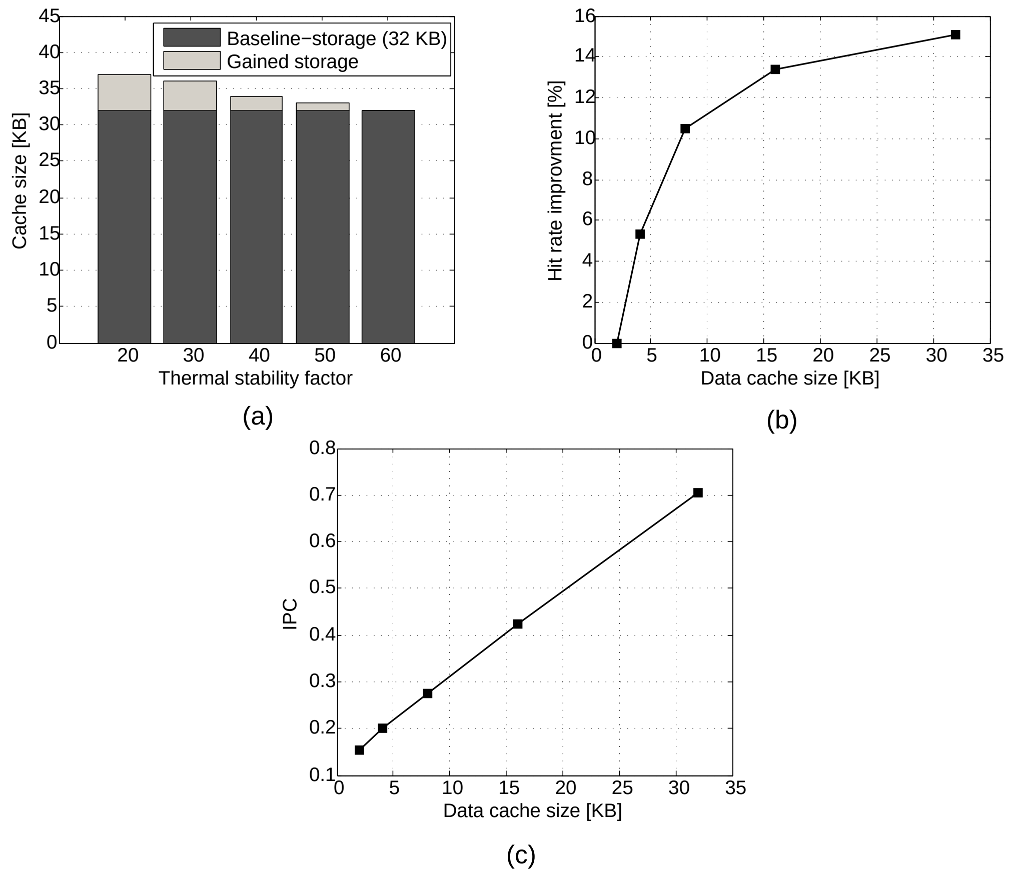


Figure 3.3: Cache benefits with the reduction of the thermal stability factor: (a) L1 cache storage density, (b) L1 cache hit rate improvement for 'gcc' workload, and (c) Performance improvement as Instruction Per Cycle (IPC) of 'gcc' workload

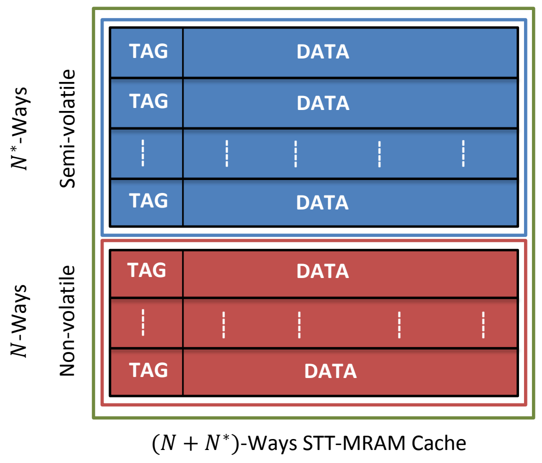


Figure 3.4: The proposed hybrid STT-MRAM architecture

For the semi-volatile STT-MRAM with Δ of 30, where the write energy is reduced by 70% (see Figure 2.6(a)), there is no data integrity guarantee against read disturb and retention failures in 'A','B' and 'F' functions. On the other hand, by employing a reliable and pure non-volatile STT-MRAM with normal Δ of 60, the associated costs with the write operation are so excessive in order to execute the 'C', 'D' and 'E' functions. Regarding these functions and in comparison to the semi-volatile STT-MRAM, the write

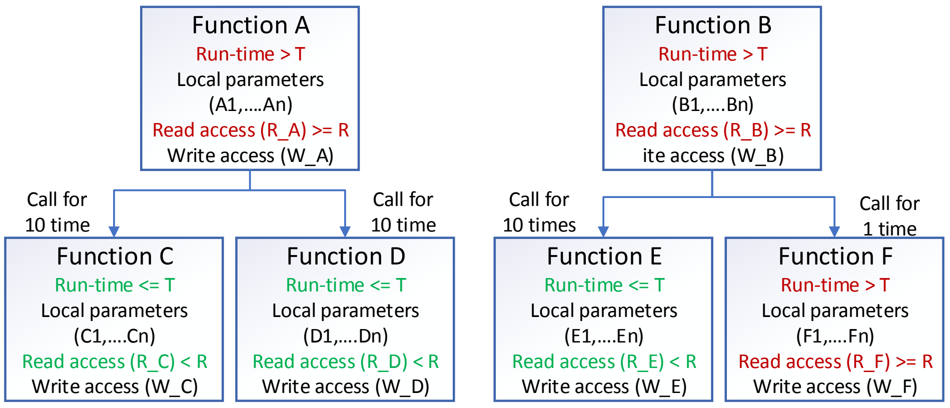


Figure 3.5: Motivation example, function diagram with their features (T is the retention time, and R defines the read disturb rate, of the semi-volatile STT-MRAM cache)

Table 3.1: Approximate total energy, performance and reliability results normalized to hybrid STT-MRAM design [See setup in Section V-A]

	Semi-volatile STT-MRAM			Non-volatile STT-MRAM			Hybrid STT-MRAM		
	Write energy	Write latency	Reliability fulfillment	Write energy	Write latency	Reliability fulfillment	Write energy	Write latency	Reliability fulfillment
Function A	0.3	0.87	No	1	1	Yes	1	1	Yes
Function B	0.3	0.87	No	1	1	Yes	1	1	Yes
Function C	1	1	Yes	3.3	1.15	Yes	1	1	Yes
Function D	1	1	Yes	3.3	1.15	Yes	1	1	Yes
Function E	1	1	Yes	3.3	1.15	Yes	1	1	Yes
Function F	0.3	0.87	No	1	1	Yes	1	1	Yes
Total	0.65	0.94	No	3,1	1.14	Yes	1	1	Yes

energy dissipation is 3.3 times more along with 15% increase in the write penalty (see Figure 2.6(a) and (b)). On the contrary, a hybrid STT-MRAM cache will provide the target reliability design for 'A', 'B' and 'F' functions along with lower power and fast design for 'C', 'D' and 'E' functions.

3.2 Compiler-Assisted and Profiling-Based Analysis for Fast and Efficient STT-MRAM On-Chip Cache Design

3.2.1 Proposed Approach

In STT-MRAM the overall write energy can become significantly lower. However, relaxing the non-volatility of STT-MRAM results in a higher probability of the retention failure and the read disturb, as illustrated in Section 2.3.4, based on Equation (2.11) and (2.12). The impact of this increase of both read disturb and retention failure is data pattern dependent (i.e., read or write intensive pattern) and data behavior dependent (i.e., long or short retained data). Therefore, in our proposed hybrid STT-MRAM cache architecture, the data integrity is guaranteed in two steps: 1) profiling-based analysis and 2) compiler-level (i.e., static code) analysis.

- *Profiling-based analysis* to estimate the functions execution time, which in turn identifies the data retention time (data life-time).

Table 3.2: Time-consuming aspect based on data type

Data type	Retention time	Target reliability
Global data	Long	High
Local data	Short	Low

Table 3.3: Read-threshold rate based on the thermal stability factor (Δ) under process variation effects

Δ	10	20	30	40	50	60
Read rate	1.1×10^{-2}	1.3×10^{-5}	1.523×10^{-8}	8.423×10^{-12}	4.66×10^{-15}	10^{-21}

- *compiler-level analysis*, in which the source code of the program is fed to the compiler (here LLVM [119]) to detect and analyze the memory access patterns.

In general, the data in a program can be distributed over three different pools of the memory, which are static, stack and heap. Static memory persists throughout the entire life of the program, and it is usually used to store global variables. Whereas, stack and heap memory is used for local data, which is accessed by only the owner function. Therefore, global data needs to be retained for longer time and can be accessed more frequently than local data. This means in a STT-MRAM cache architecture, high retention time and low read disturb rate are required for the global data. Therefore they need to be mapped to the non-volatile cache blocks whose thermal stability factor is high. Consequently, the data participation over the proposed hybrid STT-MRAM cache depends initially on identifying the data types (global and local), as illustrated in Table 3.2. However, the local data requirements (i.e., the retention time and the reliability) are not identical for all program functions, since there are parents and children functions.

The *profiling analysis* step strongly emphasizes the time consuming aspect of the functions in order to cluster them into two groups based on the required retention time in memory (i.e., long and short). Thus, the data placement at the function granularity can be successfully implemented in the hybrid STT-MRAM architecture. Since the data in the semi-volatile STT-MRAM array may suffer from not only the retention failure but also the read disturb, the data placement decision should be made based on two criteria, i) the execution time, which guarantees virtually zero retention failure rate of the stored data in both semi- and non-volatile arrays, and ii) the read intensive pattern for virtually zero read disturb probability in the semi-volatile region. Hence, the maximum read access rate associated with the semi-volatile STT-MRAM that should not be exceeded is referred to as *read-threshold rate*. Table 3.3 provides the read-threshold rate with respect to Δ values from 10 to 60. It is important to note that retention failure and read-threshold rate of the semi-volatile STT-MRAM have to be accurately estimated under the process variation and operating temperature effects, as explained in Section 2.3.3. Process variation effects need to be determined after the manufacturing process is done, while operating temperature impacts are considered by the proposed compiler-based data placement algorithm.

Based on the profiling-based and compiler-level analysis, the re-compilation process will guide the data of functions to be allocated in the semi-volatile STT-MRAM array if the following conditions are met:

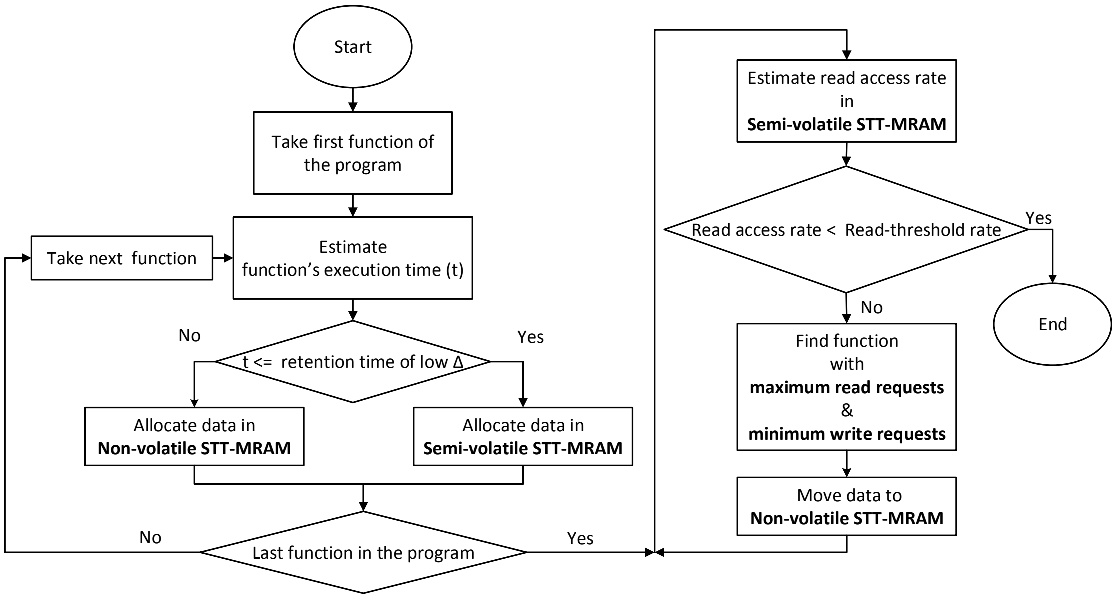

Start
Take first function of the program
Estimate function's execution time (t)
Take next function
t <= retention time of low Δ
No
Yes
Allocate data in Non-volatile STT-MRAM
Allocate data in Semi-volatile STT-MRAM
Last function in the program
No
Yes
Estimate read access rate in Semi-volatile STT-MRAM
Read access rate < Read-threshold rate
Yes
End
No
Find function with maximum read requests & minimum write requests
Move data to Non-volatile STT-MRAM

Figure 3.6: Decision diagram of data placement policy in hybrid STT-MRAM architecture

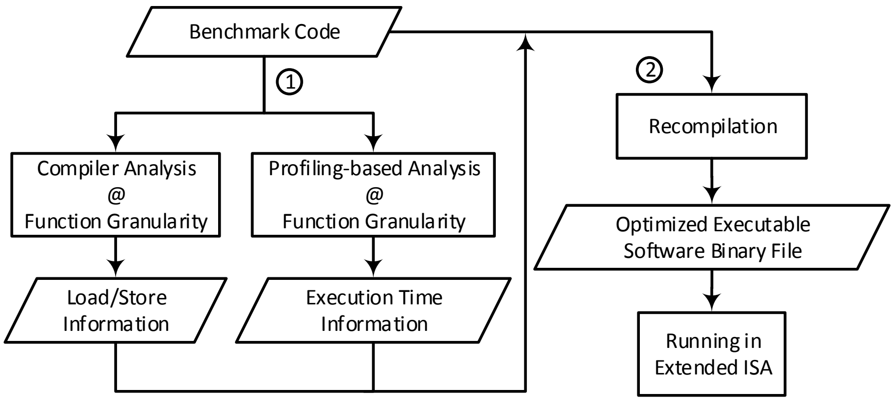

Benchmark Code
①
②
Compiler Analysis @ Function Granularity
Profiling-based Analysis @ Function Granularity
Load/Store Information
Execution Time Information
Recompilation
Optimized Executable Software Binary File
Running in Extended ISA

Figure 3.7: Used Compiler framework

- The execution time of the function is less than the retention time of the adopted low Δ in the semi-volatile STT-MRAM.
- The data access rate during the entire program execution time is less than the read-threshold rate of the semi-volatile STT-MRAM.

In case of the first condition violation, the related data should be simply retained in the non-volatile STT-MRAM blocks, while the rest data of the program functions will be initially placed in the semi-volatile STT-MRAM blocks as long as the limitation of the second condition is not exceeded. Otherwise, the data, which is initially assumed to be placed in the semi-volatile STT-MRAM blocks, needs to be redistributed over the hybrid cache blocks. The data redistribution is implemented based on the intensive read and write data pattern classification. The data with highest read requests and lowest write requests will be retained in the non-volatile STT-MRAM array instead of the semi-volatile STT-MRAM one. Figure 3.6 is the proposed diagram for the data placement process.

In order to obtain safe and accurate compiler-based data placement approach for hybrid STT-MRAM cache design, the longest retention time related to the worst-case execution time for functions has to be predicted, which is longer than any required execution time in reality. Such predicted time is difficult to be obtained by the profiling-based analysis. Thus, our design margins of retention time for both semi- and non-volatile STT-MRAM arrays are estimated precisely for very low error probability at architecture-level, which ensures that the provided retention time meets every possible execution of the function on the real platform. An FIT rate of 10 has been considered for on-chip memories as a typical FIT rate [78] . This makes the estimated retention failure less than 10^{-3}. On the other hand, the target read disturb and write error probabilities have been reduced down to 10^{-23} and 10^{-18}, respectively.

Conventional hardware-managed caches require a tag array or tag comparison logic for data accessing. Therefore, addressing policy tuned by the read/write hits and misses, which is normally adopted for dynamic run-time data management in hybrid caches, does not work with static data placement (i.e., our proposed compiler-based data placement algorithm). Consequently, in this idea, the data placement policy can be implemented with two different approaches. Firstly, as the most processors architectures are supported with an expandable instruction set architecture (ISA), the memory access instructions can be simply extended with two more load and store instructions. Each of them is specialized to store/load data from the main memory locations in a specific cache type. Then, the assembler will enforce this partitioning by translating it into micro-code. This approach has no performance or area overheads imposed on the processor design. For instance, adopting specialized load/store instructions in the write-back stage in the pipeline of the processor will not change the hardware design or the critical path timing in the pipeline. Secondly, the data distribution can be done by modifying the physical addresses (i.e., the memory locations) of the data at compile-time via data-remapping algorithm. The compiler needs to be supplied additionally with the cache-array size and cache-line size, which is equal to the memory-block size. Consequently, the compiler will be optimized to reorganize the data in the memory by modifying its addresses based on our profiling-based and static-code analysis. Afterwards, the linker will translate the data remapping into executable code. This in turn enforces the data partitioning over the semi- and non-volatile STT-MRAM cache lines at run-time. The flow chart of the proposed profiling-based and compiler level analysis steps is illustrated in the Figure 3.7.

3.2.2 Proposed Cross-Layer Framework

Our proposed idea is evaluated through a circuit-level analysis in SPICE all the way up to the system-level analysis in a performance simulator (gem5).

For the bit-cell characterization, we run a circuit-level analysis for L1 and L2-caches in SPICE, based on TSMC 65 nm transistor models and the perpendicular STT-MRAM model presented in [120]. The circuit-level results are fed to NVSim [121] to obtain the read and write latencies of the memory word for L1 and L2-caches to extract the architecture-level results.

At System-level we have two sub-components: profiling-based with compile-time analysis and run-time analysis.

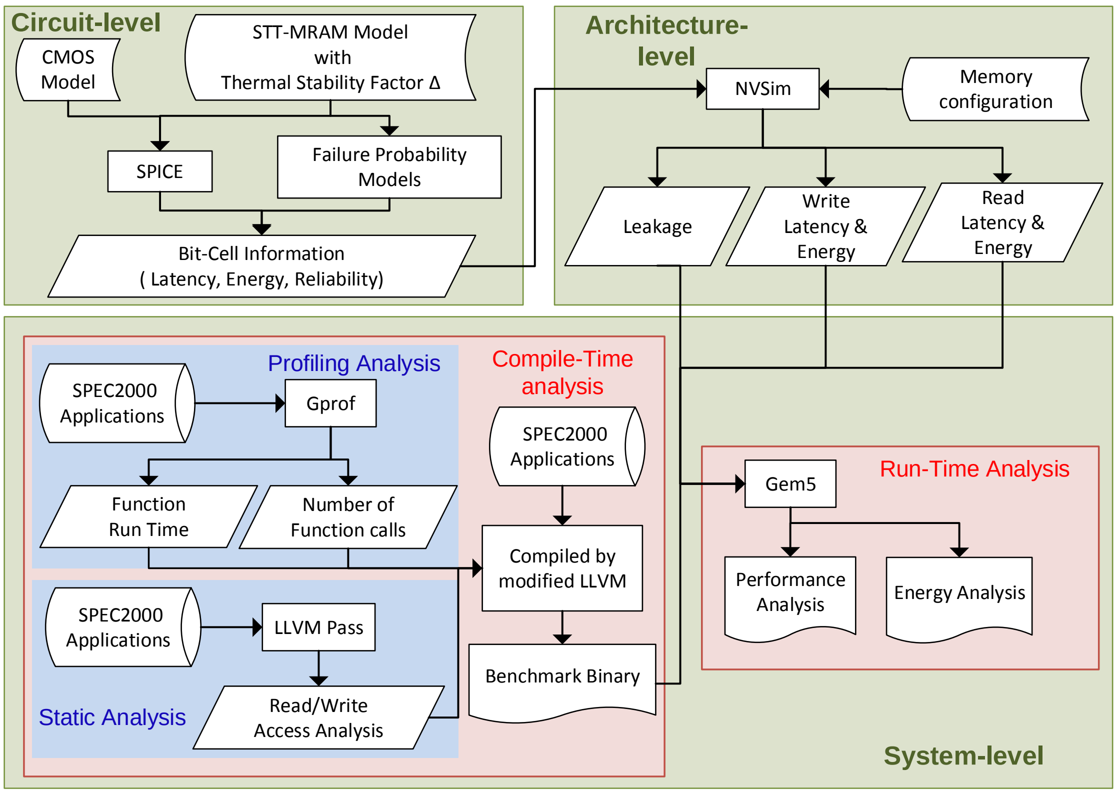


Figure 3.8: Proposed cross-layer tool flow supported with compiler-assisted analysis

- For profiling-based and compiler-level analysis, first, SPEC2000 benchmarks are compiled using LLVM [119]. Afterwards, the binary output files are run on Gprof performance tool [122]. As a result, the function information, like execution time and number of calls, are obtained. Thirdly, we use LLVM Pass framework for the memory access analysis. We write LLVM pass to count the store, load and alloc instructions at function granularity. Then, LLVM pass is run with Clang. Finally, the benchmarks are re-compiled by the modified LLVM compiler, which rearranges the data layout based on the extracted retention time and memory access analysis.
- Run-time analysis is conducted using system simulator (i.e., gem5) [118] in order to evaluate the impact of the proposed hybrid STT-MRAM cache compared to the standard non-volatile STT-MRAM cache on the overall system performance, reliability and energy consumption.

Figure 3.8 illustrates our proposed cross-layer framework.

3.2.3 Simulation Result

Since the proposed approach depends on the application requirements, we provide a detailed requirements analysis based on the hybrid STT-MRAM architecture. Afterwards, for the placement decision, the profiling and compiler level results for a set of SPEC2000 benchmarks are explained in detail. For run time analysis, we first explain the experimental setup. Then, for the total system gains at run-time, the overall performance and energy measures are reported.

Table 3.4: Read-threshold rate for semi-volatile STT-MRAM with (Δ=30) for various temperature values

Temperature	Read rate	Temperature	Read rate	Temperature	Read rate
$-20°C$	4×10^{-6}	$+30°C$	152×10^{-6}	$+80°C$	1995×10^{-6}
$-10°C$	10×10^{-6}	$+40°C$	273×10^{-6}	$+90°C$	3059×10^{-6}
$0°C$	21×10^{-6}	$+50°C$	469×10^{-6}	$+100°C$	4621×10^{-6}
$+10°C$	42×10^{-6}	$+60°C$	787×10^{-6}	$+110°C$	6774×10^{-6}
$+20°C$	82×10^{-6}	$+70°C$	1272×10^{-6}	$+120°C$	9709×10^{-6}

3.2.3.1 Hybrid Architecture Requirements Analysis

For our proposed hybrid-STT-MRAM data cache, the majority of the blocks should be based on the semi-volatile STT-MRAM technology for efficiency, while only a small part of them is non-volatile STT-MRAM-based to guarantee reliability. Based on the profiling information of SPEC2000 benchmarks, which provides an approximated vision of the required retention time in the relaxed STT-MRAM, 26 ms meets the life-time of the most data along with a sufficient margin for the operating temperature and the process variations in order to guarantee the retention requirements. However, the residency time for a few addresses can reach up to tens of seconds.

On the other hand, the retention time in the STT-MRAM cache can be defined by considering a specific retention failure rate. For the typical on-chip memories FIT rate, which is 10 failures in 1 billion hours [78], the target retention failure rate has to be less than 10^{-3}. By considering Δ of 30 and 60 for semi-volatile and the non-volatile STT-MRAM, the maximum achieved retention time under process variation and nominal operating temperature of $+30°C$ is 26 ms and 10 years, respectively. Figure 3.9 illustrates the degradation of the retention time or the increase in the retention failure rate in the semi-volatile and the non-volatile MTJ cells. The read-threshold rate in the semi-volatile array of the cache in the operating temperature range of $[-20°C, +120°C]$ is illustrated in Table 3.4 under 5% process variation as discussed in Section 2.3.3. For the nominal operating temperature of $+30°C$, the read access is limited to 152×10^6 read requests.

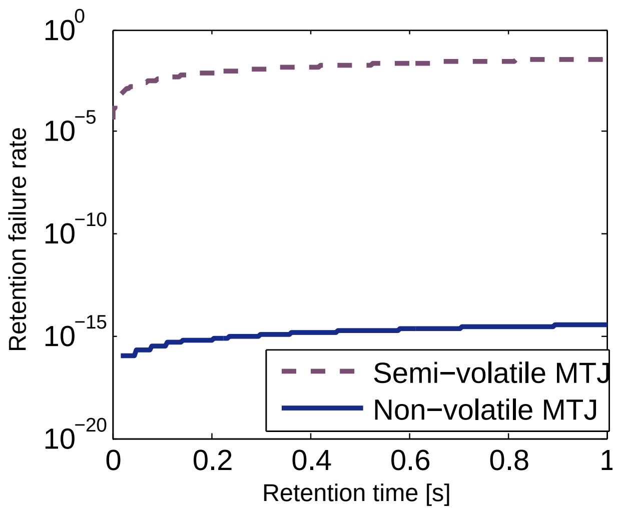


Figure 3.9: Retention time along with the retention failure rate for semi-volatile and non-volatile MTJ considering effects of 5% of process variation

3.2.3.2 Profiling-based and Compiler-level Analysis

We used 16 benchmarks from the SPEC2000 benchmark suits [123]. The C programs of the selected benchmarks are run with gprof to evaluate the required retention time. At the same time, the C programs are compiled with LLVM, and the platform-independent Intermediate Representation (IR) are obtained [119]. We applied the LLVM analysis and transformations on the obtained IR codes through LLVM passes. LLVM pass is written to count the number of load, store and alloc instructions executed in each function. As the number of functions of the benchmark may exceed 2,260 functions such as in gcc benchmark, we focused in our analysis on the main functions that consume more than 90% of the entire execution time.

The required retention time (t) is used as a primary criterion for selecting the cache block (i.e., semi-volatile or non-volatile) to allocate the data. However, the extracted numbers of load and store instructions are considered as the second factor for the data placement.

As previously mentioned, for the semi-volatile STT-MRAM blocks, the maximum reliable retention time is 26 ms and the threshold value of the read access rate is 152×10^{-6}. By considering these two criteria, Figure 3.23 and 3.24 illustrate the adopted placement strategy for effective data placement in the proposed hybrid STT-MRAM cache in order to achieve a low power and reliable design.

Table 3.5 illustrates the details about the functions whose execution time is longer than the retention time of the semi-volatile STT-MRAM blocks (i.e., longer than 26 ms). In this table, about 15.53% of the program functions, on average, requires long time to be executed and hence the related data items need longer retention time. However, based on our run time observations, as discussed in Section 3.1, the percentage of the data which has long vulnerable time is around 5% only. This is because of either the application run time behavior or the cache replacement policy. These two observations are not possible to be predicted at compile-time, and even considering these observations at run time is very complex and expensive. Therefore, we adopt only profile-based and compiler-level analysis as a worst but conservative case.

Table 3.6 shows the functions of the workloads, where the read access rate exceeds the read-threshold value in the semi-volatile STT-MRAM resulting in read disturb. For SPEC2000 benchmarks, almost 9.66% of the functions, on average, is exposed to the read disturb, if the semi-volatile STT-MRAM is adopted for data cache. Consequently, for the L1 data cache and based on the profiling and compile-time analysis, we can assume that around 75% of data (Table 3.7) can safely be placed in semi-volatile STT-MRAM blocks, since they will not suffer from retention or read disturb failures. However, there is as a maximum 25% of the data (Table 3.7) in L1 data cache, which needs to be supported against the potential failures. This can be achieved either by employing a non-volatile STT-MRAM for this part of data or by protecting them with a robust error correcting code [59, 61].

For our proposed hybrid cache solution, the capacity of the semi-volatile STT-MRAM can be 3 times as big as the size of the non-volatile STT-MRAM, based on Table 3.7 results. Therefore, for 32 KB hybrid-STT-MRAM data cache with 64 byte line-size, 24 KB and 8 KB are considered as semi-volatile STT-MRAM ($\Delta = 30$) and non-volatile STT-MRAM ($\Delta = 60$), respectively.

Table 3.5: Functions with execution time longer than 26 ms

Benchmark	Function	Execution time [ms]	Percentage over entire program
gcc	yyparse	589.34 ms	1.67%
bzip2	generateMTFValues	41 ms	54.2%
gzip	deflate	451 ms	9.64%
	deflate-fast	353 ms	2.32 %
mcf	price-out-impl	105 ms	7.43%
	primal-net-simplex	575 ms	0.07%
	write-circulations	220 ms	0.25%
parser	initialize-memory	200 ms	0.12%
wupwise	initialize-memory	710 ms	2.13%
	rndcnf	350 ms	1.42%
	muldoe	430 ms	0.71%
swim	inital	30 ms	9.83%
	MAIN	320 ms	3.12%
mgrid	inital	50 ms	66.21%
ammp	mm-fv-update-nonbon	1810 ms	44.36%
	f-nonbon	150 ms	5.15%
	tpac	1580 ms	0.98%
	fv-update-nonbon	130 ms	0.25%
	AMMPmonito	290 ms	0.25%
lucas	mers-mod-square	2140 ms	38.32%
average	15.53%		

3.2.3.3 Run Time Analysis

3.2.3.3.1 Simulation Setup At system-level, the evaluations are performed using various applications of the SPEC2000 benchmark suite [123]. For each application, the simulation is conducted over 0.2 second of the run time. Table 3.8 summarizes the set-up for these experiments.

3.2.3.3.2 Performance Analysis Indeed, the performance improvement of the hybrid STT-MRAM cache is associated with the hit rate improvement and the reduction of the write access latency because of lowering Δ from 60 down to 30. The profiling-based and compiler analysis are used to approximately calculate the average of the write margin reduction per access due to relaxing Δ in the hybrid STT-MRAM cache design. Table 3.9 shows the impact of the proposed approach translated into reduction of the write margin, which reaches up to 12%, such as with applu, fma3d and art benchmarks.

On the other hand, the performance is affected by the hit rate improvement as discussed in Section 3.1. Hence, by lowering the Δ from 60 down to 30 for the same hardware area, the 32 KB data cache will increase up to 35 KB. This in turn can increase the total performance up to 8.1%.

3.2.3.3.3 Energy Analysis Energy analysis is conducted based on the dynamic and static energy-efficiency of the proposed hybrid-STT architecture. As the static energy is the product of leakage and time, the 10.8% performance optimization leads to the same ratio of reduction in the static energy in the overall system.

Table 3.6: Functions with read rate higher than the read-threshold in the semi-volatile STT-MRAM

Benchmark	Function	Percentage over entire program
parser	hash	16.36%
	table-pointer	12.83%
	form-match-list	7.81%
	xfree	5.68%
	prune-match	4.76%
	xalloc	3.62 %
	region-valid	3.57%
	match	3.24%
gzip	longest-match	56.1%
	ct-tally	1.93%
	send-bits	1.65%
mcf	compute-red-cost	3.3%
	bea-compute-red-cost	2.13%
	bea-is-dual-infeasible	1.62%
art	g	1.18%
quake	phi2	3.13%
wupwise	zaxpy	20.33%
	dcabs1	0.83%
mesa	get-1d-texel	6.67%
ammp	a-next	0.25%
average	9.81%	

Table 3.7: The percentage of workloads partitioning over the proposed hybrid STT-MRAM architecture

Benchmark	mesa	applu	mcf	twolf	gcc	gzip	bzip	parser	wupwise
Semi-volatile	93.33%	100%	85.2%	100%	98.33%	28.36%	45.8%	57.99%	74.58%
Non-volatile	6.67%	0%	14.8%	0%	1.67%	71.64%	54.2%	42.01%	25.42%
Benchmark	**swim**	**mgrid**	**art**	**equake**	**ammp**	**lucas**	**fma3d**	**average**	
Semi-volatile	87.05%	33.79%	98.82%	96.87%	48.76%	62.68%	100%	75.8%	
Non-volatile	12.95%	66.21%	1.18%	3.13%	51.24%	38.32	0%	24.2%	

However, the key factor that influences the STT-MRAM data cache energy is the write energy, since the dynamic access energy in the L1 cache is the major contributor in comparison to the leakage energy. As seen in Table 3.10, the write energy can be optimized per access by 49.7%, on average, and hence, the dynamic energy consumption (read and write) of the hybrid STT-MRAM architecture is reduced, on average, by 51%, as illustrated in Figure 3.10.

3.2.3.4 Discussion

The proposed hybrid STT-MRAM architecture, which is supported with the profiling-based and compiler analysis, has been implemented in this chapter for L1 data cache. However, our idea is applicable for any on-chip cache level for better energy-efficient, reliable and high performance design of the STT-MRAM-based memory.

Table 3.8: Simulation setup

gem5 configuration	ISA ALPHA
Processor	Single-core, 2 GHz, Out-of-order, 4-issue
L1 instruction cache	32 KB, 4-way set associative, 64 Byte line size
L1 data cache Semi-volatile STT-MRAM	24 KB, 4-way set associative, 64 Byte line size $\Delta = 30$ with 5% process variation (10.8 ns/0.988 ns) write/read latencies
L1 data cache Non-volatile STT-MRAM	8 KB, 4-way set associative, 64 Byte line size $\Delta = 60$ with 5% process variation (11.15 ns /0.988 ns) write/read latencies
SPEC2000 Applications	gzip, bzip2, mcf, twolf, gcc, mesa, art

Table 3.9: Write latency reduction per access of the proposed hybrid cache

mesa	applu	mcf	twolf	gcc	gzip	bzip	parser	wupwise
11.1%	12%	10.2%	12%	11.8%	3.4%	5.5%	6.9%	8.9%

swim	mgrid	art	equake	ammp	lucas	fma3d	average
10.4%	4%	11.9%	11.6%	5.8%	6.5%	12%	9%

Table 3.10: Write energy reduction per access of the proposed hybrid cache

mesa	applu	mcf	twolf	gcc	gzip	bzip	parser	wupwise
61.2%	65.6%	55.9%	65.6%	64.5%	18.6%	30%	38%	48.9%

swim	mgrid	art	equake	ammp	lucas	fma3d	average
57%	22.2%	64.8%	63.5%	31.9%	40.1%	65.6%	49.7%

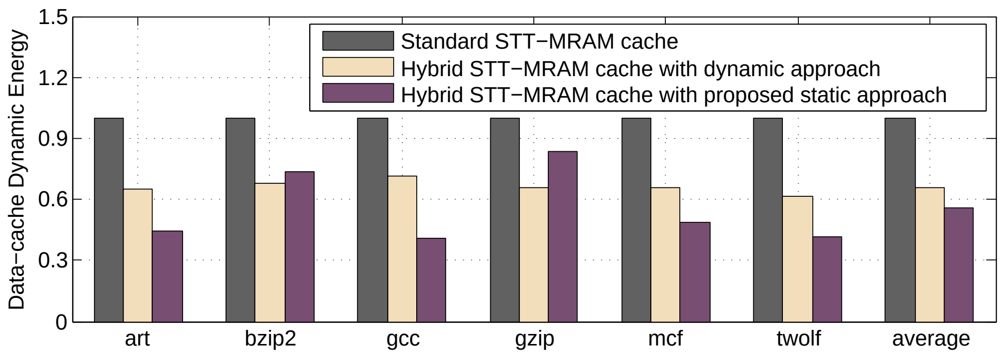


Figure 3.10: Total dynamic energy overhead (read+write) of the hybrid STT-MRAM data-cache supported with the proposed static data placement for various SPEC2000 workloads compared to the dynamic data placement and the standard non-volatile STT-MRAM data-cache

3.3 Dynamic Behavior Predictions for Fast and Efficient STT-MRAM On-Chip Cache Design

Cache access requirements due to the application read/write behaviors vary over the run-time, and from one application to another. For instance in SPEC2006 benchmarks, there are write intensive benchmarks (e.g., lbm), read intensive benchmarks (e.g., gcc and solex), and alternately read and write intensive phases benchmarks (e.g., omnetpp, see Figure 3.11). The read or write phases are defined as a period of execution time

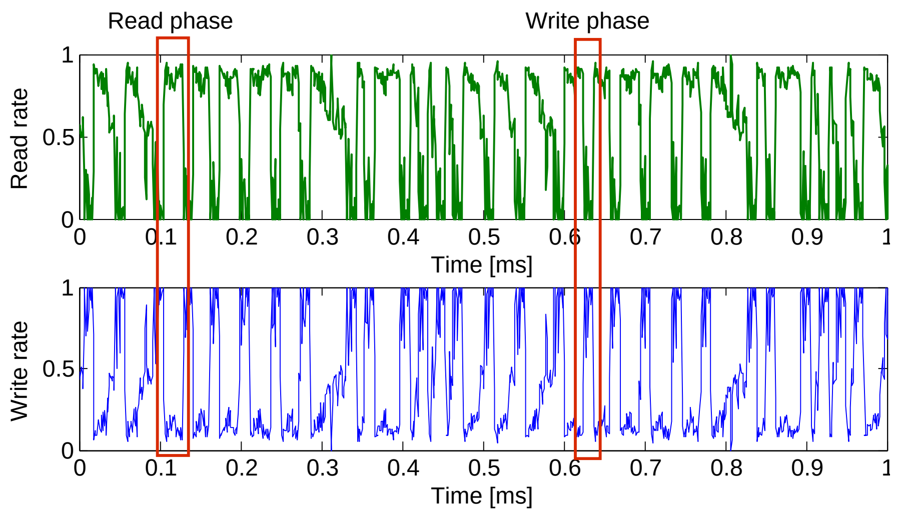


Figure 3.11: Read and write behavior of omnetpp application

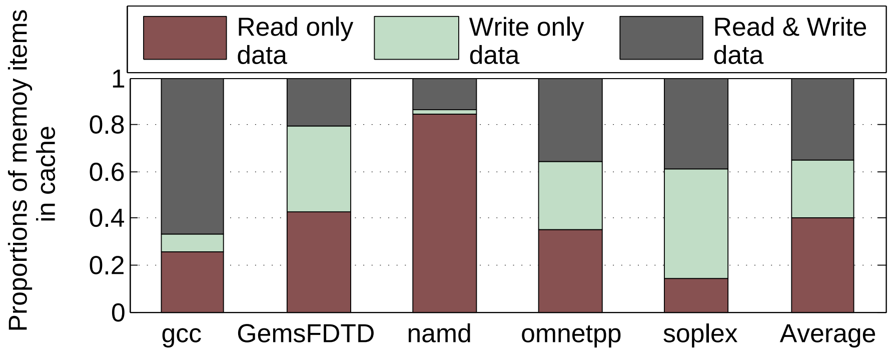


Figure 3.12: Read and write intensive data proportions in L1 data-cache lines for different SPEC2006 applications

that exhibits a consistent read or write access rate which is distinguishable from the rates in other phases, as shown in Figure 3.11. The read or write intensive phase (or application/benchmark) refers to the application (phase) in which the cache is mostly accessed by read or write requests from the CPU side, respectively.

However, for a hybrid STT-MRAM cache, it is not enough to consider only the read and write behaviors of the entire cache as a whole. Although this may provide hints about the read and write access requirements from the application perspective, the behavior of each memory item (data) in the cache is required to be analyzed individually. Figure 3.12 shows that the read and write patterns vary significantly in a single cache line for different memory items along with the run-time in one application. As seen in Figure 3.12, more than 65% of the SPEC2006 data in the cache lines is mostly accessed by either only read or only write requests.

Our main contribution in this section is to exploit *Machine Learning Approaches* at application-level for simple but effective data-pattern classification and prediction in a hybrid STT-MRAM memory architecture, where the migration costs will be reduced at low hardware overheads in comparison to existing static, on-the-fly, or compile-time data management methods. Three active dynamic behavior prediction approaches for the cached data are proposed in this work with a prediction accuracy of 75%, on average.

At system-level, the performance and energy consumption gains of the hybrid STT-MRAM data-cache supported with a supervised dynamic behavior learning and prediction mechanism is evaluated using detailed simulations of SPEC2006 workloads. The system results demonstrate that the proposed hybrid STT-MRAM cache provides dynamic cache energy (read and write) reduction and performance optimization, on average by 38.2% and 11.2%, respectively, in comparison to a standard NVM STT-MRAM cache. However, compared to the state-of-the-art [124], where hybrid SRAM with standard NVM STT-MRAM cache architecture is adopted, the total energy consumption is reduced 10.5%, on average, with hybrid SVM/NVM STT-MRAM cache.

3.3.1 Hybrid STT-MRAM Architecture

Based on the device-level and application-level observations, we adjust a hybrid STT-MRAM architecture with $N1$-ways SVM array and $N2$-ways NVM array, as proposed in [64, 124]. This hybrid STT-MRAM architecture is mainly employed in order to achieve a fast and energy-efficient STT-MRAM write operation along with a reliable STT-MRAM read operation, as the write intensive data can be retained in the the SVM array, while the read intensive one are mapped to the NVM blocks.

For the hybrid STT-MRAM cache, run-time prediction approaches to identify the data behavior of each memory item (address) in the cache for the near or far future are required. However, it is difficult to predict the data behavior only from the first access. Therefore, the cache-miss information is proposed for the first allocation of the data in the cache. All read-miss and write-miss blocks will be allocated to NVM and SVM, respectively. Therefore, DEMUX is added between the current level of cache (e.g., L1 data-cache) and the higher level (e.g., L2 data-cache), as illustrated in Figure 3.13(a).

In case of cache-hit, where both cache ways (SVM and NVM) will be accessed simultaneously from the lower-level (e.g., CPU), the cache-hit information is used to guide the prediction method and hence the migration scheme. Hit information consists of i) the type of access (read or write) and ii) the type of accessed way (SVM or NVM). Based on

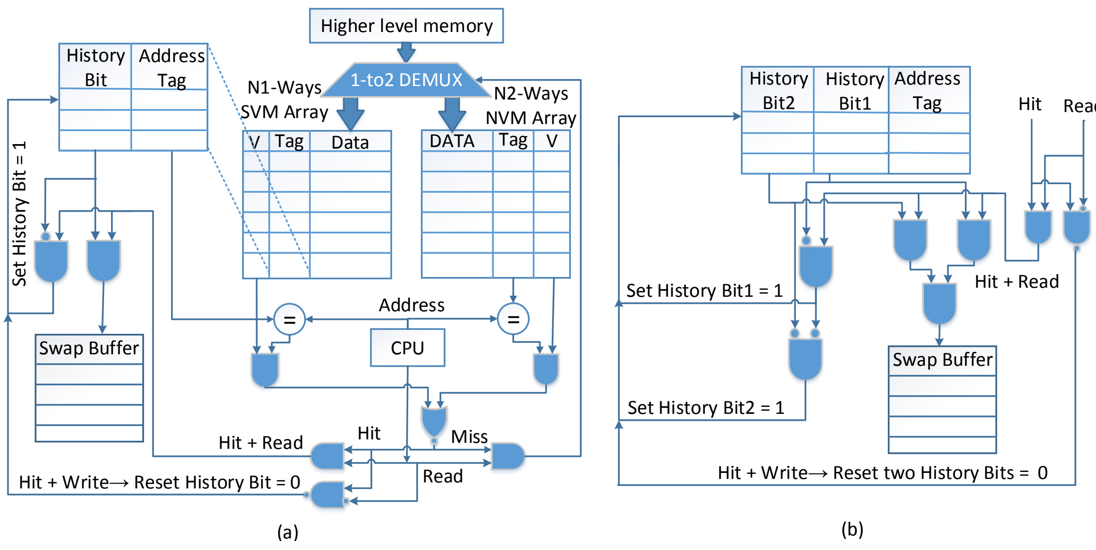


Figure 3.13: The proposed hybrid STT-MRAM architecture supported with dynamic data palcement scheme

this information, a control signal will be activated to migrate the data between different ways of cache via Swap Buffer 3.13(a).

Figure 3.13(a) and (b) illustrate further information about the architecture of one proposed prediction scheme in case of read cache hit access (see Section 3.3.2.1.3). Further details of all proposed prediction approaches are explained in the next section.

3.3.2 Dynamic Cache Accesses Prediction for hybrid cache data management

In the hybrid cache architecture, write rate is considered as the criterion for data migration between the SVN and NVM STT-MRAM blocks. Monitoring write rate for all data in the cache can be classified into two categories, namely i) static-code or compiler-assisted write rate analysis and ii) on-the-fly write rate estimation. Although the first category exhibits less run-time and hardware overheads, as the data placement is totally done at compile-time [64], it requires complex extensions to ISA, and it is application dependent under specific operating conditions. Thus, the first category (i.e., compiler-assisted write rate analysis) cannot deal with normal dynamic changes of data behavior. For on-the-fly write rate estimation, two approaches can be used, namely static and dynamic write behavior prediction of the cached data. *Static on-the-fly behavior prediction* [124] refers to setting up specific and fixed mechanisms (e.g., fixed write rate) for the entire memory or each cache line based on the historical access information, as it is assumed to be repetitive and identical in the next execution-time. Approaches based on static behavior predictions are relatively simple to be implemented, but the flexibility is very low, as it is highly application dependent. Therefore, it has low accuracy and leads to frequent data migrations, resulting in high overheads in the system performance and energy. *Dynamic on-the-fly behavior prediction* refers to predicting the next write rate based on the updated supervised run-time data behavior, hence the prediction accuracy is higher than the static approach.

In this work, depending on the supervised learning data-pattern classification, three dynamic prediction approaches are proposed for write pattern predictions of each memory item in the data-cache. Furthermore, the erroneous data induced due to the misprediction (i.e., due to a wrongly allocated read-intensive data to SVM block, which can lead to read disturb) can be mitigated with an *Error Correcting Code* (ECC), whose strength is based on the read disturb probability of the SVM blocks, as explained in Section 3.3.2.3.

3.3.2.1 Proposed Dynamic Prediction Approaches

3.3.2.1.1 Phase prediction based on Fair Discrete Sequential Prediction In this approach, we try to predict the future accesses using the phase phenomenon. The input is a finite-order of cache accesses. The task is to find statistical regularities in the input so that the ability to predict the next stream of accesses for future time (i.e., next phase). Although the read/write access behavior of each data-cache line is varying over the run-time, for certain durations, this data could exhibit a consistent behavior that can identify the data behavior phases (i.e., read or write phases). Based on the phase prediction, the next predicted accesses should also be consistent, until the phase switches. In the Fair Discrete Sequential Prediction, the conditional probability for the

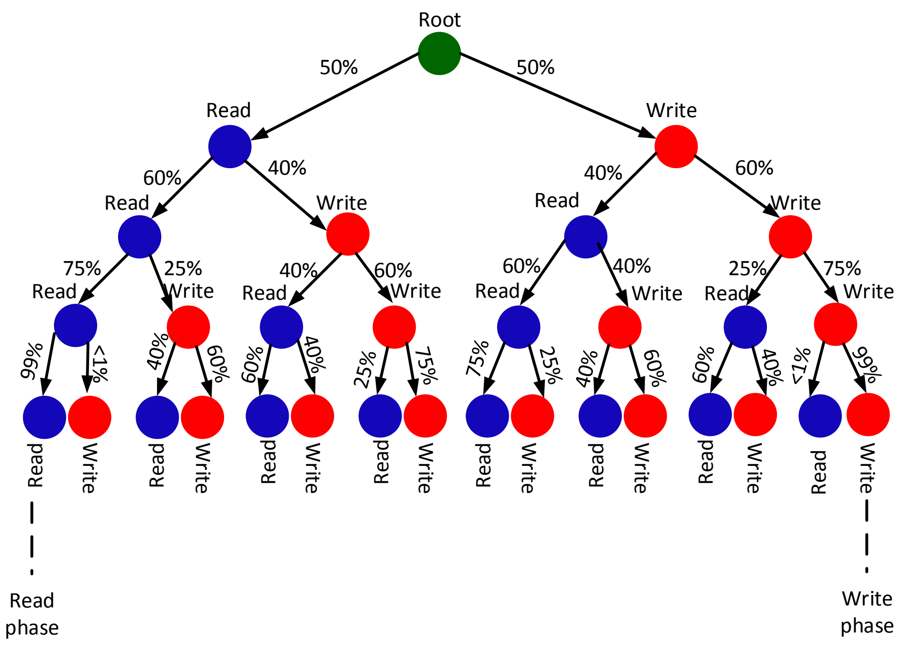


Figure 3.14: The steady-state probability tree for access phase prediction

sequences of cache accesses is required to be calculated for identifying phase switching or the class of next cache accesses, where the next phase or class of cache accesses is predicted based on the corresponding access as follows:

$$P(y_n \mid x_n, D_{n-1}) = \int P(y_1 \mid x_1, \theta) \ldots P(y_{n-1} \mid x_{n-1}, \theta)\, P(\theta)\, d\theta \tag{3.1}$$

where, y_n is the next predicted phase based on the corresponding access x_n and previous sequence of accesses D_{n-1}

$$D_{n-1} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{n-1}, y_{n-1})\} \tag{3.2}$$

The conditional probabilities of subsequent read and write data accesses in cache are easy to be calculated and presented in form of a probability tree. The levels of the probability tree are shown in Figure 3.14. The root node corresponds to the lack of any information about type of data access and all nodes below the root have the same probability distribution. Each path of the tree represents a unique sequence of accesses corresponding to the label on the node of the tree, while edges are labeled with transitions probabilities.

A case study has been illustrated in Figure 3.14, where three sequential identical accesses for a data in the cache determines type of the next accesses (i.e., phase) for that data, and hence specifies where to map the data in the hybrid cache array. Figure 3.16 shows the phase prediction accuracy for SPEC2006 benchmarks, which reaches up to 85%, while the average accuracy is around 75%.

3.3.2.1.2 Bayes Classifier Machine learning techniques can be used for an efficient data behavior prediction in the cache, where the behavior can be learned to be predicted depending on history access records. Bayes classifier, which is a probabilistic classifier based on Bayes theorem, is proposed to predict the class (i.e., read or write) of the majority of the next sequential accesses. The input features of this classifier are the types of last sequential accesses. The Bayes Classifier assumes that all features are conditionally independent, and any read or write access is conditionally independent of every other accesses. Therefore, the *Naive Bayesian Classification* is considered with a small training dataset in order to decide the most probable class of the next access as the following model:

$$C = Pr(C = c_j) \prod_{i=1}^{n} Pr(X \ = \ x_i \mid C \ = \ c_j) \tag{3.3}$$

where X is the vector of n independent features or the last n sequential accesses, and C is the class of the next accesses.

In order to prepare the training dataset, the desired feature of Bayes Classifier are extracted from tracing the cache accesses. These features are the order and the type of the cache accesses. Training dataset is represented in form of input/output dataset or chain of last accesses/next access patterns. The training dataset takes the format of sets $\langle x_1, x_2, x_3, ..., x_n, c_j \rangle$, where $x_1, x_2, x_3, ..., x_n$ represent the input features, while c_j represents the target output or the classification node. Table 3.11 shows the used dataset with their meanings. Back-ward looking and forward-looking sliding windows of sequence of data accesses in the cache are the time before and after output was made. In our case, the target output or the classification C is performed based on forward-looking sliding window.

Once the training dataset is prepared, the Naive Byes algorithm just requires to estimate the prior probabilities $Pr(C \ = \ c_j)$ and the conditional probabilities $Pr(C \mid X)$, as follows:

$$Pr(C = c_j) = \frac{\# \ of \ class \ c_j}{\# \ total \ outputs} \tag{3.4}$$

$$Pr(X \ = \ x_i \mid C \ = \ c_j) = \frac{\# \ of \ X \ = \ x_i \ and \ class \ c_j}{\# \ of \ class \ c_j} \tag{3.5}$$

$$c_j \wedge x_i = \begin{cases} 0 & for \ read \ access \\ 1 & for \ write \ access \end{cases} \tag{3.6}$$

For five input features along with the major type prediction of the next 5 accesses, up to 84% of the prediction outputs are accurate, while the average of achieved accuracy is around 76% (see Figure 3.16). To better illustrate how the method works, Table 3.11 shows a part of the training database generated by gcc benchmark along with a test data case. In order to predict the output class as read or write, the conditional probabilities for both read and write classes need to be separately calculated, and hence, the highest probability will be considered as a predicted output class. Read operation is represented logically as ‘0’, while logic ‘1’ refers to the write.

3.3.2.1.3 State-based Tow-level Adaptive Training Here we use a similar concept used for branch prediction [125, 126]. Branch prediction schemes can be classified

Table 3.11: Part of the training dataset used by benchmark gcc and test data

	1st last access	2nd last access	3rd last access	Next accesses majority
Training database	1	1	0	1
	0	0	1	0
	1	0	1	0
	1	0	0	1
	0	1	0	1
	0	1	1	0
	0	1	0	0
	0	0	1	1
Test data	1st last access	2nd last access	3rd last access	Predicted accesses majority
	0	0	1	?

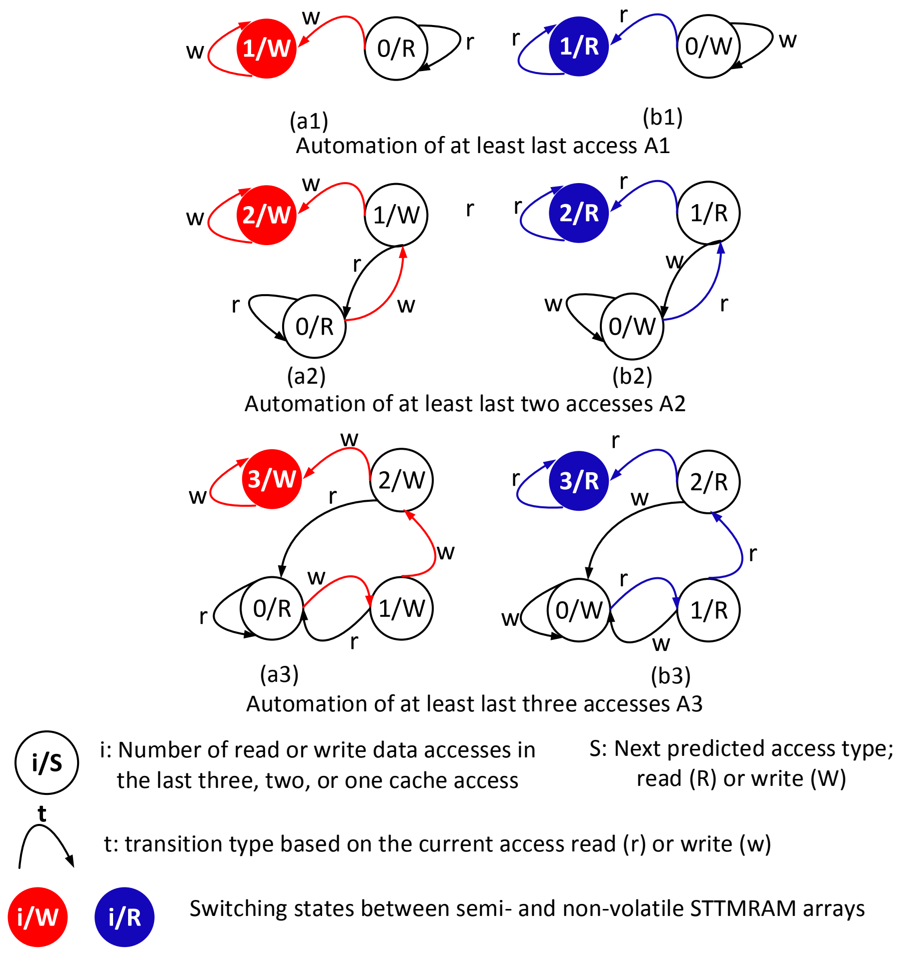

w
1/W
w
0/R
r
r
1/R
r
0/W
w
(a1)
(b1)
Automation of at least last access A1
w
2/W
w
1/W
r
r
2/R
r
1/R
r
w
r
0/R
w
w
0/W
r
(a2)
(b2)
Automation of at least last two accesses A2
w
3/W
w
2/W
r
r
3/R
r
2/R
w
w
r
r
0/R
w
1/W
r
w
0/W
r
1/R
w
(a3)
(b3)
Automation of at least last three accesses A3
i/S
i: Number of read or write data accesses in the last three, two, or one cache access
S: Next predicted access type; read (R) or write (W)
t
t: transition type based on the current access read (r) or write (w)
i/W
i/R
Switching states between semi- and non-volatile STTMRAM arrays

Figure 3.15: State-based predictor

into static and dynamic. Dynamic branch prediction schemes use the knowledge of run-time behavior to make the prediction. 2-bit hysteresis or saturation counter and static training schemes are proposed in [125] to make dynamic predictions. In 2-bit hysteresis or saturation counter, the workload execution history dynamically alters the counter value at branch granularity (called as one-dimensional of a history table with two-bit counters), and hence modifies the branch states. On the other hand, static training scheme uses the history information collected form last n run-time execution to make the prediction.

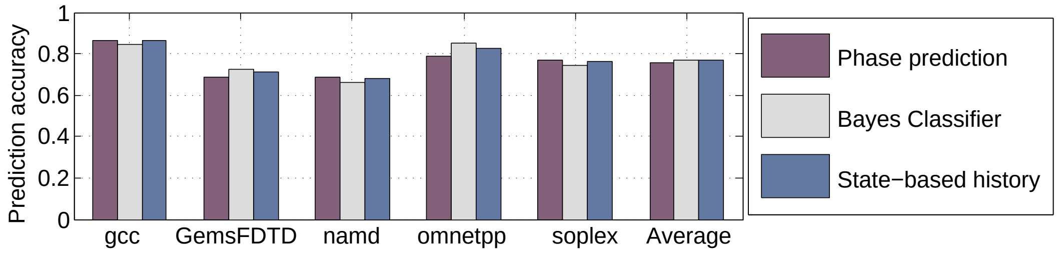


Figure 3.16: Prediction accuracies of the proposed approaches

Dynamic training approach is proposed in [126], where the behavior is predicted in two levels. The first level is the history of branches during the current execution time, thus branch history register is needed for information collection at this level. The second one is for collecting and updating the pattern information of each branch in the history pattern table. In programs, branches are different from each other. Each Branch has a unique behavior that varies according to the implementation conditions. On the other side, the data in the cache is very similar in behavior (i.e., read/write access rates and even required retention time). Therefore, two-level dynamic training approach can be abridged into one-level as all data has similar pattern information, where the history register and history pattern table can be merged into one data structure (i.e., one-dimensional of a history table).

State transition diagrams used in this scheme to predict the data behavior in the cache based on the one-dimensional history table information are shown in Figure 3.15. In this figure, a1, a2 and a3 finite-state machines are used to monitor the data behavior in the non-volatile STT-MRAM based on the number of write accesses, and hence predict when to migrate the data to semi-volatile STT-MRAM array. On the other hand, read pattern is predicted by monitoring the data in semi-volatile array as shown in b1, b2 and b3 state machines.

For automation A1, one miss-prediction is enough to migrate the data between different memory arrays, such as one read access to data in semi-volatile STT-MRAM or one write access to data in non-volatile STT-MRAM. Nevertheless, this scheme has very low prediction accuracy and significant migration overheads.

A2 automation records in one bit the last type of access. If there is a miss-prediction (e.g., read in semi-volatile and write in non-volatile STT-MRAM), this bit will be set to “1”, otherwise “0”. However, for miss-prediction case, if the bit value was already one, which means two consecutive mis-predictions, the accessed data will be moved between the memory arrays, and the bit value will be initialized with “0”.

Finally, A3 automation needs two bits to store the pattern history information of at least last two data accesses, where three consecutive mis-predictions is the criterion for data migration. Nevertheless, A2 scheme has enough accuracy up to 86% to be considered, as shown in Figure 3.16.

3.3.2.2 Proposed Prediction Structure

A simple prediction structure (i.e., history queue) required to be added to the tag array of each cache line. The length of the history queue relays on the adopted prediction scheme for retaining the number of sequential type of last accesses or miss-accesses for the *Phase prediction* (i.e., at least 2 required bits) and the *State-based history prediction* (i.e., at least 1 required bits), respectively. Whereas, in the *Bayes Classifier*, the history queue stores a current test data as shown in Table 3.11. A2 scheme State-based history prediction approach will be considered for the rest of this work because of its high accuracy for only one history bit. Figure 3.13(a) and (b) illustrates the architecture of A2, A3 State-based history prediction scheme in case of read cache hit access, respectively.

3.3.2.3 Handling Mis-predictions

Figure 3.17 presents four prediction classifications produced by each proposed prediction approach: 1) read true-prediction, 2) write true-prediction, 3) read mis-prediction and 4) write mis-prediction. Write mis-prediction, whose proportion is on average 15%, may induce data corruption, as the data is miss-predicted as a write intensive pattern, and hence it is stored in the SVM blocks. The adopted Δ value in SVM satisfies the maximum retention time required to retain data with no retention failure. As a result, the data corruption occurrence in SVM due to the write mis-prediction is only because of the read disturb issue.

The read disturb probability depends on two main factors: 1) the adopted low Δ value in the SVM region, and 2) the number of the sequential read mis-predictions between two write true-predictions. The read disturb induced by a read access followed with another read access is always propagated to the memory output and has a chance to impair the final output of the workload. In contrast, the write access overwrites the possible read disturb of a previous read access.

For write mis-prediction analysis, we identify a read record as a series of read accesses minus one for a particular memory item (data) in SVM cache line. The read record includes the reads between either loading the data into the cache line or modifying it with a write access, and either evicting or modifying it again with another write access. Consequently, each data in the SVM array may have a set of read records. Figure 3.18 shows an example of serial read records, which belong to a memory item in the SVM cache during the workload execution.

A fault generation mechanism is built in order to flip the bit-cell contents at read access of each record in the SVM array randomly following the read disturb probability. The corresponding pseudo-code for fault generation is presented in Algorithm 1. The output is the maximum number of bits flipped per read record, and hence it identifies the required error correction capability. For SPEC2006 applications (gcc, GemsFDTD, namd, onmetpp, soplex), the fault generation scheme is run for 100 times. Consequently, the

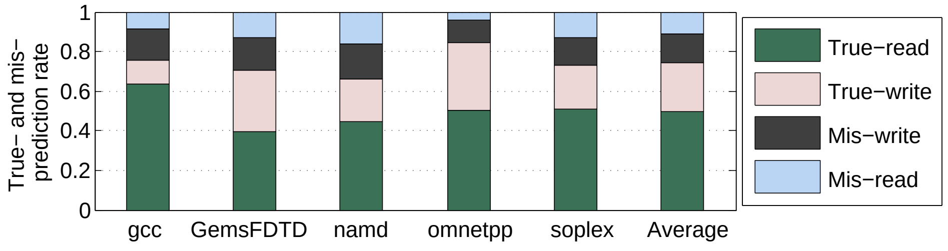


Figure 3.17: True- and mis-prediction rates of the phase prediction approach

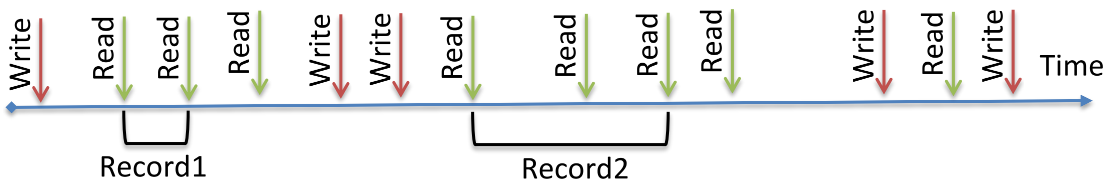


Figure 3.18: Read record analysis in SVM block

Algorithm 1 Fault generation mechanism

```
Input Read disturb probability of SVM STT-MRAM $P_{RD}$
        and the maximum number of experiments (N)
Output The maximum number of flipped bits ($E_{Max}$)
1:  $n \longleftarrow 1$
2:  repeat
3:    For each retained data in SVM array do
4:      For each read record do
5:         Initialize the number of flipped bits ($E \longleftarrow 0$)
6:         For $i$ from 1 to last read in this record do
7:         For each read bit do
8:         Generate random number from distribution range $x\varepsilon[0, 1/P_{RD}]$
9:         if (x == 0) then
10:        Flip the content of the bit
11:        Increase the number of flipped bits by one ($E \longleftarrow E + 1$)
12:        end if   end for   end for
13:        if ($E > E_{Max}$) then
14:        $E_{Max \longleftarrow E}$
15:        end if   end for   end for
16:    $n \longleftarrow n + 1$
17:  until ($n == N$)
18:  return The maximum number of flipped bits ($E_{Max}$)
```

maximum flipped bits per read record is one for GemsFDTD, namd and soplex benchmarks, while there is no read disturb flipped bits in gcc and onmetpp benchmarks. Therefore, supporting an SVM array at each line with a single error correction code guarantees the data integrity induced by the write mis-predictions.

On the other hand, the read mis-prediction, whose proportion is on average 10%, only degrades performance and energy gains. This is because of performing the write operations in the NVM region, which could ideally be done in the SVM region.

3.3.3 Simulation Results

In this section, we explain our cross-layer frame work. Then, we present the experimental analysis to show the benefits of using hybrid STT-MRAM data-cache combining non-volatile and semi-volatile blocks in the scope of machine learning by using state-based history prediction approach to predict the data behavior and to decide the data migration accordingly. Finally, the prediction mechanism overheads along with the performance and energy analysis are discussed to demonstrate the total gain of the proposed

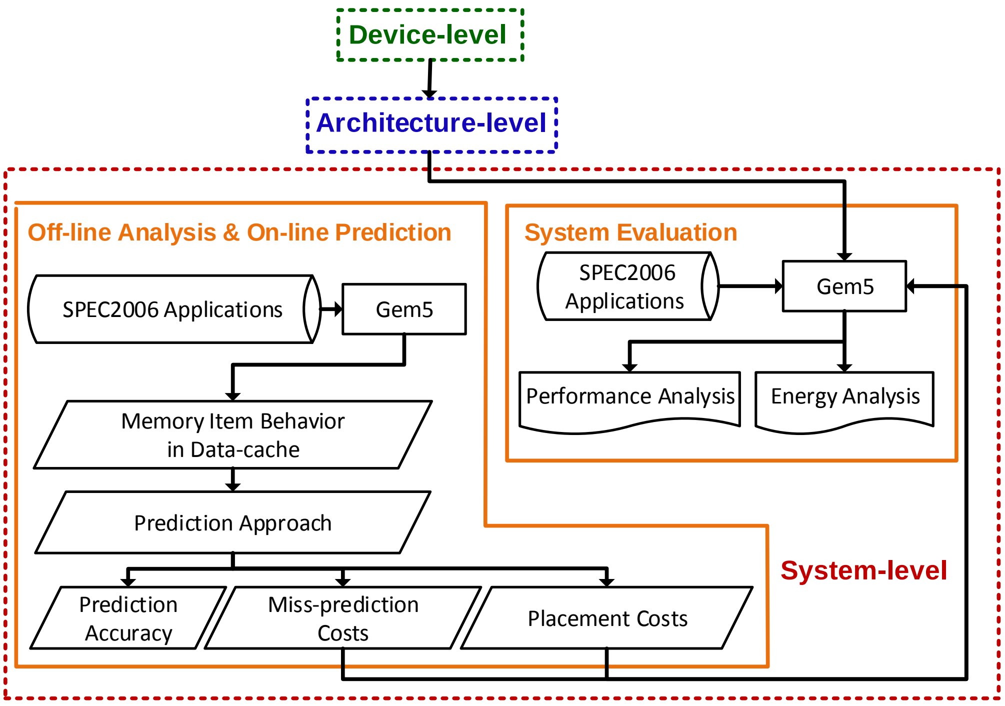


Figure 3.19: Proposed system-level tool flow

hybrid STT-MRAM array for on-chip memories in comparison with the sate-of-the-art RWHCA [124] technique. In RWHCA, the SRAM technology is integrated with standard NVM STT-MRAM to compensate for the effects of inefficient write operations of STT-MRAM, while saturated write access counter is considered for data migrations in the hybrid memory architecture.

3.3.3.1 Cross-Layer Framework and Experimental Setup

Figure 3.19 illustrates our developed cross-layer framework adopted in this idea. It is at device- and architecture-level similar to previous static approach framework(see Figure 3.8). However, the main updates of the dynamic approach have been done at system-level. Table 3.12 summarizes the set-up for the framework and the experiments.

3.3.3.2 Prediction Overheads

The required additional bits for the data behavior prediction approaches are proposed to be integrated in the tag array. For data-cache of 32 KByte with block size of 64 Bytes, the incurred additional bits is less than 1.1% per block (cache line), see Table 3.13. It is also worth noticing that for the Bayes Classifier approach, the training dataset or the covariance matrix needs also additionally to be stored, whose size is driven from the target accuracy. The higher accuracy incurs more input features that induces exponentially larger training dataset. According to our experimental results, the prediction accuracy will no longer increase for more than 5 features. Therefore, for the training data of 5 features, which provides prediction accuracy up to 84% with an average of 75%, a look-up table of 4 Bytes will be reserved for the required additional bits of the covariance matrix.

Table 3.12: Simulation setup

Device level	TSMC 40 nm transistor model Perpendicular STT-MRAM model with radius of 20 nm [120]
Architecture level	NVSim [121] for read and write latencies of SVM and NVM
System level	
Configuration	gem5 simulator with ISA X86 [118]
Processor	Single-core, 2 GHz, Out-of-order, 4-issue
L1 instruction-cache	32 KB, 4-way set associative, 64B line size, SRAM
L1 data-cache for write intensive data	16 KB, 4-way set associative, 64B line size SVM STT, $\Delta = 30$, ECC1, 1.2 ns/8.6 n2 [read+decoding, write+encoding]
L1 data-cache for read intensive data	16 KB, 4-way set associative, 64B line size NVM STT, $\Delta = 60$, NoECC, 0.7 ns/11.6 n2 [read, write]
L2-cache	512 KB, 8-way set associative, 64B line size, SRAM
SPEC 2006	gcc, GemsFDTD, namd, onmetpp, soplex
Execution time	of 1000,000 upto 3000,000 read and write CPU requests After skipping one million warm up instructions

Table 3.13: Latency, energy, area and storage overheads of prediction approach implementations for 23 KB cache

Prediction approach		Time [ns]	Total Power [mW]	Area [μm^2]	Storage [%]
Proposed machine learning approaches	Phase prediction	0.169	0.022	39.33	0.4%
	Bayes Classifier	0.265	0.044	125.77	1.17% + 4 Byte
	State-based history A2	0.096	0.011	18.522	0.19%
	State-based history A3	0.096	0.016	26.989	0.4%
State-of-the-art	RWHCA [124]	0.107	0.0257	41.806	0.4%

On the other hand, the prediction process impairs read/write access latency, while its circuitry imposes energy and area overheads. The proposed three prediction circuitries were synthesized with TSMC 40 nm standard cell library. Table 3.13 shows the corresponding maximum delay, area and power overheads. For a micro-processor with a clock cycle not less than 0.5 ns, the increased cache latency due to the prediction penalty has no impact on the overall system performance, because the cache access with or without prediction approaches requires the same number of the cycles to complete the operation, see the system set-up Table 3.12 for the cache read and write latencies.

However, the *State-based history prediction* approach with one and two bits (i.e., A2 and A3 schemes, respectively) offers the lowest overheads even with a comparison to the sate-of-the-art RWHCA [124] technique. Therefore, we will consider A2 of State-based history prediction approach for estimating the performance and energy gains of the hybrid STT-MRAM cache.

3.3.3.3 Performance and Energy Analysis

Each SVM write over NVM reduces the corresponding dynamic energy and improves the access performance by 72.5% and 35.0%, respectively. At system-level, the performance and energy gains are application dependent, since the ratio of read and write

CPU requests vary over the executed applications. As illustrated in Figure 3.20, the write access proportion is around 42%, on average. Therefore, if all write requests are done in the SVM block, while the read requests are executed in the NVM block, the energy and performance gains can reach up to 57.7% and 17.1%, respectively, as seen in Figure 3.21. However, this amount of the achieved gains is unrealistic, and the actual gain of the proposed approach is less compared to the standard NVM one because of two main reasons. Firstly, the prediction accuracy is around 75%, on average, as reported in Sec. 3.3.2. This results in energy and performance overheads related to the mis-predictions. Secondly, the data replacements costs (i.e., costs of mapping and migrating data between NVM and SVM blocks) incur non-negligible energy overhead, since each migration requires one read and write operation. Consequently, the average actual gains of the data-cache dynamic energy and the system performance provided by the proposed approaches are 38.2% and 11.2%, respectively, as seen in Figure 3.21.

On the other hand, compared the state-of-the-art RWHCA [124], where SRAM technology is integrated with NVM STT-MRAM, and supported with a saturated counter scheme for data migration, our work provides the following advantages: Firstly, thanks to zero leakage current of both NVM and SVM of STT-MRAM technologies, SVM is a more interesting solution than SRAM in the hybrid cache architectures. As seen in Figure 3.22, the total energy consumption (static and dynamic) of hybrid SVM/NVM STT-MRAM cache reduces compared to hybrid SRAM/NVM STT-MRAM cache, on average, around 10.5%. Secondly, thanks to the short read pulse and low read energy of SVM technology, its dynamic energy consumption and performance are similar and very close to SRAM. This is especially important, as the hit rate can be improved up to 13% due to increase the size of 16 KB data cache around 2 KB under the same area constraints. Finally, it is important to note that the prediction accuracy for some benchmarks will increase significantly, if we replace our proposed A2 state-based history prediction with A3 scheme, where three consecutive mis-predictions are required to predict the data behavior for less frequent migrations. As shown in Figure 3.16, gcc and soplex benchmarks exhibit better prediction accuracies with Phase prediction (i.e., three sequential identical accesses) than A2 State-based scheme (i.e., two consecutive mis-predictions). Consequently, the proposed state-based scheme can be implemented in away of integrating A2 and A3 schemes together in a one design, which masks it, unlike RWHCA method, adaptable to the prediction accuracy variation over the executed workloads or execution time.

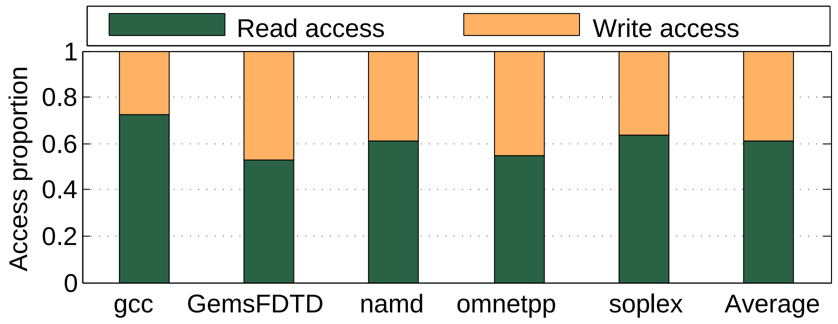


Figure 3.20: Proportions of read and write accesses for various SPEC2006 benchmarks

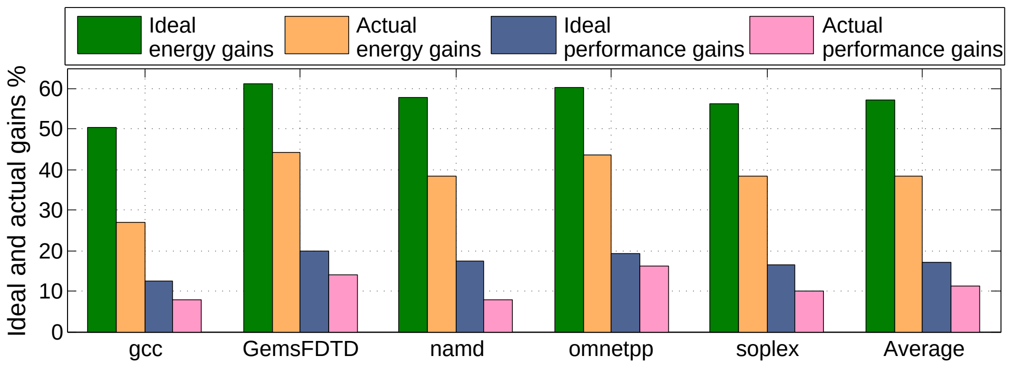


Figure 3.21: Ideal and actual performance and energy gains of the proposed hybrid STT-MRAM data-cache compared to standard one

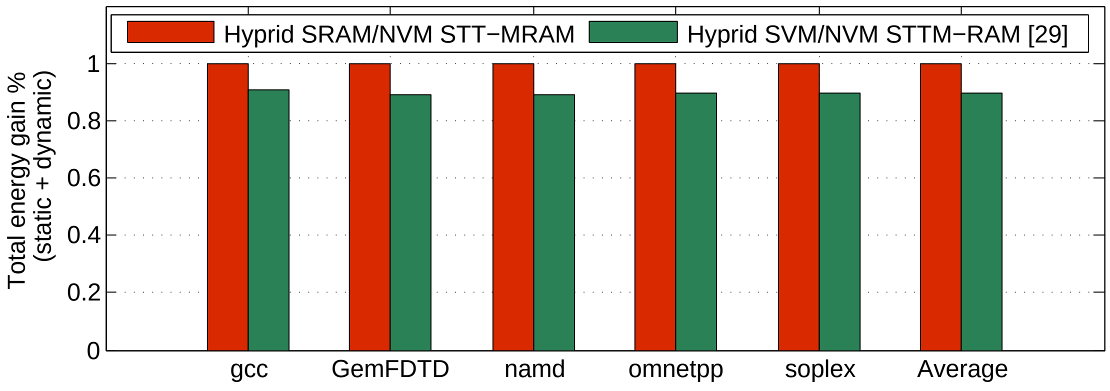


Figure 3.22: Comparison of total (static and dynamic) energy consumption gains at hybrid cache level normalized to hybrid SRAM/NVM STT-MRAM cache architecture [124]

3.4 Statict and Dynamic Data Placement Approaches Comparison

In the proposed static data placement approach, since write-intensive data is mapped to the relaxed STT-MRAM at compile-time, better performance and energy gains are achieved. These gains in the dynamic approaches can be reduced due to two reasons: 1) write-intensive data pattern classification accuracy, where the wrong classification leads to increase the write rate in the non-volatile STT-MRAM, and hence, performance and energy penalty, and 2) data migration costs between semi and non-volatile STT-MRAM blocks. Therefore, as the data behavior varies significantly over the time, the data pattern classification accuracy degrades and the data migrates frequently between cache regions. This in turn leads to high performance and energy penalties, which may erode the achieved gains of adopting hybrid cache architecture. Moreover, the control circuitry required for dynamic approaches, which continuously monitors each cached data, impairs read/write access latency and imposes energy, area and storage overheads. In contrast, our proposed static data placement scheme is firstly built based on the compiler and profiling information. Then it is applied at compile-time. Consequently, the aforementioned run-time costs related to the data migration and the control circuitry are eliminated with the proposed static data placement policy.

Figure 3.10 shows a comparison of energy gains achieved in the hybrid STT-MRAM data-cache of our static data placement policy against the dynamic data placement as proposed in [104]. In the dynamic data placement, the data swapping between hybrid STT-MRAM blocks depends on the access counter saturation. In [104], the energy efficiency of the hybrid STT-MRAM reaches, on average, up to 34% compared to the standard STT-MRAM. However, these gains will further decrease due to the energy consumption by the control circuitry, while the energy efficiency increases up to 44%, on average in our static data placement policy.

Note that for gzip and bzip2 benchmarks, the dynamic data placement is more energy efficient than our proposed policy. This is because the data in these benchmarks has a constant repetitive behavior, which can improve the run-time data placement and hence reduces the overheads related to the dynamic policy significantly, such as data migration costs.

3.5 Summary

Combine SVM with NVM STT-MRAM for a hybrid STT-MRAM architecture is proposed to be adopted as on-chip memories, which possesses energy-efficient, high storage capacity, better performance and high reliability. Static or dynamic data management policies are required to support the proposed hybrid architecture.

A static policy relies on the profile-based and compiler analysis in order to obtain data residency and access pattern. Our experimental results show that compared to the standard non-volatile STT-MRAM architecture in the same area configuration, the hybrid STT-MRAM supported with the proposed static data management approach reduces the overall static and dynamic energy of the cache by 8.1% and 44%, respectively. In addition, the system performance has been improved up to 8.1%.

For a dynamic data management policy, three different behavior prediction approaches are proposed with accuracy of 75%, on average. Our experimental results show that the proposed hybrid STT-MRAM architecture supported with a behavior prediction approach reduces the dynamic energy consumption by 38.2%, while system performance gain reaches up to to 11.2%, in comparison with the standard non-volatile STT-MRAM.

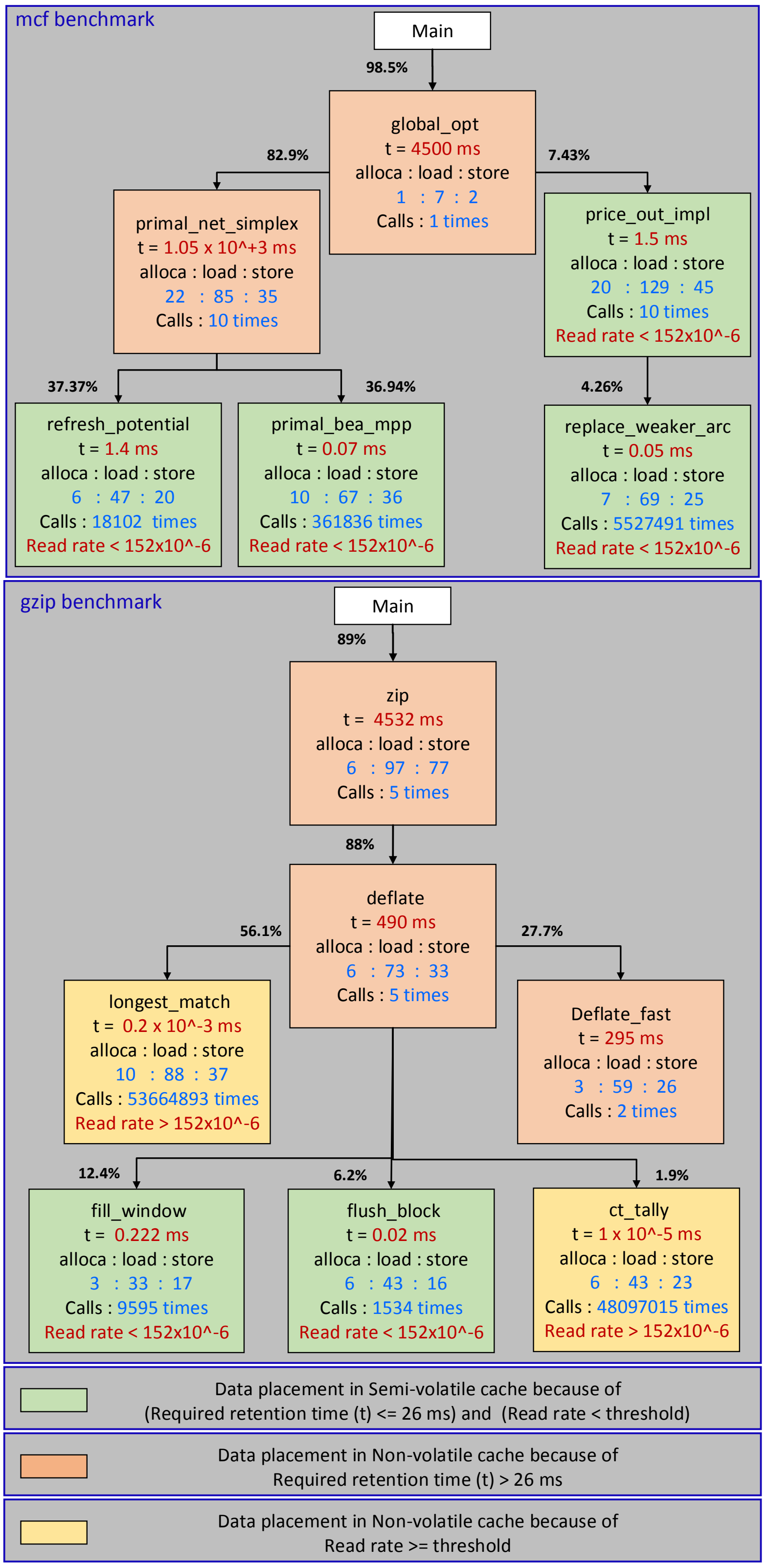


Figure 3.23: Data placement based on profiling-based and compiler analysis for 'mcf' and 'gzip' benchmarks

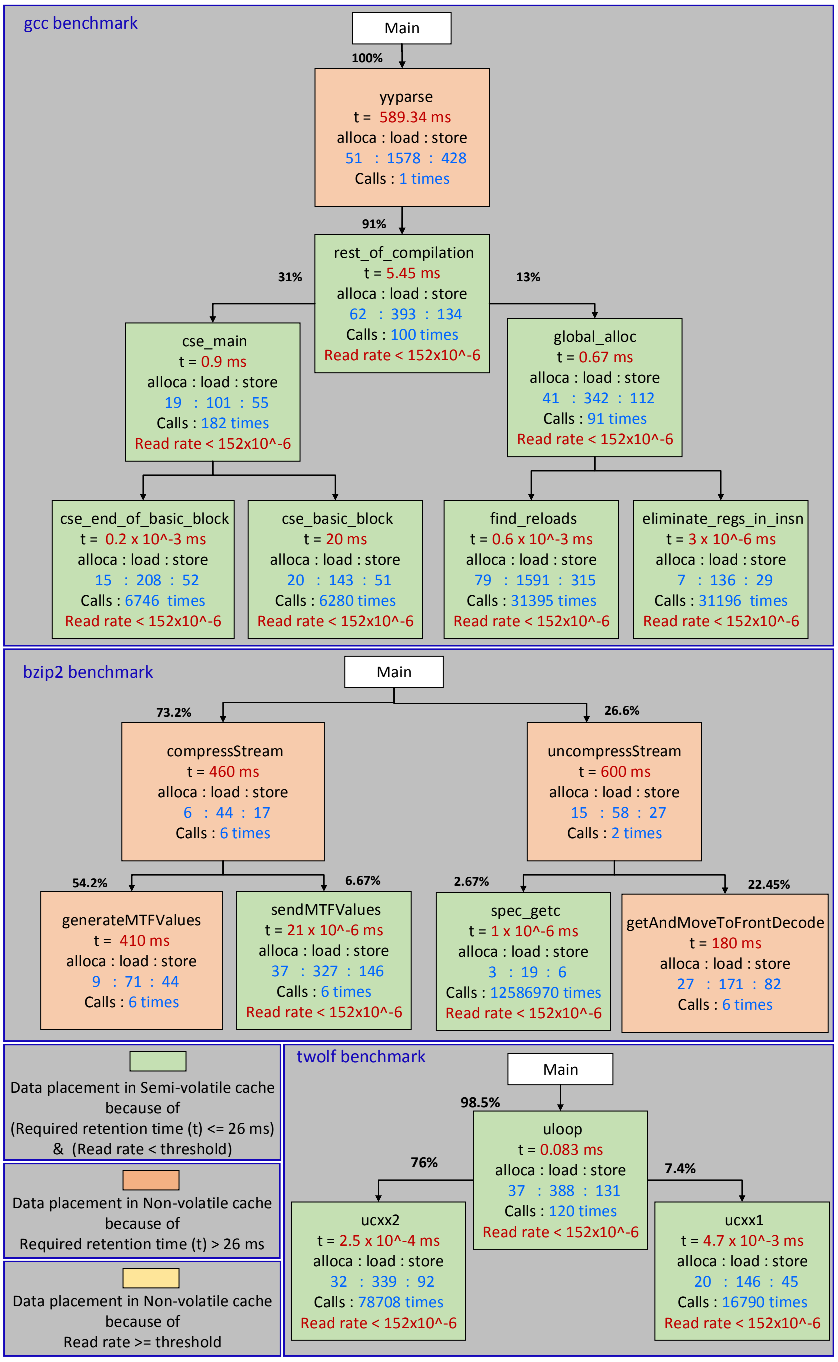


Figure 3.24: Data placement based on profiling-based and compiler analysis for ‘gcc’, ‘bzip2’ and ‘twolf’ benchmarks

Chapter 4

Exploiting STT-MRAM for Approximate Computing

In this chapter, we show the potential of the STT-MRAM technology in the scope of approximate computing. In general, approximate computing relaxes the bounds of accurate computing for the applications which are inherently error resilient and in return, it improves the efficiency of the system by other means such as performance and energy improvements [99, 100]. Many applications, for instance in audio or image processing domains, possess an intrinsic fault tolerance, and thus can deal with noisy (i.e. erroneous) data. For example, when the output of multimedia processing algorithms (images, videos, etc) is left to human perception, strict exactness may not be required and imprecise results may often be sufficient. Hence, with approximate computing, an increasing error rate can be tolerated to improve the write/read energy/latency of STT-MRAM with minimal cost. Moreover, even the overall system energy efficiency can further be improved due to shorter execution times thanks to the improved cache access latencies.

This chapter starts with a comprehensive study of a maximum reduction achieved in the STT-MRAM characteristics and requirements (i.e., read/write margin, read/write current, and non-volatile ability). Afterwards, we find a methodology to determine an optimal thermal stability factor adopted at device-level in order to find the right balance between acceptable output accuracy at application-level as well as energy and performance gains at system-level.

4.1 Relaxing STT-MRAM Parameters

Figure 4.1 presents a comprehensive view for possible relaxing approaches in STT-MRAM technology with respect to reliability constraints.

4.1.1 Relaxing Thermal Stability Factor

In this section, we deal with the induced error rates to trade-off the energy and performance improvements by lowering the Δ value. Indeed, lowered Δ reduces the write access latency and energy significantly at the cost of increasing the retention failure and

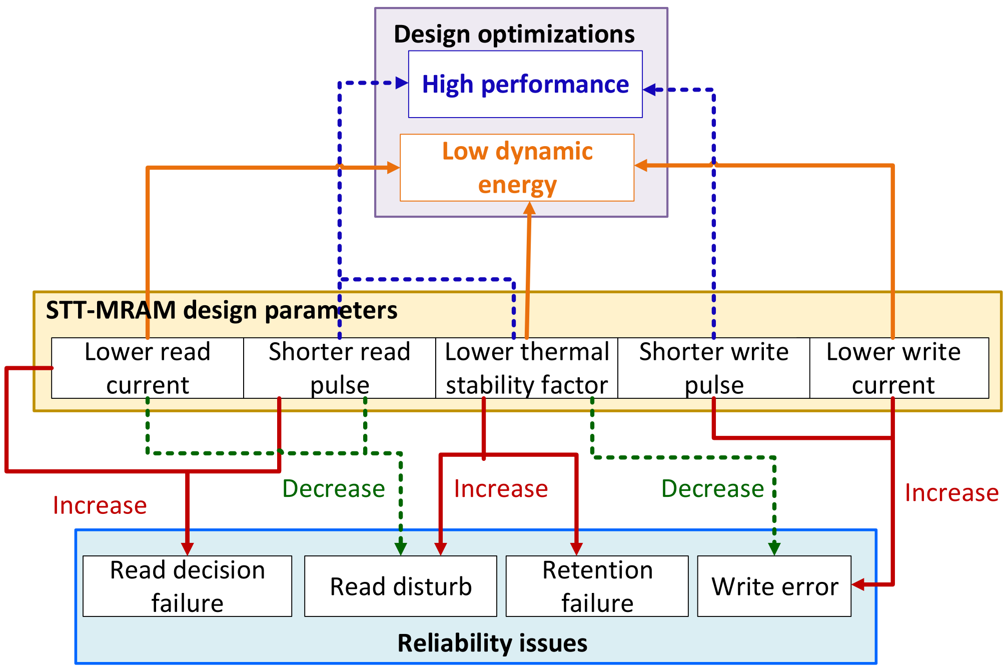


Figure 4.1: Relaxing STT-MRAM parameters and requirements diagram

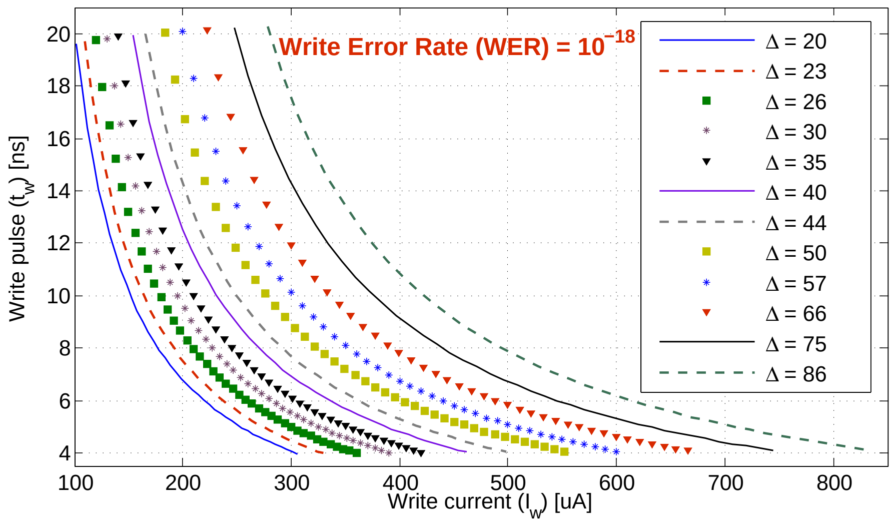
read disturb probabilities, as discussed in 2.3.4 and illustrated in Figure 2.6. Such failure rates can be exacerbated under temperature and process variations, since they affect the adopted Δ (Figure 2.5). Table 2.1 shows temperature and process variation combined effects on the reliability issue for lowering Δ from 60 down to 30. the Δ value of 60 and 30 refers to the ability of retaining the data, on average, for around 10 years and multi milliseconds, respectively. In our framework (see Section 4.4.1), the improvements of the write latency and energy per access for scaled Δ from 60 to less than 30, reach up to 25% and 70%, respectively. In contrast, the retention failure and read disturb probabilities increase exponentially and become non-acceptable for the Δ less than 40.

4.1.2 Relaxing Write Requirements

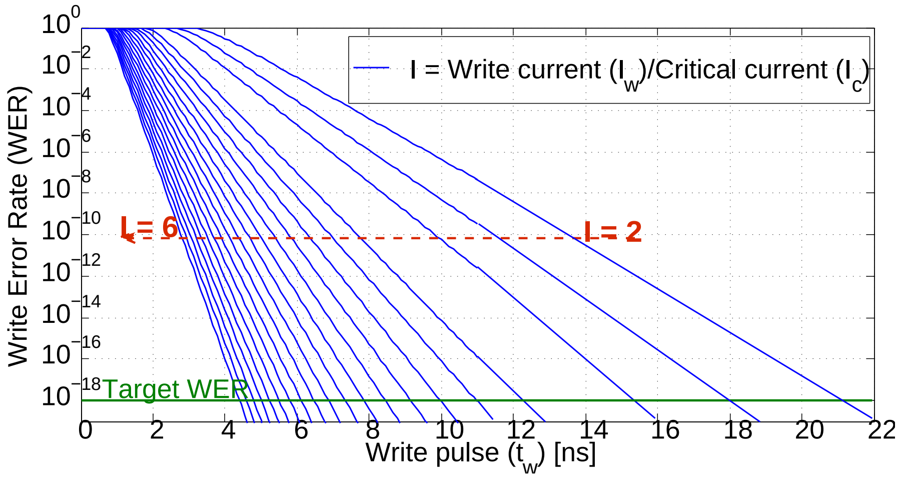
In STT-MRAM, a write operation is identified with four main parameters:

1. Thermal stability factor (Δ).
2. Write current (I_w).
3. Write pulse (t_w).
4. Target WER.

In general, writing in the STT-MRAM needs a high write current, which is much higher than the critical one (I_c). On the other hand, the magnetic orientation switching in an MTJ-cell is intrinsically stochastic due to the random thermal fluctuations, the switching time of bit-cell can significantly vary over the entire memory array. Such stochasticity can further be exacerbated due to temperature and process variation effects. For a reasonable WER (i.e., 10^{-18} [55]) and certain Δ value, there is a trade-off between the



(a) Write current versus write pulse scaling for different Δ values and WER=10^{-18}



(b) Write current and write pulse scaling versus WER

Figure 4.2: Relaxing write characteristics

write margin and the write current (i.e., write energy), as seen in Figure 4.2(a). An efficient write operation at design-time can be obtained with lower Δ value at a cost of higher retention and read disturb errors. However, at run-time, lower write current or sorter write pulse reduces the write costs along with higher WER.

Improving STT-MRAM write operation based on lowering the Δ has been discussed in the previous section (Section 4.1.1). Figure 4.2(b) illustrates the induced write efficiency by relaxing the write pulse or current at a cost of higher WER. Table 4.1 shows the actual WER for either relaxing the write current with a fixed write pulse or vice versa along with related energy and performance gains, where the actual WER is orders of magnitudes higher than the target one (i.e., $WER = 10^{-18}$).

4.1.3 Relaxing Read Requirements

The improvement in the read operation, which is on the critical path of the system performance, can be achieved at a cost of a higher probability of either read decision

Table 4.1: Effective WER with related gains for target WER of 10^{-18} with 1) different write currents (I_w) for a fixed write pulse (t_w) of 4.5 [ns] or with 2) different write pulses (t_w) for a fixed write current ratio ($I = I_w/I_c$) of 2

Write current ratio (I_w/I_c)	2	3	4	5	6
Write pulse (t_w) [ns]	4.5				
WER	0.12	5×10^{-7}	2×10^{-11}	5×10^{-16}	1×10^{-18}
Write energy gains	88.8%	75%	55.5%	30%	0%

Write pulse (t_w) [ns]	4.5	5.7	7.8	10.89	21
Write current ratio (I_w/I_c)	2				
WER	0.12	0.006	5×10^{-5}	5×10^{-8}	1×10^{-18}
Write performance/energy gains	78.5%	72.8%	62.8%	48%	0%

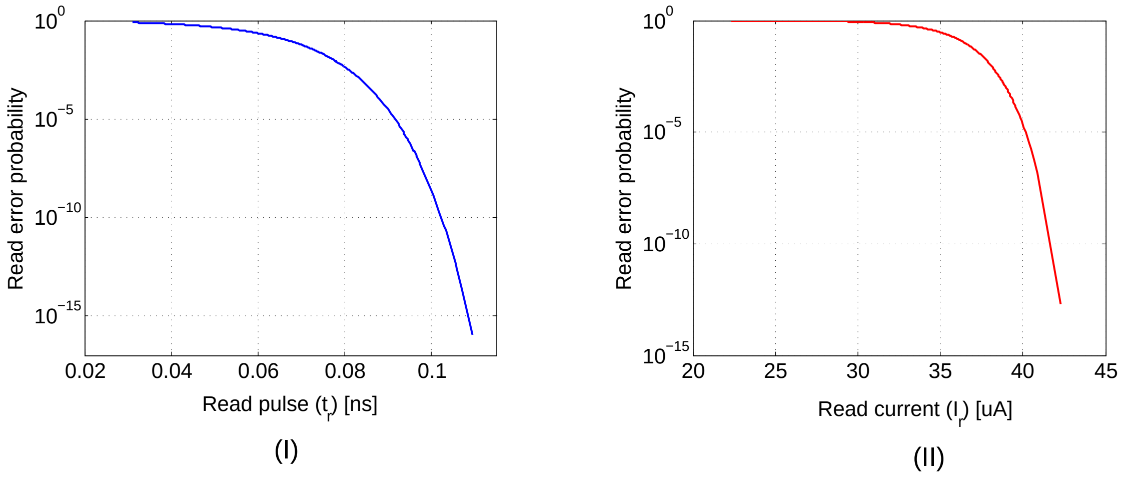
failure or read disturb.

In a read decision mechanism, the performance is improved with a shorter margin, while the energy is reduced by a lower sensing current. This can be achieved along with increasing the probability of an incorrect bit-cell content determination. Quantifying the read margin and the current versus the read decision probability trade-off is obtained by running Monte-Carlo simulation to generate 10,000 samples for read pulses and currents. This process is implemented at device-level under 5% process variation based on the framework described in Section 4.4.1. Figure 4.3(a) shows the read error probability for either relaxing the sensing current with a fixed margin or vice versa. As noticed, the produced error probability is orders of magnitudes higher than the target one. In contrast to previous mechanism and based on the read disturb Equation (2.11), adopting higher read current for shorter read pulse or increasing the read margin for lower read current results in a higher read disturb probability, as shown in Figure 4.3(b) and (c).

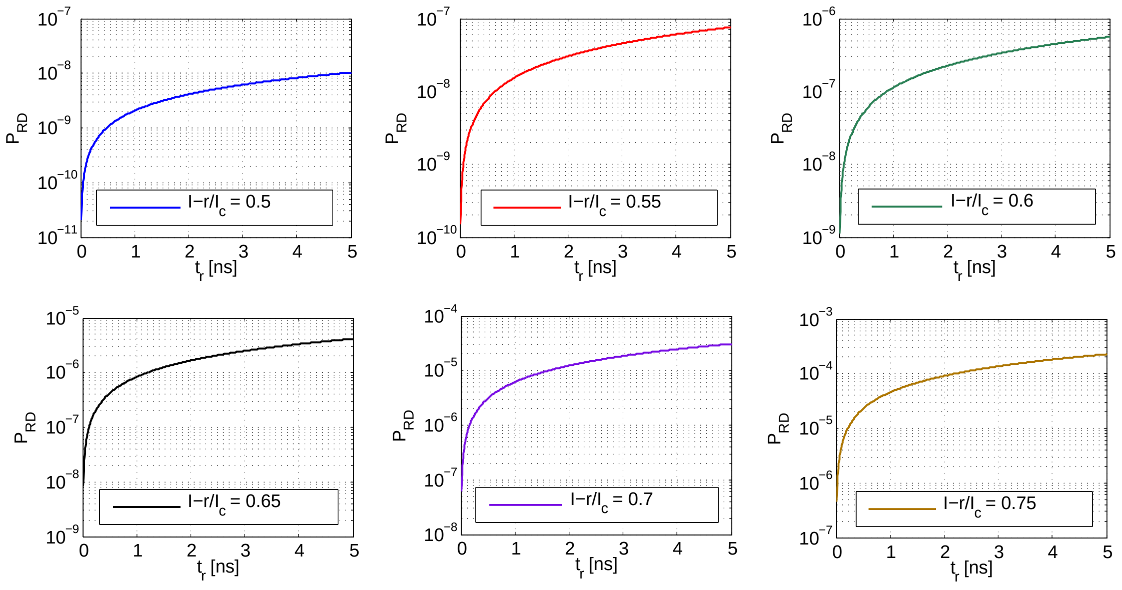
For energy-efficient read operation, lowering the read current under a fixed read decision failure probability is achieved by increasing the read margin from 1.5 [ns] (i.e., 3 clock cycles for 2GHz CPU) to 2 [ns] (i.e., 4 clock cycles). However, this leads to increase the read disturb probability by 1.34×. On the other hand, lowering the read current (e.g., 10%) within a fixed margin and read disturb probability incurs an orders of magnitude increase in the read decision failure probability.

4.2 Approximate Computing

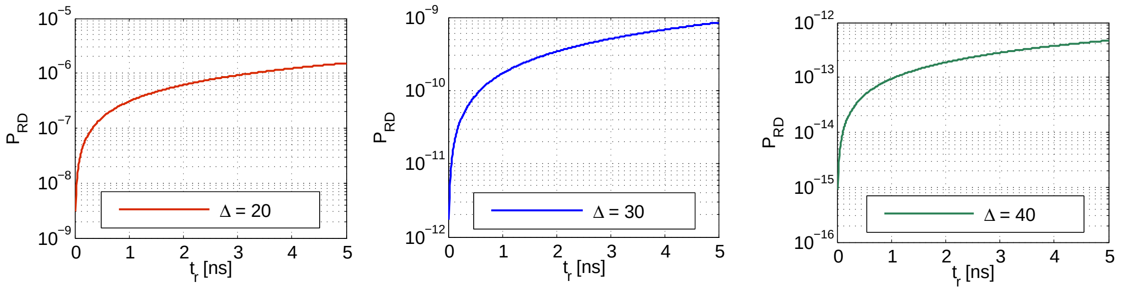
The *Approximate Computing* (AC) concept has recently emerged as a promising approach which relies on relaxing the bounds of precise computing to improve the performance and energy efficiency by orders of magnitude. This is done by leveraging the applications resilience to the errors and producing an acceptable output quality within the tolerable ranges instead of error-free output. Furthermore, many of the applications, which are data intensive and hence on-chip memory-exhausting, provide opportunities to exploit approximate computing very efficiently due to the huge existing gap between the level of accuracy required by the perceptual limitations of humans and that provided by the system, such as image, audio and video processing applications. Thus, for such applications, AC has the potential to significantly optimize the behavior of the on-chip



(a) Read decision technique I) Read pulse relaxing II) Read current relaxing



(b) Read disturb technique with relaxed read pulse



(c) Read disturb technique with relaxed read current

Figure 4.3: Relaxing read characteristics

memory in terms of performance and energy. In this chapter, we employ this approximate computing scheme in memories at architecture-level and perform the evaluation using approximate computing applications.

4.3 Approximate computing in STT-MRAM with Low Thermal Stability Factor (Δ)

This section presents the approximate computing methodology for STT-MRAM with relaxed retention time (lower Δ). Firstly, the overview is explained, followed by a detailed description of each abstraction component of our proposed methodology to trade-off accuracy versus performance and energy. In the end, we explain how some parts of data, which are critical and reliable, have to be protected in our framework.

4.3.1 Overview

Since on-chip memories (e.g., L1 cache) often do not require long data retention periods due to frequent accesses, the thermal stability factor of bit-cells in those memories can be reduced, which lowers the data retention time. As a result, the write latency will be reduced, as well as, the amount of current required to switch the bit-cell content will be smaller. Consequently, the overall write energy will become significantly better.

However, the drawback of this approach is an exponential increase in the retention failure and read disturb rates. Therefore, to analyze this trade-off, we have developed a cross-layer framework (as demonstrated in Figure 4.4), in which we carried forward the impact of Δ reduction to the memory architecture and application-level. We have calculated the impact of various Δ values on failure rates as well as write energy and latencies at the device level (similar to Figure 2.6). These values are projected at the architecture-level for per access energy and delay as well as failure rates. We have performed a fault injection model at memory architecture-level to accurately mimic the possibility of read and retention failures. At system-level, the application output is evaluated using the *Signal to Noise Ratio* (SNR) metric. This way we can find the optimal value of Δ for the best trade-off between acceptable quality of the output (correctness) as well as the performance and energy improvement. In the context of approximate computing, we allowed erroneous (i.e. inaccurate) data to be fetched by the processor. Hence, leveraging approximate computing can improve the writing energy and latency of STT-MRAM. Additionally, as the write latency of STT-MRAM is reduced, the overall system energy efficiency can be further improved due to shorter execution times.

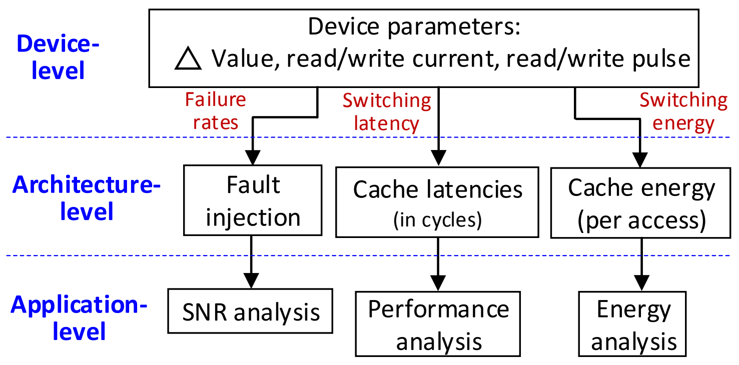


Figure 4.4: Illustration of cross layer approach for approximate computing in STT-MRAM.

Overall, in this chapter, we analyze the range of parameters from both STT-MRAM technology, system architecture and application requirements to achieve an acceptable level of quality while maximizing energy and performance gains. For that purpose, we employ image processing applications, in particular as a case study, we use JPEG format. This is a lossy compression method for digital images to store or transmit data in an efficient form without losing the ability to re-extract an acceptable version of the image. For this reason, the JPEG format provides some inherent level of fault tolerance, which means that it fits very well to the idea of approximate computing.

4.3.2 Adaptation of Thermal Stability Factor

As discussed in Section 2.3.4, the thermal stability factor has a strong influence on the reliability of STT-MRAM cell at both idle and access time because of the reduction in I_c. Consequently, various critical optimizations for STT-MRAM design can be performed, namely i) performance optimization which is obtained due to the reduction of the required time to switch the MTJ resistance state (i.e., write latency t_w), ii) as the write access power of STT-MRAM depends on I_c as: $P_w = {I_c}^2 \cdot R_{MTJ}$, where the resistance state of MTJ cell is fixed, a low I_c value leads to a reduction in power of the STT-MRAM cell, and iii) the energy consumption ($E_w = P_w \cdot t_w$) can also be dramatically reduced because of the small t_w and P_w values, since they are the outcomes of low Δ value. Figure 2.6 shows the effects of Δ value reduction on failure probabilities, write latency and write energy. In our framework, for the Δ reduction from 60 to 20, the improvements of the write latency and energy per access reach up to 25% and 70%, respectively. In contrast, the failure probability increases significantly and become non-acceptable for the Δ value below 32 (see Figure 2.6).

4.3.3 Fault Injection Approach

There are retention failures and read disturb failures associated with low Δ values for the STT-MRAM memory. Therefore, an accurate fault injection model has to be built based on the combined failure rates, i.e., *Failures In Time (FIT)*. The combined *FIT* rate has to be converted into access-based failure probability. The accesses in memory (e.g., L1 data cache) are performed by read and write requests. However, we perform the fault injection model at only read access, since it is more frequent compared to the write access. We compute the *Fault Injection Rate (FIR)* per read access based on *FIT* value of the adopted Δ and the *Read Rate (RR)* as follows: *FIR=FIT/RR*. Table 4.2 provides the combined *FIT* rate corresponding to Δ values from 20 to 60.

Table 4.2: The overall mean time to failure (MTTF) for various Δ values (for setup see Section 4.4.1)

Δ	MTTF [s]	Δ	MTTF [s]
20	$9.22X10^{-7}$	34	1.11
25	$1.37X10^{-4}$	35	3.01
31	$5.52X10^{-2}$	40	447.59
32	$1.50X10^{-1}$	45	$6.7X10^{+4}$
33	$4.08X10^{-1}$	60	$2.17X10^{+11}$

4.3.4 Tolerate Acceptable Error Rate Using Approximate Computing

To define the accepted value of the lowered Δ for cache memories, we have to estimate the acceptance quality of the extracted image by observing the influence of the Δ value on the error rate which will be translated into the read-access-based injected failures. The metric is based on the computed *Signal to Noise Ratio (SNR)* between the approximated image (faulty image) and golden image (faulty-free image). In other words, *SNR* can be served as a quality measure for the approximated image, which is calculated based on the variance of the signal (i.e., the golden image) (σ_S^2) and the variance of the noise (i.e., the faulty image) (σ_N^2) as Equation. (4.1), and is expressed in decibel (dB).

$$SNR = 20\ log_{10} \frac{\sqrt{\sigma_S^2}}{\sqrt{\sigma_N^2}} \tag{4.1}$$

The value of *SNR* impacts directly the performance and energy consumption of the proposed scheme. However, the minimum acceptable *SNR* value depends mainly on the application requirements. For instance, in image processing applications, the minimum required *SNR* value of the faulty image *SNR Threshold* (SNR_{th}) is set at the level where the human brain and eyes can differentiate between a faulty image and a golden one. This means that if the *SNR* value of the output is less than SNR_{th}, it is considered as an unacceptable output quality. See Figure 4.5, where SNR for an acceptable JPEG image quality has to be more than $SNR_{th} = 50$. However, for more performance and energy improvements, other smaller *SNR* values can be considered at the cost of less but still acceptable output quality. The corresponding pseudo-codes for generation of golden and faulty images are presented in Algorithm1 and Algorithm2.

Algorithm 2 Golden image generation

```
Input  Error-free output constraint {$\Delta_{max}$}
Output  Error-free image
1: Set the corresponding STT-MRAM based data cache configurations
2: Execute the simulation without fault injection mode
3:return Error-free image
```

4.3.5 Protection of Critical Data

A major challenge of using an approximate storage is that most applications which are highly amenable to the approximate computing paradigm, have a mixture of control data, so-called the critical data, which cannot tolerate any errors and the data that may be approximated or unreliable. Since the workloads in our case study are image processing applications (JPEG), which use a lossy compression technique to produce a similar coloured image with reduced size and an acceptable quality, any imprecision in the region of a code that stores the *meta-data* (the header of the image) totally corrupts the output image. This means header data is controlling the image data and should not be approximated. In contrast, the compressed image data (i.e., quantization) has tolerance to imprecision and any errors in it may only lead to some degree of quality loss in the output image. Therefore, the stored data has to be classified into reliable

Algorithm 3 Faulty image generation

```
Input Approximate computing constraints {$\Delta_{min}$, $\Delta_{max}$, $SNR_{th}$}
      and the maximum number of experiments (N)
Output Acceptable and non-acceptable percentage of image quality
1: For $\Delta$ from $\Delta_{min}$ to $\Delta_{max}$ do
2:    Set the corresponding STT-MRAM based data cache configurations
3:    Calculate the fault injection rate per read access (FIR)
4:    For $n$ from 1 to $N$ do
5:       Execute the simulation with fault injection mode
7:       Calculate $SNR$
8:       if $SNR > SNR_{th}$ then
9:          Increase the acceptable quality percentage
10:      else
11:         Increase the non-acceptable quality percentage
12:      end if
13:   end for
14: end for
15: return Acceptable and non-acceptable percentage of image quality
```

and unreliable data, which are header and quantization, respectively. There are various solutions to protect the critical part of the data.

The critical data size is very small compared to that of the non-critical data. To make critical data error resilient, one way is to design a heterogeneous memory array for the data cache which has high and low Δ values. In this design, the critical data has to be stored to the cells of high Δ to guarantee the error free operation. However, this requires changes to the fabrication parameters of the two arrays as well as complex cache controller to allocate data to different arrays. A less complicated way to protect the critical data is either to use multiple copies of the content of this data, such as dual or triple modular redundancy, or to use *Error Correction Code* (ECC), which protects the data by adding check bits. Since the size of critical data is significantly smaller than approximate-able data, the overhead due to extra bit-cells of either the repeated bits of dual or triple modular redundancy or the check bits of ECC approach is minimal.

4.4 SIMULATION RESULTS

In this section, we present the experimental analysis to show the benefits of using STT-MRAM in approximate computing by using image processing applications (JPEG) as a case study.

4.4.1 Simulation Setup

For our evaluations, we extended the gem5 simulator with a fault injection framework in order to support the retention time of each particular value of Δ adopted for an STT-MRAM based data cache. On the other hand, the instruction cache has to be faulty-free. This is why we assumed $\Delta = 60$ for the instruction cache to guarantee high retention time. This assumption is reasonable, as the number of write accesses to the instruction cache is considerably lower. Therefore, the performance and energy overheads of adopting higher Δ for such a cache would be negligible. Table 4.3 summarize the set-up for our experiments. The evaluations are performed by running 3 applications of image processing from MiBench applications 100 times in order to make our fault injection model non-deterministic. Each output for each experiment has a different level of quality loss according to the error rate of the used Δ. We use SNR as the metric system to evaluate the quality of the applications output.

4.4.2 SNR Evaluations

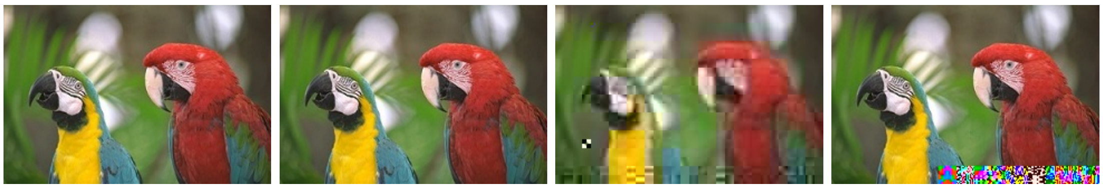
The SNR measurements reveal the degree at which an application could produce satisfactory and reasonable output due to the use of the relaxed thermal stability factor of STT-MRAM in data cache for approximate applications. This is to determine the achievable limits of the performance and energy improvements for STT-MRAM technology. Our analysis depends on extracting 100 different values of SNR for each Δ value up to 70 for the three image applications.

Figure 4.5 shows acceptable and non-acceptable faulty image quality along with the original one. After observing the images along with the related SNR, we can define the values of SNR_{th} which are 50, 50 and 60 for djpg, jpegtran and cjpeg, respectively. This means that the faulty image can be considered as acceptable, if its SNR value is greater than SNR_{th} (as explained in Algorithm 2).

According to the SNR measurements, we count the acceptable images from the entire 100 faulty images related to each Δ value for all the image applications (see Figure 4.6). This image shows that for $\Delta \leq 25$, the percentage of acceptable quality is zero, for the three image applications, whereas, for $\Delta \geq 40$, all the images are acceptable. For Δ values equal to 32, 33 and 34, our results show that the percentage of the acceptable quality is more than 95%. Therefore, values of Δ can be considered for approximate computing, in which the output with quality loss is less than 5%. Please note that in our framework, the retention failure rate dominates over the read disturb rate for the SNR measurements (see Figure 2.6 (b)). For the applications with faster read requirements, higher read currents are necessary, and consequently, the read disturb rate will be higher. In such cases, the SNR_{th} evaluation criteria can be changed, and accordingly, the percentage of the acceptable quality can be altered.

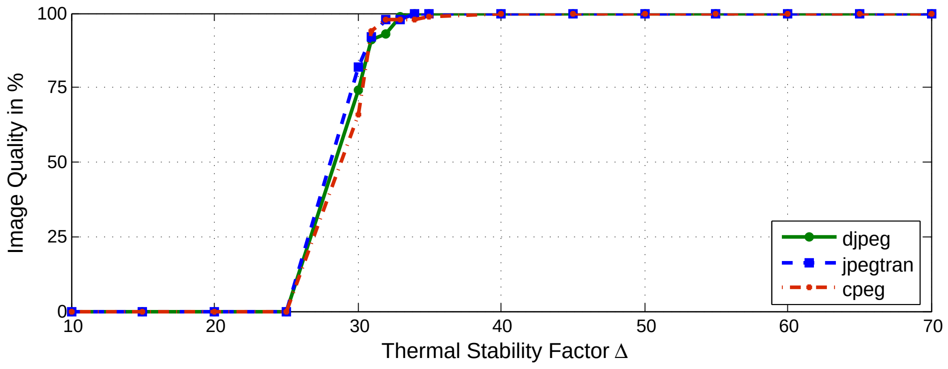
Table 4.3: Simulation setup in gem5

gem5 confiquration	ISA x86
Processor	Single-core, 0.5/1/2 GHz, Out-of-order, 4-issue
L1 data-cache	64 KB, 2-way set associative, 64B line size STT-MRAM, different write latencies and Δ values
L1 instruction-cache	32 KB, 2-way set associative, 64B line size STT-MRAM, $\Delta = 60$
MiBench Applications [127]	djpeg, cjpeg, jpegtran

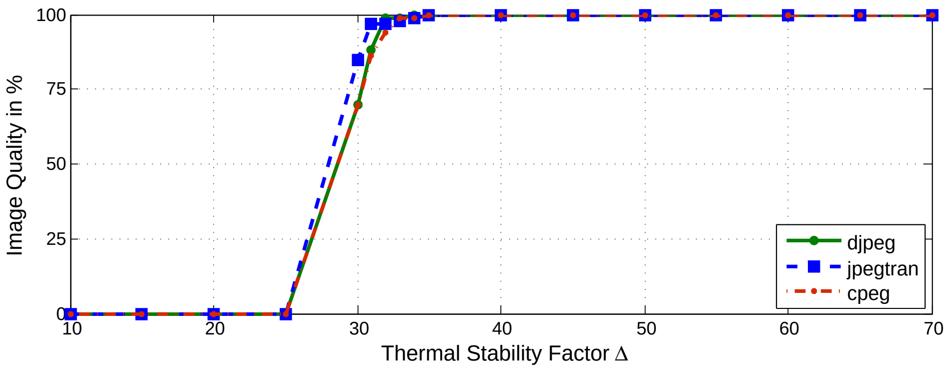


(a) Golden image (b) Acceptable image qualtiy $\Delta = 25$ & $SNR = 50$ (c) Non-acceptable image quality $\Delta = 26$ & $SNR = 23$ (d) Non-acceptable image quality $\Delta = 27$ & $SNR = 19$

Figure 4.5: Images of various Δ and SNR values for STT-MRAM data cache for "jpegtran" workload



(a) 1 GHz clock frequency

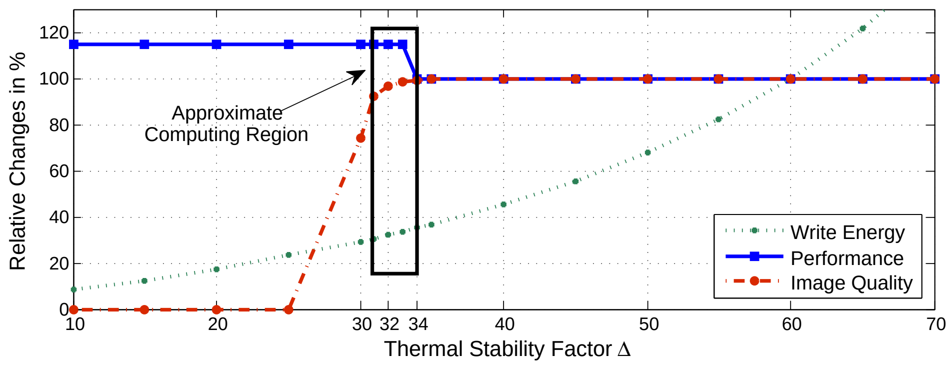


(b) 2 GHz clock frequency

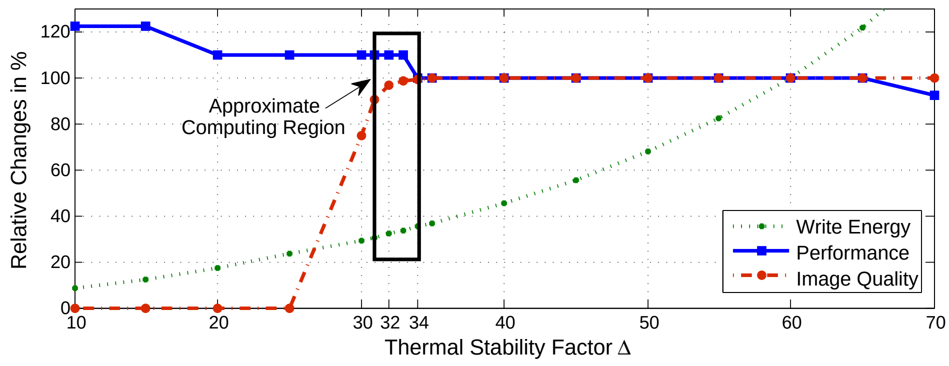
Figure 4.6: Acceptable image quality for various Δ values

4.4.3 Performance and Energy Analysis of Optimal Δ for Approximate Computing

The performance influence of the lower Δ value for STT-MRAM data cache is closely associated with the processor frequency. On one hand, for a given frequency, the write latency is reduced by 0.5 ns when Δ is lowered to 32, as illustrated in Figure 2.6(a). On the other hand, the performance gain further increases by a low clock frequency of the processor as well. For instance it reaches to 25% and 10% at clock frequency 1 GHz and 2 GHz, respectively (see Figure 4.7). Furthermore, according to our experiments, 66% of the total energy of STT-MRAM for the L1 cache is consumed for write operations in data cache. Therefore, any improvement in the energy consumption of STT-MRAM



(a) 1 GHz clock frequency



(b) 2 GHz clock frequency

Figure 4.7: Relation between image quality, performance (runtime) and write energy for an STT-MRAM based data cache

cell in the data cache leads to the overall system energy efficiency. Results show that we are able to improve the energy consumption per access up to 70% with a Δ value 32.

Based on the performance and energy improvements of the low Δ value along with the output quality, we can define the optimal Δ value for a particular system setup. Figure 4.7 illustrates this trilateral relationship (i.e., energy, performance, quality). According to this figure, the optimal configuration for approximate computing applications can be obtained by relaxing the thermal stability factor from the usually applied 60 down to 32, where the overall system energy reduction reaches up to around 46% and the performance gains are 25% and 10%, at clock frequency 1 GHz and 2 GHz, respectively.

4.5 Summary

In this chapter, we have developed a cross-layer framework, from technology parameters to architecture and application levels to evaluate the applicability of STT-MRAM technology for approximate computing. We have also provided a methodology to determine the optimal thermal stability factor in order to find the right balance between acceptable accuracy as well as energy and performance gains. Our results show that the energy reduction of 46% along with a performance gain of up to 25% can be achieved at system level with a reasonable output quality.

Chapter 5

STT-MRAM Using Non-uniform and Adaptive Error Correcting Scheme

In this chapter, we propose a new opportunistic write scheme for STT-MRAM to improve its performance in order to be used for fast on-chip memories. The proposed technique terminates the write operation opportunistically by allowing unfinished switching of the bits with large write latency. Then, we enable a fast, yet robust ECC, namely *Lazy Error Correction Code* (Lazy-ECC). This is achieved by decoupling error detection and correction steps, and performing speculative computation on the unchecked data. Furthermore, we exploit the low *thermal stability factor* (Δ) for faster and lower energy MTJ switching. As a result, the associated error rate of the proposed approach is relatively high because of: i) the unfinished write, and ii) the higher retention failure and read disturb rates due to the smaller Δ. To address these challenges, we present a comprehensive cross-layer study going from device level up to architecture level on the combined effects of the process variation, the operating temperature and the stochastic write operation. Based on this cross-layer analysis, we propose a runtime adaptive approach to manage the combined effect of these phenomena on the overall write latency and reliability at the architecture-level.

In the proposed Lazy-ECC scheme, 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 the data integrity. Consequently, our proposed technique enables fast, reliable and energy-efficient STT-MRAM design for the fast memories. Furthermore, an adaptive and non-uniform error correcting scheme has been proposed by considering the combination of the process variation and the operating temperature effects on the read disturb, retention failure and write error rate.

5.0.1 Error Correcting Codes (ECCs)

In the information theory, ECC(n, k, e) specifies that for the length of the input data which is equal to k bits, the length of the encoded data should be n bits, and the coding technique is able to correct e errors. A memory array with ECC capability has an encoder in its input and a decoder on its output. The data is encoded using the encoder unit and then is stored in the memory. Upon a read request, first the data is read from the memory and after performing the decoding process, the data is sent to the output.

Among all various multiple bit correction schemes, BCH code is more appropriate for STT-MRAM as it can correct multiple random errors [128]. Error detection in BCH is a fast process which is done by generating and evaluating syndromes. In contrast, error correction is quite slow because the positions of all errors have to be detected using an iterative algorithm [98].

5.0.2 Reducing STT-MRAM Write Margin by Exploiting ECCs in Opportunistic Write Policy

Recent studies [94, 128] exploited ECC to improve the STT-MRAM performance through reducing the write margin to a large extent by correcting errors in the tail of the stochastic write distribution. 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, $WP_{word}(t_w)$, which the write probability of an n-bit word in t_w, can be computed as:

$$WP_{word}(t_w) = WP_{bit}(t_w)^n = (1 - WER_{bit}(t_w))^n \tag{5.1}$$

Where $WP_{bit}(t_w)$ is the write probability for a single bit. In order to determine the write probability of n-bit word protected with ECC(n,k,e), a binomial distribution is used as follows:

$$WP_{word}(t_w) = \sum_{i=0}^{e} \binom{n}{i} \cdot (1 - WP_{bit}(t_w))^i \cdot WP_{bit}(t_w)^{n-i} \tag{5.2}$$

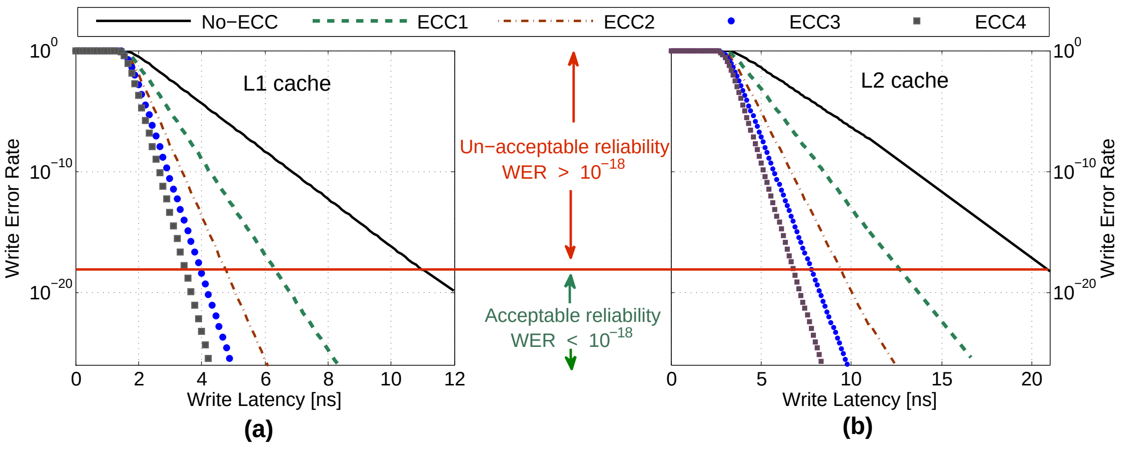
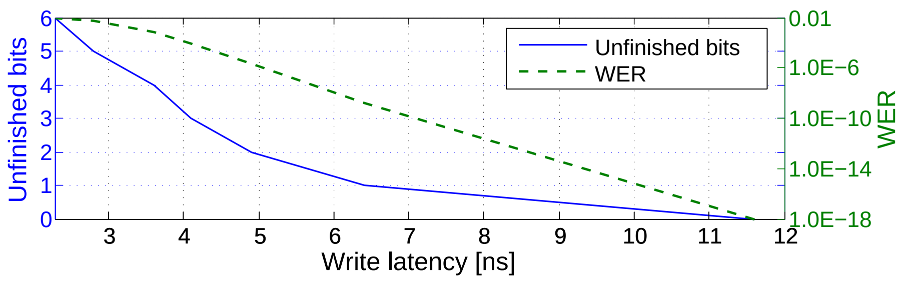
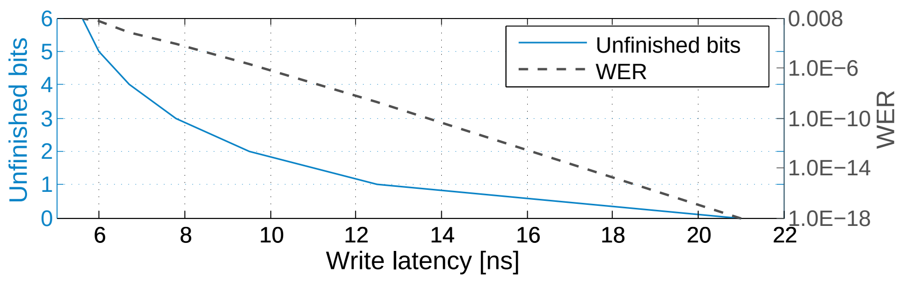


Figure 5.1: Write error rate vs. write latency for 64-bit data with different ECCs excluding ECC encoding; (a) For L1 cache with $\Delta = 30$ and $I_w/I_c = 3$; (b) For L2 cache with $\Delta = 40$ and $I_w/I_c = 2$; See setup in Section 5.3.1]



(a) Unfinished bits with corresponding probability vs. write latency for L1 cache ($\Delta = 30$, $I_w/I_c = 3$)



(b) Unfinished bits with corresponding probability vs. write latency for L2 cache ($\Delta = 40$, $I_w/I_c = 2$)

Figure 5.2: Write latency reduction with opportunistic write policy for 64-bit data for L1 and L2 caches, setup in Section 5.3.1

Please note that, Equation 5.2 can be simplified to Equation 5.1, if $e = 0$. This means that No-ECC is adopted to estimate the write probability with required margin.

Figure 5.1 (a) and (b) illustrate the WER value for various write latencies in a 64-bit data-word with different BCH schemes that are able to correct 1 up to 4 errors (denoted as ECC1 to ECC4). As shown, for achieving an acceptable WER value (e.g. 10^{-18} [66]), the write latency is effectively reduced from 11.6 ns to 3.6 ns for L1 cache (see Figure 5.1 (a)) and from 21.0 ns to 6.7 ns for L2 cache (see Figure 5.1 (b)) by increasing the ECC robustness up to ECC4. Table 5.1 represents the storage overhead along with the minimum delay for the encoding and the decoding circuitry for word size of 64-bits synthesized using TSMC 65nm standard cell library.

The opportunistic write policy is an energy efficient and high performance policy in which the write operation is terminated when most of the bit switchings are completed. It is used to overturn the write margins in STT-MRAM memory from over-estimations to under-estimations. Therefore, it is supported with a new ECC mechanism to cover the gap between the under- and the over-estimations. The opportunistic write is implemented at word-level (64 bits). In this regard, the opportunistic write operation will be terminated faster than the normal one with a particular probability of having unfinished bits. In case there are some unfinished bits, the proposed Lazy-ECC scheme has to correct and rewrite the unfinished bits. Figure 5.2 depicts the opportunistic write margin distribution at word-level for L1 and L2 caches. As L1 cache filters most accesses, the write access latency of L1 cache is on the critical path of the system performance. Therefore, the adopted thermal stability factor is low (Δ=30) and the write current is high to guarantee fast write operations. However, in case of L2 cache, a higher thermal stability factor (Δ=40) is considered to guarantee higher retention time, as the data residency time in L2 is higher than in L1 cache. As shown in this figure, to have an acceptable write error rate (10^{-18}), the write latency has to be almost 12 ns and 21 ns for L1 and L2 caches, respectively. However, these margins can significantly be reduced at the expense of higher write error rate, which can be translated into the expected number of erroneous bits in the word, hence the required error correction capability. For instance, the write latency can be reduced to below 4 ns and 7.8 ns for L1 and L2 caches, respectively, translating to 3× reduction, by expecting up to four bit errors per word.

5.1 Lazy Error Correcting Code

In this section, it is shown that the conventional ECC-based approach has a large decoding latency. This impairs the read latency and makes STT-MRAM quite inefficient for fast memories such as L1 and L2 caches. In order to address this shortcoming, we propose a new technique to combine opportunistic write and perform speculative computation while data is being checked for possible errors.

5.1.1 Motivation

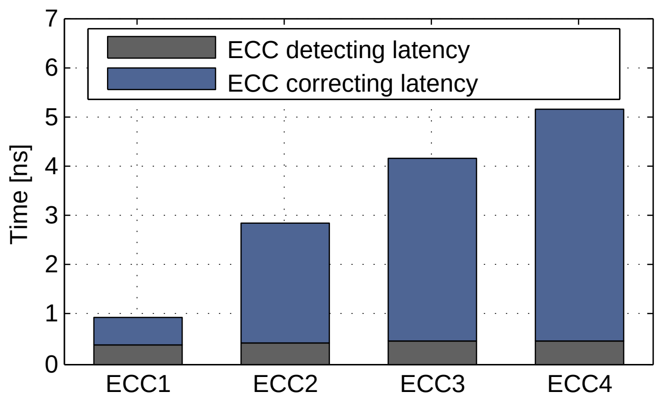
As shown in Section 5.0.2, the excessive write margin of STT-MRAM in fast memories such as L1 and L2 caches could be reduced to a large extent by exploiting small value of Δ and robust ECCs. However, robust ECC schemes have excessively large decoding latencies which negatively affect the read latency, and hence, the overall performance. In order to evaluate such claim, we synthesized the encoder/decoder circuitry of different ECCs for minimum possible latency. Table 5.1 reports the breakdown of write and read latencies for ECC1 to ECC4 for L1 and L2 STT-MRAM caches. For a single error correction, we reported the data for *Single Error Correction Double Error Detection*(SECDED) scheme which is commonly implemented in memories, while for 2-4 error correction, the BCH scheme is exploited as recommended by [94, 128]. As shown, although the write latency is reduced by exploiting more robust ECCs, the decoding latency grows significantly. Hence, the reduction in write latency comes at the cost of increasing the read latency to the extent that the overall read latency surpasses the write latency. This means that while a more robust ECC can bring the write latency within the required performance threshold, the read latency increases and falls beyond the desired performance limit.

In ECC(n, k, e), the syndrome vector required for error detection has a length of $n - k$ bits. Each bit of the syndrome vector can be obtained by a separate XOR tree with maximum a depth of $log(n)$ [98]. This means that the error detection circuitry has a very simple structure and also very small latency. However, the error correction circuitry is quite complicated. Table 5.2 reports the error detection and correction latencies for different ECCs applied on 64-bits data and synthesized with TSMC 65 nm standard cell library. As shown, the detection time is always smaller than the correction latency and the gap between these two significantly increases as ECC becomes more robust (11× for ECC-4). Therefore, separating error detection and error correction phases can provide some benefits, as will be discussed next.

Table 5.1: Read and write latencies of 64-bit STT-MRAM L1 and L2 caches with 16 KB and 512 KB capacity [Experimental setup in Section 5.3.1]

			No-ECC	ECC1 SECDED	ECC2 BCH	ECC3 BCH	ECC4 BCH
Storage overheads			0%	11%	18.9%	25.5%	31%
L1 cache	Write [ns]	ECC Encoding	—	0.400	0.525	0.530	0.545
		Memory Write	11.610	6.456	4.909	4.100	3.628
		Overall	11.610	6.856	5.434	4.630	4.173
	Read [ns]	ECC Decoding	—	0.580	2.459	3.698	4.714
		Memory Read	0.898	0.899	0.899	0.899	0.900
		Overall	0.898	1.479	3.358	4.597	5.614
L2 cache	Write [ns]	ECC Encoding	—	0.400	0.525	0.530	0.545
		Memory Write	21.00	12.50	9.50	7.80	6.70
		Overall	21.00	12.90	10.025	8.33	7.245
	Read [ns]	ECC Decoding	—	0.580	2.459	3.698	4.714
		Memory Read	1.120	1.121	1.121	1.121	1.122
		Overall	1.120	1.701	3.580	4.819	5.836

Table 5.2: Error detection and correction latencies [ns]



	Only Detection [ns]	Detection and Correction [ns]
ECC1	0.372	0.580
ECC2	0.403	2.459
ECC3	0.440	3.698
ECC4	0.442	4.714

5.1.2 Proposed Lazy-ECC Approach

The proposed Lazy-ECC scheme is independent of the processor architecture. Our objective is to eliminate the ECC penalty from the cache read path, which is on the critical path of the system performance. This concept can be implemented for both in-order and out-of-order, simple-scalar and super-scalar processor architectures.

For reverting all speculative computations based on erroneous data, only detecting the existence of an error is enough and the correction can be performed later. Therefore, our proposed approach decouples error detection from error correction to accomplish quick error checking. This reduces the costs associated with the reverting due to the smaller error sphere and proper error confinement.

The key to effectively address the long decoding latency of the conventional ECC-based approach is to eliminate the decoding latency of the robust ECC 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 our proposed approach, non-checked data is simultaneously propagated to the decoder and to other parts of the system (*s*ink). This means that data validation is not completed prior to data access and the propagated data might be erroneous. For this reason, this new ECC approach is referred to as *Lazy-ECC* approach.

In this approach, a fully combinational one stage decoding circuitry has to be avoided. 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 addition, it is crucial to perform the error checking at the same pace that the data is accessed from the memory. To boost the performance of the decoder, the detection and correction stages could also be further divided into smaller stages [129].

In Lazy-ECC, the erroneous data has to be detected before reaching the critical part 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 reaching 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 on the erroneous data are reverted and the computations are repeated based on the corrected data.

5.1.3 Instruction Cache Implementation

As a case study, the implementation of the Lazy-ECC approach for an instruction cache of a processor is presented. The connection between different units are shown in an illustrative block diagram in Figure 5.3. As shown in Table 5.2, the error detection latency is very small, which is in the worst case less than 500 ps and hence, in this example we assume that it can be done within one clock cycle in a 2 GHz processor. Once the processor reads an instruction from the instruction cache, the instruction is propagated to both the ECC decoder and the fetch stage of the processor. While the instruction is being processed in the fetch stage, the decoder circuitry checks for possible errors.

In case the data read from the memory is correct, execution continues without any interruption from the decoder circuitry. On the contrary, in case an error is detected using the decoder circuitry, it activates the error signal to revert the erroneous instruction from the pipeline before it propagates to the next stage. In addition, NOP instructions are inserted in the pipeline until the error is corrected. For error correction, the decoder corrects data and then the corrected data is written back to the memory. Once the correcting process is terminated, the processor continues its operation by re-fetching the corrected data from the instruction cache. In other words, in the error-free case, the read latency of Lazy-ECC is equal to the case with no-ECC, while providing the same level of error correction capabilities as robust ECCs.

For the other memory arrays, Lazy-ECC can be implemented in a similar manner. For instance, for data cache, the speculative computation can be done in the load/store unit while error detection is performed in parallel.

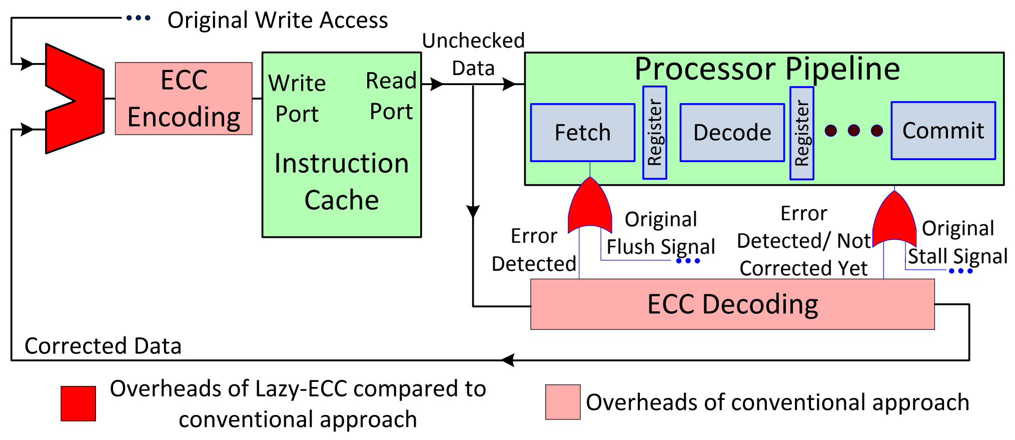


Figure 5.3: Block diagram of Lazy-ECC for instruction cache

Table 5.3: Effective WER of different ECCs for a target WER of 10^{-18}

	ECC1	ECC2	ECC3	ECC4
WER	1.7×10^{-9}	2.0×10^{-6}	9.0×10^{-5}	6.6×10^{-4}

5.1.4 Performance Gain Analysis

Our proposed approach improves the performance for cases with no errors while it increases the latency for cases with error. Therefore, the fraction of accesses with error is a key parameter in evaluating the performance improvement of our proposed approach.

As mentioned earlier, the write latency is typically determined according to a reasonable WER (e.g. 10^{-18} [66]). Therefore, it is expected that the acceptable probability of uncorrected errors of all conventional ECC-based techniques would be the same (i.e., 10^{-18}). However, the actual effective WER seen at the L1 and L2 cache outputs (i.e., before correction) is much higher (i.e., WER $> 10^{-18}$), and it reaches to that target level after error correction. Table 5.3 shows the actual WER at the cache output for various ECCs. As it can be seen, this is orders of magnitude higher than the target WER.

In the Lazy-ECC approach, all errors at the memory output are propagated to the system (processor) because the errors are not corrected during the read access. The error recovery rate is equal to the actual effective WER (Table 5.3). In case the recovery is required, the performance penalty by our approach would be small because the cycles used for speculative computation are also used for error detection and correction. Until this point our approach is similar to the conventional one. Afterwards, we have to write the data to the memory and read it again. However, this small penalty is negligible compared to the gain that we have for the majority of error-free accesses.

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 read, write operations 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 (T_{Lazy}) in our approach can be calculated as follows:

$$T_{Lazy} = \mathit{TER} \times (t_r + t_D + t_E + t_w + t_r) + (1 - \mathit{TER}) \times t_r. \tag{5.3}$$

Since the values of write error, retention failure and read disturb probabilities are small, the expected average read latency of our approach can be estimated by:

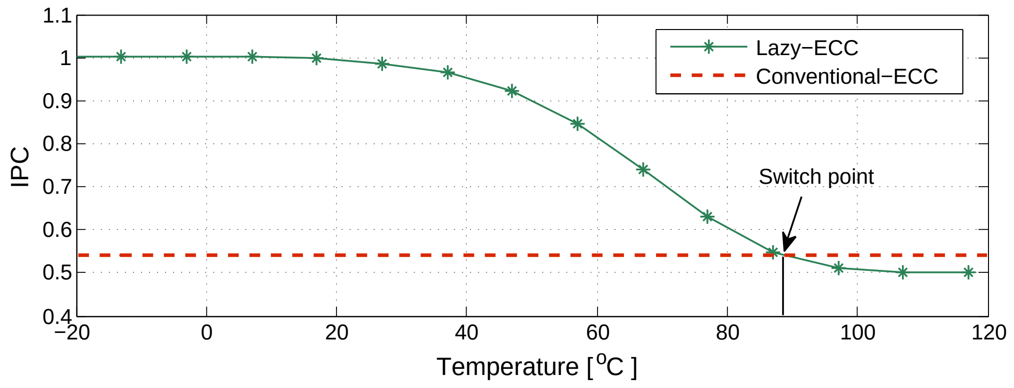
$$\lim_{\mathit{TER} \to 0} T_{Lazy} = t_r \tag{5.4}$$

5.2 Non-uniform and Adaptive ECC Approach

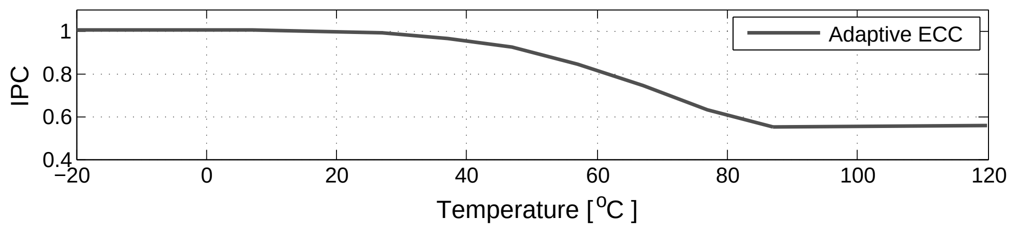
5.2.1 Adaptive ECC Approach to Address Temperature-induced Failures

As shown in Section 2.3.4, retention failure and read disturb probabilities increase exponentially under elevated temperatures, while WER slightly decreases. This means that the overall performance of the Lazy-ECC scheme is highly affected by higher operating temperature, as the penalty of Lazy-ECC scheme depends mainly on the TER value according to the Equation (5.3). Whereas, the cache performance with the conventional-ECC scheme is totally independent from the TER value and hence from the operating temperature. Therefore, we propose an adaptive and hybrid ECC scheme for fast and reliable STT-MRAM 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 for switching the adopted ECC approach. The optimal case can be achieved by adopting Lazy-ECC when TER is low enough due to low operating temperature. Afterwards, the performance degrades at higher temperatures due to TER increase. However, the cache performance with Lazy-ECC is still better than with conventional-ECC, since the TER value is smaller than TER_{th}. If the operating temperature goes beyond the threshold value 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 decrease, while the performance of conventional-ECC scheme is fixed under different operating temperatures. Figure 5.4(a) illustrates the performance behaviors under the temperature variation extracted by applying Lazy- and conventional-ECC on L1 and L2 caches. It is obvious that a pure ECC scheme (either Lazy-ECC or conventional-ECC) can not provide an optimal performance, hence, a hybrid scheme is required. In other words, for a certain adopted Δ value (i.g., 30), a Lazy-ECC scheme provides better performance compared to the conventional one even with higher temperature. However, as the temperature reaches a particular value (e.g., $88°C$ for Lazy-ECC, as shown in Figure 5.4(a)), we have to switch from the Lazy-ECC to the conventional-ECC to gain more performance, as illustrated in Figure 5.4(b)). Otherwise, the overall performance of the system is significantly impacted. The value of such threshold temperature is a function of the Δ value and the adopted write margin.



(a) IPC of Lazy-ECC versus conventional-ECC



(b) IPC of Adaptive ECC Approach

Figure 5.4: Instruction Per Cycle (IPC) of Lazy-ECC, conventional-ECC and Adaptive ECC Approach under different operation temperature values for L2 cache ($\Delta = 40$, $I_w/I_c = 2$) and L1 cache ($\Delta = 30$, $I_w/I_c = 3$), setup in Section 5.3.1

5.2.2 Non-uniform ECC Approach to Efficiently Address Process Variation Effects

As discussed in Section 2.3.3 and 2.3.4 process variation causes a considerable variance in the Δ value for the MTJ-cells of STT-MRAM memory, which significantly affects the write current (I_w) and the critical current (I_c) of MTJ-cell. This induces wide variations in the retention time, read disturb and write error rate (i.e., TER). Hence, their minimum and maximum values over the entire STT-MRAM memory array can differ by several orders of magnitude, as seen in Figure 2.6.

One method to address the process variation effects is to fix the TER_{th} of the proposed adaptive ECC approach based on the cell with minimum Δ in a memory array, i.e., the worst case process condition. However, this leads to an inefficient implementation of the adaptive ECC approach, because of the early switching to the conventional-ECC. As illustrated in Figure 5.5, by considering the process variation effect, the operating temperate of $27°C$ will be adopted for switching. Whereas, fixing the TER_{th} based on the normal case (i.e., with no process variation consideration) increases the operating temperature of the switching up to $88°C$. However, the normal case could lead to accumulation of errors beyond the error correction capability and hence results in failures of the memory system.

As explained in [130], the process variation in STT-MRAM shows strong spatial correlations, which means that the variation among neighboring cells are much smaller compared to cells that are far apart from each other. Based on this observation, the cache lines can be grouped according to their thermal stability factor (Δ) value. Each group is defined as a collection of cache lines which exhibit a very close Δ value that are

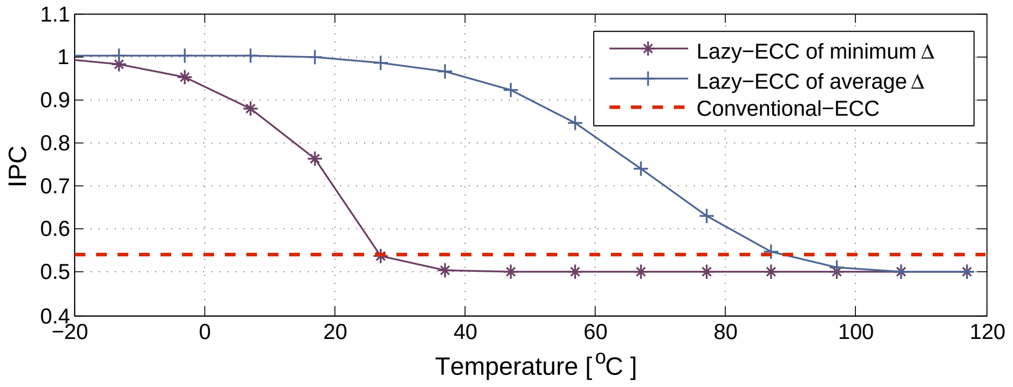


Figure 5.5: IPC of Lazy-ECC versus conventional-ECC with and without process variation consideration (i.e., minimum Δ and average Δ) for L2 cache ($\Delta = 40$, $I_w/I_c = 2$) and L1 cache ($\Delta = 30$, $I_w/I_c = 3$), setup in Section 5.3.1

easily distinguishable from those of the rest of the lines. Consequently, each group has a particular *Group Error Rate* instead of TER, and hence a particular threshold value (GER_{th}), which defines the operating temperature for switching between Lazy-ECC and conventional-ECC scheme for that group. As a result, a non-uniform adaptive ECC scheme is proposed for an efficient and reliable STT-MRAM cache design. We found that with dividing the cache into 2,4 and 8 groups at cache line granularity, the group size of minimum Δ can be reduced by 38%, 75% and 84%, respectively.

At design time, cache lines will be clustered into N groups based on the minimum thermal stability factor per line. Therefore, $log_2(N)$ flag-bits are required to be added to the tag array, which is typically based on SRAM technology, for identifying the lines of the same group. After manufacturing, the characterization testing process determines the minimum Δ values per line based on any of the existing retention testing methods [78, 131]. Although the retention testing is non-trivial and time consuming process, it will be performed once after manufacturing. This , however, enables significant improvement of the STT-MRAM design at run-time based on our proposed non-uniform ECC technique.

Once the Δ for each cache line is identified, the classification of cache lines into groups could effectively be done as in [65] using the standard k-means clustering algorithm [132] and the flag-bits associated with the group information of each cache line are set accordingly. The switching operating temperature of the groups is simply saved in a lookup table. At runtime, adjusting the ECC approach of each group is performed based on the operating temperature.

5.3 Simulation Results

This section demonstrates the effect of non-uniform adaptive ECC approach applied on the STT-MRAM-based L1 and L2 caches of a high performance processor. In addition, the Lazy-ECC is compared with the conventional ECC approach where the decoding circuitry has negative impact on the read performance in terms of the overall performance and energy.

Table 5.4: Read and write latencies of 64-bit STT-MRAM L1 and L2 caches with 16 KB capacity supported with Lazy and conventional ECC-based approaches

L1 cache						
Operation		No-ECC	ECC1	ECC2	ECC3	ECC4
Write [ns]		11.610	6.856	5.434	4.630	4.173
Read [ns]	Lazy-ECC	0.898	0.899	0.899	0.899	0.900
	Conventional	0.898	1.479	3.358	4.597	5.614

L2 cache						
Operation		No-ECC	ECC1	ECC2	ECC3	ECC4
Write [ns]		21.00	12.90	10.025	8.33	7.245
Read [ns]	Lazy-ECC	1.120	1.121	1.121	1.121	1.122
	Conventional	1.120	1.701	3.580	4.819	5.836

Table 5.5: Simulation setup in gem5

Processor	Single-core, 2 GHZ, Out-of-order, 4-issue
L1-cache (Data and Instruction)	32 KB, 4-way set associative, 64B line size STT-MRAM, $\Delta = 30$, $I = 3$ diff. read/write latencies obtained from Table 5.4
L2-cache	512 KB, 8-way set associative, 64B line size STT-MRAM, $\Delta = 40$, $I = 2$ diff. read/write latencies obtained from Table 5.4
SPEC2000 Applications	gzip, bzip2, mcf, twolf, vpr sjeng, gcc, art, mesa, lbm

5.3.1 Simulation Setup

For the bit-cell in L1 cache, we used $\Delta = 30$ and $I_w/I_c = 3$ to guarantee fast write operations, while for the L2 cache bit-cell, we assumed higher thermal stability factor $\Delta = 40$ and $I_w/I_c = 2$ to guarantee high retention time with lower dynamic energy consumption.

At architecture-level, in order to compute the memory access latencies (i.e., write and read latency), we model the stochastic behavior distribution using Matlab based on Equation (2.13). 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 and L2 caches of an Alpha processor modeled in gem5 [118]. Table 5.4 reports the breakdown of write and read latencies for Lazy and conventional ECC-based approaches in L1 and L2 STT-MRAM caches. The setup for our system-level experiments are summarized in Table 5.5.

5.3.2 Non-uniform Adaptive ECC Results

According to the MTJ process variation, L1 and L2 cache lines can be clustered into 4 groups based on Δ value of their MTJ-cells, thus the mapping of cache lines to the

Table 5.6: Minimum Δ, switching operation temperature and the percentage of each group for STT-MRAM L1 and L2 caches

	flag-bits of group	Minimum Δ	Switching temperature	Group percentage
L1-cache with Mean Δ value of 30	00	25	$27°C$	7%
	01	27	$53°C$	30%
	10	28	$67°C$	42%
	11	29	77°C	21%

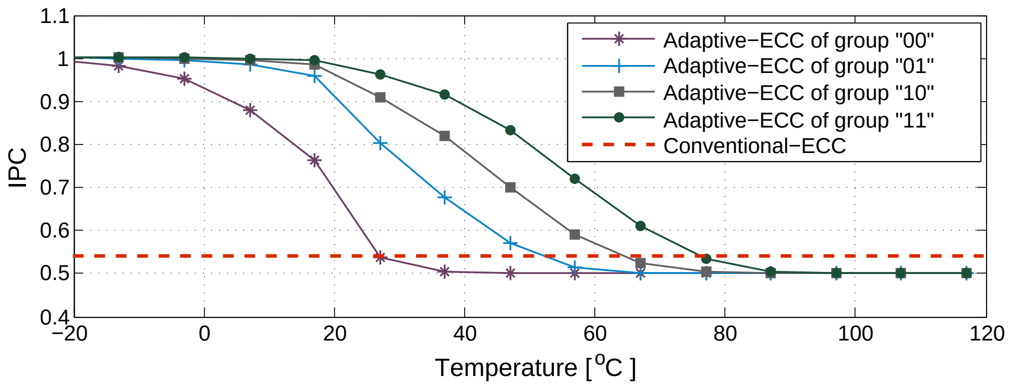


Figure 5.6: Instruction Per Cycle (IPC) of 4-groups-adaptive ECC approach applied for L1 cache, $\Delta = 30$, $I_w/I_c = 3$

groups can be identified by adding 2 flag-bits to the tag array. Additionally, two simple look-up tables of 4 rows are specified to save the operation temperature for switching between the ECC schemes in L1 and L2 caches, where the row indexes define the groups.

In fact, as the L1 cache filters most read and write accesses, the access rate for higher level cache (i.e., L2-cache) is low. On the other side, a higher thermal stability factor is adopted for L2 cache, which reduces temperature and process variation induced failures. Therefore, adopting a pure Lazy-ECC approach in the L2 cache regardless of the process variation and the operating temperature effects has no negative impact on its performance. Thus, it is sufficient to apply the non-uniform adaptive ECC approach for only L1 cache. Table 5.6 illustrates the operating temperature required for switching and the percentage of each group in L1 cache. Figure 5.6 depicts the performance under the temperature variation extracted of the proposed non-uniform and adaptive ECC approach using 4 groups applied for L1 cache.

5.3.3 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 that 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. 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 ECC approaches. For each application, the simulation was conducted over one billion instructions after skipping

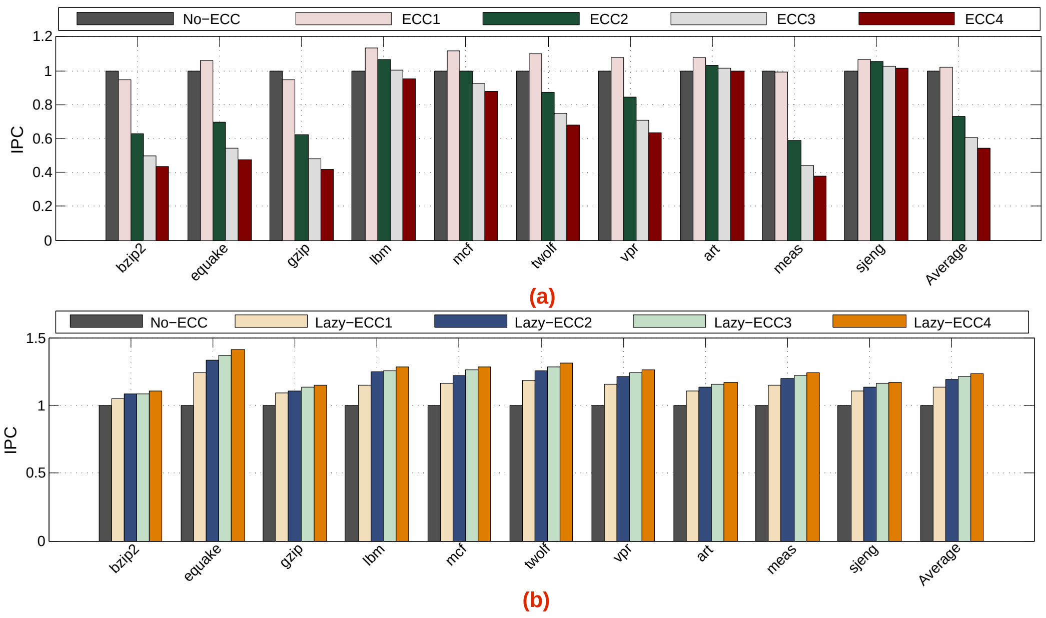


Figure 5.7: Performance of (a) the proposed Lazy-ECC approach and (b) conventional-ECC approach, for various workloads of SPEC benchmarks. All results are normalized to No-ECC

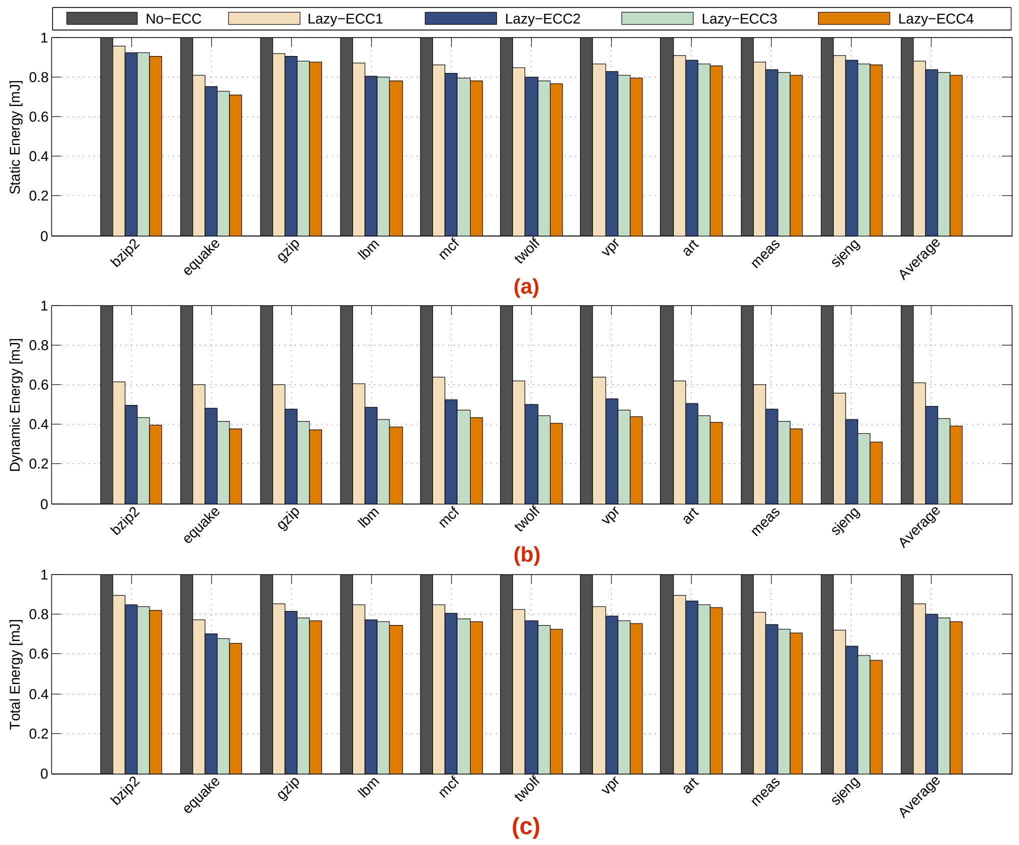


Figure 5.8: L1+L2 energy consumption: (a) static energy consumption (Leakage), (b) dynamic energy consumption (read + write), and (c) total energy consumption (Leakage + dynamic) of the proposed Lazy-ECC and conventional-ECC approach for various workloads of SPEC benchmarks. All results are normalized to No-ECC

Table 5.7: The average reduction of dynamic, static and total energy consumption for L1 and L2 caches of the Lazy-ECC approach

	Lazy-ECC1	Lazy-ECC2	Lazy-ECC3	Lazy-ECC4
Dynamic energy	39%	51%	57%	60%
Static energy	12%	17%	18%	20%
Total energy	15%	20%	22%	24%

one million warm up instructions. The comparison is done based on the *Instruction Per Cycle* (IPC) metric which shows the overall performance.

Figure 5.7(a) illustrates the IPCs extracted by applying the conventional-ECC approach on the L1 and L2 caches of the processor. In the conventional approach, there is an enhancement of the overall system performance only for ECC1 which is on average less than 5%. 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 5.7(b). As shown in this figure, the average IPC improvements of the error bit correction of 1,2,3 and 4 are around 13.7%, 19%, 21.3% and 23.5%, 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. Note 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 and art), 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., equake, vpr and twolf), a Lazy-ECC approach results in a large gain in the overall performance compared to the conventional-ECC.

5.3.4 Energy Analysis

Energy analysis is done for both dynamic and static energy consumption of L1 and L2 caches. In fact, as the L1 cache filters most accesses, the access rate for higher level cache (i.e., L2 cache) is low. Therefore, the access energy is a major contributor for L1 cache, while for L2 cache, the leakage energy (i.e., static energy which is the product of the leakage power and the execution time of the workload) is the dominant portion of its total energy. Consequently, the total energy consumption will be affected significantly by reducing the write latency from L1 cache perspective, which is reduced by Lazy-ECC by up to 3X. On the other hand, from L2 cache perspective, the execution time decreases from No-ECC to Lazy-ECC up to 23.5% because of two reasons: i) fast write termination, ii) the exclusion of the ECC decoding latency from the read path. Consequently, the static energy decreases. The static, the dynamic and the total (L1 and L2 caches) energy consumption for our proposed opportunistic write with Lazy-ECC approach along the conventional-ECC approach are shown in Figure 5.8(a), (b) and (c), respectively. Table 5.7, presents the reductions in the dynamic, static and total energy consumption of Lazy-ECC approach.

5.3.5 Related Work

A First-In First-Out (FIFO) based technique is proposed in [133], 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 5.1), respectively. This means that the depth of the FIFO at each input is at least 19 bits. Since 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 solution (5320 flip-flops vs. few combinational gates). Moreover, the main shortcoming of a FIFO-based technique is that it fails to capture all system inputs when errors occur 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 the processor inputs during the error correction time could be lost, resulting in failure. Therefore, in applications with a high error rate, our proposed technique provides a 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 6.6×10^{-4}. Hence, the probability of having write error in each cycle is 5.3×10^{-5}. Consequently, the probability of having the second write error in a timing window of 19 cycles after the first write error could be computed based on the exponential distribution to be $1 - e^{-\lambda \times cycle}$ where λ is 5.3×10^{-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 window 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.

5.4 Summary

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 the high decoding latency. In this chapter, we presented an opportunistic write technique equipped with our Lazy-ECC approach to exclude the decoding latency from read access by performing speculative computation based on unchecked data. We have also provided a non-uniform and adaptive ECC approach to address temperature and process variation induced failures at runtime. The proposed approach was implemented on the L1 and L2 caches of a high performance processor. The simulation results reveal that this approach significantly improves the overall performance and reduces the energy consumption by up to 23.5% and 24%, respectively, compared to the conventional approach with a negligible area overhead in the processor.

Chapter 6

Adaptive Approach for STT-MRAM Performance and Power Optimization

At system-level, two opposing forces of energy consumption are at play, as the system energy is the product of power and time. Any increase in the write current increases the power, but reduces the latency, hence there is a minimum energy point corresponding to the optimal write current. Based on the sensitivity of the application to the latency of the write operation, which correlates to write access rate, we can adaptively increase the write current for effective performance improvement.

Thus, we propose a dynamic write approach for STT-MRAM to adjust the write current value on-the-fly according to the performance needs.

In this chapter, we make the following contributions:

- We identify the opportunity of using an adaptive write current scaling approach for STT-MRAM by presenting a detailed energy-performance trade-off at device, architecture and system levels.
- Based on our cross-layer analysis, we obtain the optimal write current levels for different performance and energy needs of the system.
- We predict the sensitivity of the application performance to write latency, based on the write rate, to adjust the write current level at run-time.
- We allow to control the write current at run-time by modifying the write circuitry of our STT-MRAM design, which allows multiple levels of write currents.

We evaluate the proposed approach by analyzing the performance and energy consumption of a system for both L1 and L2 STT-MRAM caches using detailed simulations of SPEC2000 workloads. The results demonstrate that the effective write latency across all application phases is reduced by 52.4% and 55.7% for L1 and L2 caches, respectively. This corresponds to a system performance improvement of 15.5%, on average. The associated memory dynamic energy overhead is 1.7% because of the write current increase, and memory leakage energy overhead is 1.5% on account of the proposed write circuit.

6.1 Adaptive Write Current Scaling

In this chapter, the STT-MRAM design is improved based on the strong correlation between the write access rate and the write latency to maximize the performance along with minimal energy overhead. At run-time, the write current can be adjusted to different values (e.g., high, medium and low). In situations, where the write rate is high, the overall performance is very sensitive to the write latency. Thus, it is reduced by adjusting the write current. In contrast, when the performance is less sensitive to the write latency, the write current is reduced to save write energy. To enable such an adaptation, the overall execution time is divided into distinct phases, based on the sensitivity of the performance to the write access rate. For optimal performance improvements, we need to find an optimal write current value for each run-time phase. Therefore, this chapter provides an analysis of write latency and its energy relation from device to system level, which accordingly is used for run-time phase identification and optimization.

6.1.1 Write Latency and Energy Relation

6.1.1.1 At device-level

The opportunity of improving the performance by increasing the write current at device-level is illustrated in Figure 6.1. This figure shows the write energy and the write latency relations with the write current for both L1 and L2 caches. For L1, since the write operation is on the critical path, simple ECC-1 for one bit error correction is used to reduce the impact of stochastic write by around 50%, at a cost of only one cycle decoding/encoding penalty per each read/write access. Moreover, the adopted thermal stability factor (Δ) is low to guarantee fast write operations. Figure 6.1(a) illustrates the energy and latency plots versus the write current for target WER of 10^{-9} with $\Delta = 30$. On the other hand, the normal WER of 10^{-18} is considered for L2 with higher Δ equals to 40 to guarantee high retention time (see Figure 6.1(b)). As write energy and latency plots in Figure 6.1(a) and Figure 6.1(b) present, the energy consumption is minimized for the write current values of $102\,\mu$A and $82\,\mu$A, which correspond to write latencies of 19.8 ns and 10.7 ns for L2 and L1-cache lines, respectively. However, the write latencies decrease modestly along with a significant increase in the write energy for a higher write current. The figure shows that by increasing the write current two times, the write latency decreases by 66%, while the related write energy increases by 35% per access for L1 and L2 caches. It is important to note that such an increase in the write current does not impair the MTJ reliability due to time dependent dielectric breakdown (TDDB) of the oxide barrier.

6.1.1.2 At architecture-level

The continuous extracted device latency is from a system-level perspective quantized into multiple clock cycle times with a fixed length, thus, the write latency and energy are discretized. Therefore, the curves of the write energy and latency against the write current are no longer smooth or monotonic (see Figure 6.2). As shown in Figure 6.2, different energy points have the same write latency, which in turn reduces the search space for the energy-latency optimal pareto points. Furthermore, Figure 6.2 illustrates

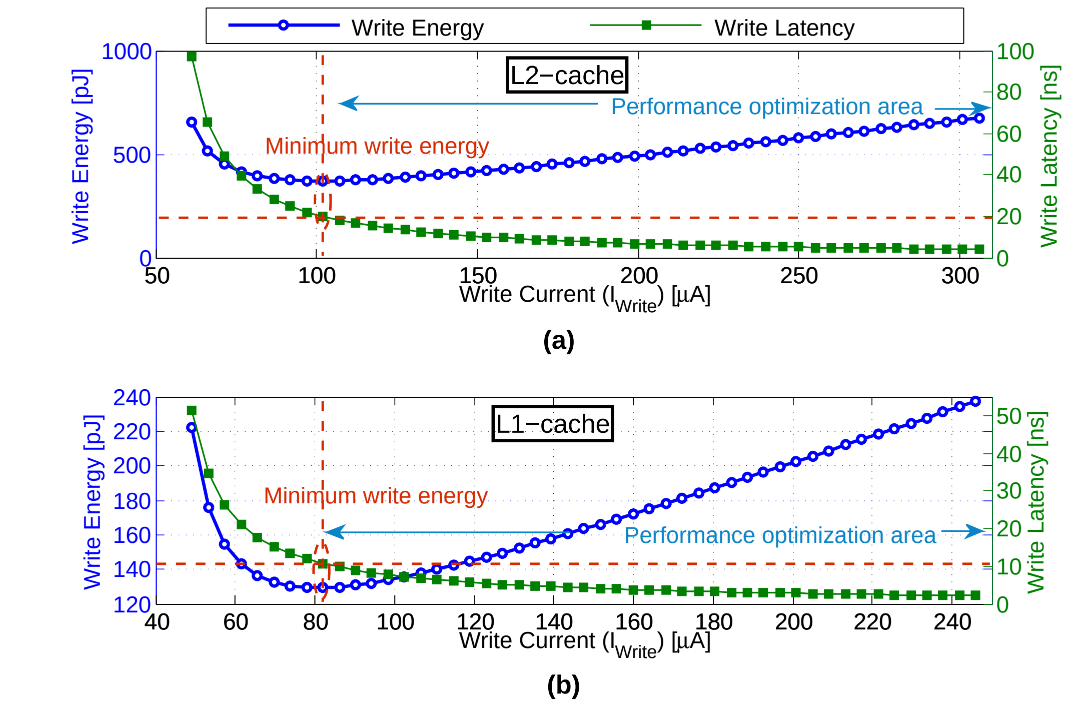


Figure 6.1: Impact of increasing the write current on write energy consumption and write latency for a 512-bit cache line in case of 'AP' switching at device-level [$\Delta = 30$ for L1, $\Delta = 40$ for L2, detailed setup in Table 6.3]

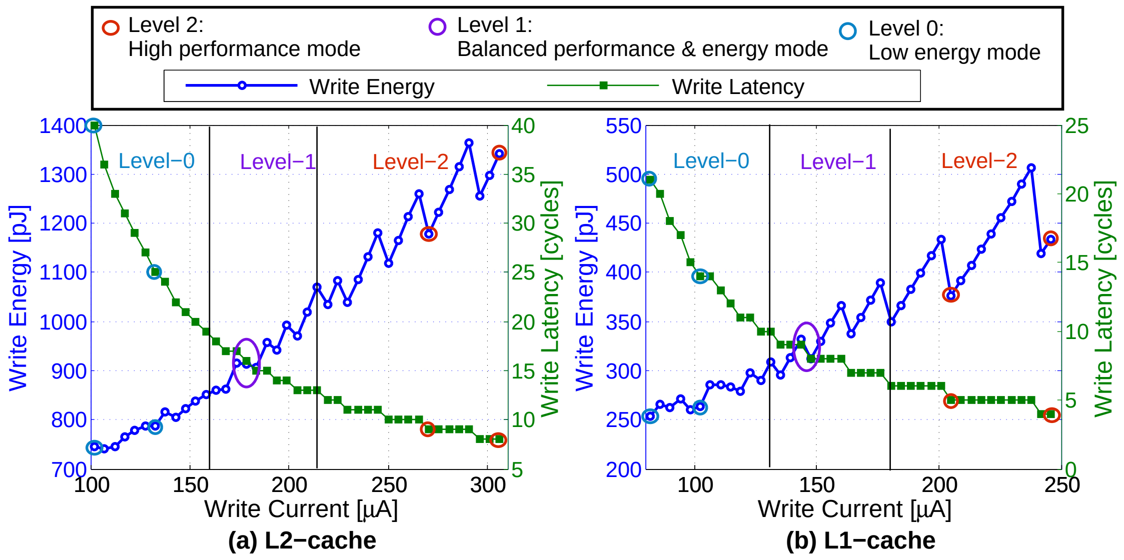


Figure 6.2: Impact of increasing the write current on write energy consumption and write latency for a 512-bit cache line in case of 'AP' switching at system-level [$\Delta = 30$ for L1, $\Delta = 40$ for L2, detailed setup in Table 6.3]

that the energy and latency curves are spread over three regions of the write current (low, medium and high). This helps to determine the initial candid points for all regions. Identifying the optimal pareto points needs for each level (i.e., Level-0, Level-1 and Level-2) system-level results (i.e., system performance and energy) with detailed memory access patterns of applications. This enables to identify the impact of write latency and energy on the overall system performance and energy, respectively.

Table 6.1: STT-MRAM design configuration for L1 and L2 caches

				Write access rate					
				Low		Medium		High	
				Energy-efficient mode Level-0 configurations		Balanced-energy-performance mode Level-1 configurations		Fast-performance mode Level-2 configurations	
Cache level	Δ	WER	Candidate	Current [μA]	Latency [ns]	Current [μA]	Latency [ns]	Current [μA]	Latency [ns]
L1	30	10^{-9}	Point-1	82	10.7	147.6	4.1	200.9	2.8
			Point-2	102.2	7.1	164	3.6	246	2.1
L2	40	10^{-18}	Point-1	102	19.9	183.6	7.7	275.4	4.5
			Point-2	132.6	12.5	193.8	7.1	306	4

Table 6.1 summarizes the STT-MRAM parameters for both device and architecture levels, including two candid points (Point-1 and Point-2) for each scaling level. The selection of the actual point requires a system level analysis which will be described next.

6.1.2 Proposed Write Phase Prediction Methodology

Our phase analysis method is split between an off-line write rate analysis and a run-time write rate prediction.

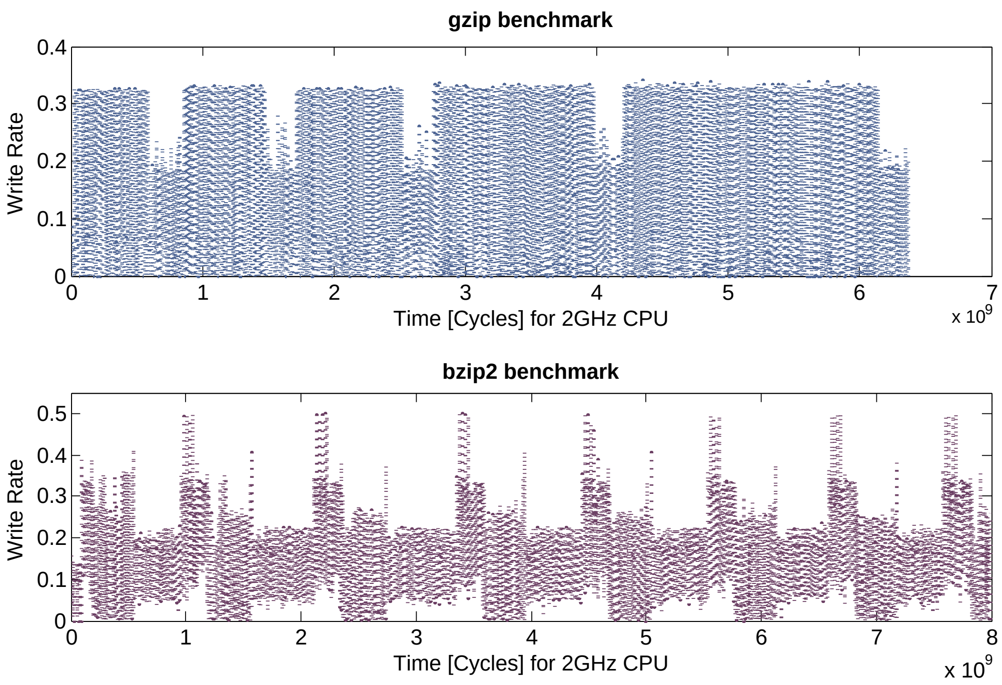
6.1.2.1 Off-line Write Access Analysis

The write access phase is defined as a period of execution time that exhibits a consistent write access rate which is distinguishable from the write rates in the other phases. From our off-line analysis, we generate a set of training data for run-time control using memory intensive workloads of the SPEC2000 benchmarks suite. From the write access counter measurements, we observed the following: i) Write rate can significantly vary over the time. ii) Write rate is highly periodic in repetitive and long-stable patterns for more than 90% of the entire run-time for all executed benchmarks. iii) While the periods and write rates are different for L1 and L2 caches, the repetitive and stable behavior is common for both. Figure 6.3 plots write rate values for L1 and L2 caches over the entire execution time for the compression programs bzip2 and gzip. While we have done the analysis for all memory-intensive benchmarks and observed similar trends, we only show these two benchmarks for brevity.

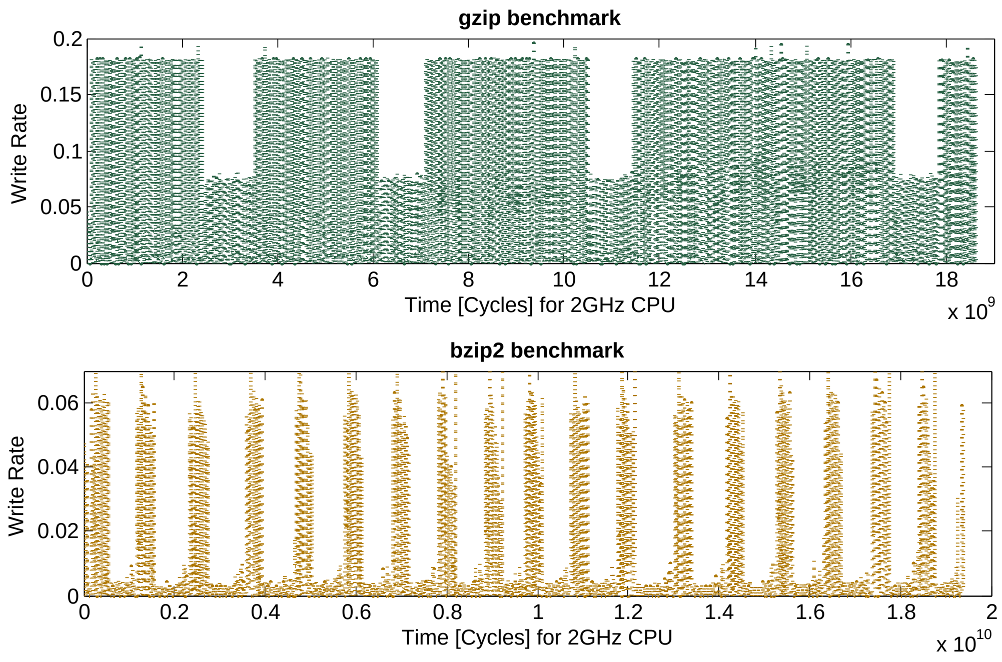
It is important to note that the amount of the performance improvement offered by the adaptive write current scaling approach is derived from the amount of the Instruction Per Cycle (IPC) improvement. For on-chip memory, the IPC is closely linked with the number of memory write accesses. As the number of write accesses increases, a significant acceleration for the write operation is required to reduce the IPC degradation due to the long write latency. On the other hand, for low write rates, the IPC is not significantly affected by the adopted write latency. Therefore, in this case, the write parameters corresponding to the minimum energy point will be considered. This is the foundation for building our approach based on the off-line write phases analysis.

6.1.2.2 On-line Write Current Control Algorithm

The off-line analysis of stable write rate phases is the main guide to predict the write rate at run-time. However, the major challenge of the predictor is to accurately identify the



(a) L1-cache write access rate



(b) L2-cache write access rate

Figure 6.3: Off-line write access phase analysis for L1 and L2 caches

correct time for changing the write current value (i.e., switching between modes). This relies on identifying the stable duration of write rate. The simplest way is to perform the prediction periodically. Therefore, we divide the execution time into small windows with a constant length. Each window consists of two parts: i) *Tracing Period* (TP), and ii) *Stable Period* (SP). SP is orders of magnitudes longer than TP. The tracing period information tells the predictor that the write rate, which was stable during the last tracing period, will be stable for the next stable period. Based on the off-line write

Algorithm 4 Control algorithm for the $(i+1)^{th}$ iteration

```
Input    1- Off-line analysis:
{Window length $L_W$, tracing period $TP$}
         2- Run-time history information:
{Run-time phase ($Phase_j$), write rate of the previous window $WR_{Wi}$}
         3- STT-MRAM design configurations:
{Look-up table for available write current levels with correspond write rates }
Output   1- Write rate for the current window $WR_{Wi+1}$
         2- Adopted write current ($I_w$) for the stable period $SP_{Wi+1}$
         3- Run-time phase for $W_{i+1}$
 1: For run-time from end of $W_i$ to end of $TP_{Wi+1}$ do
 2:     Calculate the number of write operations
 3: end for
 4: $WR_{Wi+1} \longleftarrow \frac{total\ write\ operations\ of\ TP_{i+1}}{TP}$
 5: if $WR_{Wi}$ != $WR_{Wi+1}$ then
 6:     End the current run-time phase ($Phase_j$)
 7:     Set correspond write current $I_w$ for $WR_{Wi+1}$
 8:     Start new run-time phase ($Phase_{j+1}$)
 9: end if
10: return $Phase_{j+1}$, $WR_{Wi+1}$ and $I_w$
```

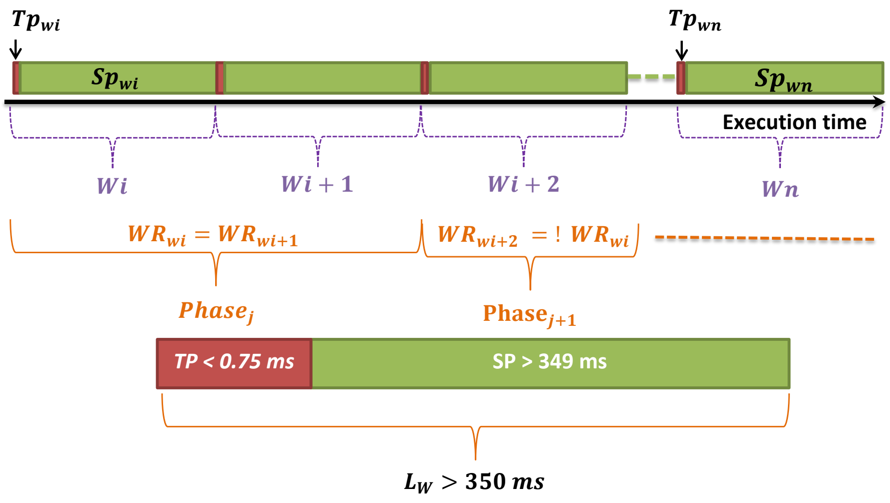
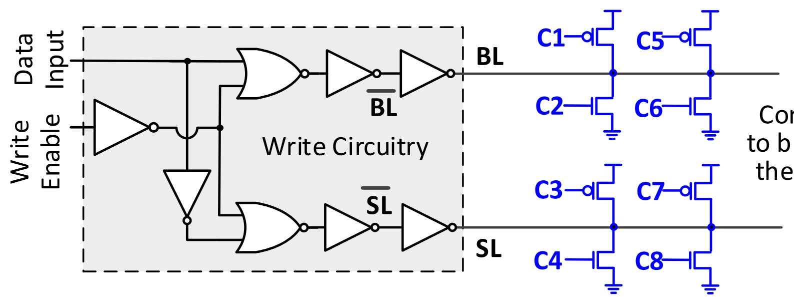


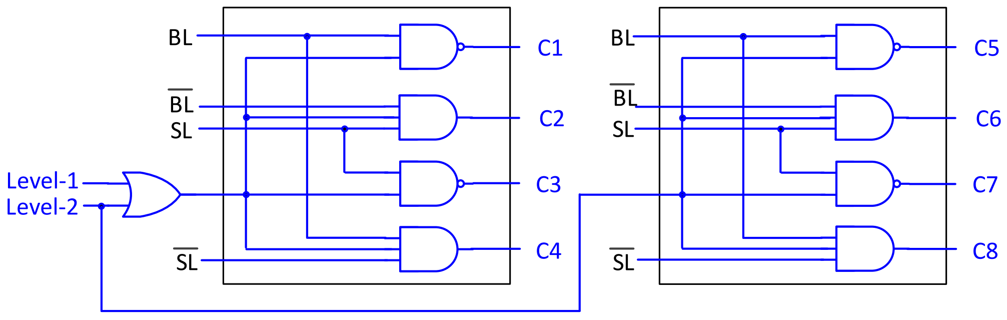
Figure 6.4: Run-time write access rate prediction

rate analysis, we identify the possibility of dividing the execution time into windows. The optimal length of the run-time window is determined by greatest common divisor of all observed off-line stable phases.

At run-time, the proposed predictor is implemented by a periodic control algorithm. The data learned during off-line analysis such as the window parameters (i.e., length, TP and SP) and the observed levels of the write rate with the required write current values are passed to the control algorithm. The pseudo-code of the iterative control algorithm is presented in Algorithm 1, which explains the diagram of run-time phases as shown in Figure 6.4.



(a)



(b)

Figure 6.5: Circuit implementation for the adaptive write technique. (a) Schematic for the dynamic write. (b) Circuit diagram of the control components

Table 6.2: Truth table for control circuit

Level-1	Level-2	BL	SL	Operation	Mode
X	X	0	0	No write operation	—
0	0	0/1	1/0	Normal write operation	Energy-efficient
0	1	1	0	C1&C4 are ON	Balanced-
0	1	0	1	C2&C3 are ON	energy-performance
1	X	1	0	C1&C4&C5&C8 are ON	Fast-
1	X	0	1	C2&C3&C6&C7 are ON	performance

6.1.3 Modification of Write Circuitry

The circuit-level implementation of the proposed write circuitry to enable the selection of multiple levels of write currents is shown in Figure 6.5. Here, the write circuitry is the same as the one used for the standard write operation. The write circuitry is optimized to deliver the current that is adjusted based on the minimum energy point (i.e., energy-efficient mode) by default. In order to increase the write current and switch between levels, several transistors are employed that are activated using control circuits (C1, C2, .., Cn). Table 6.2 shows the truth table for switching between the three levels.

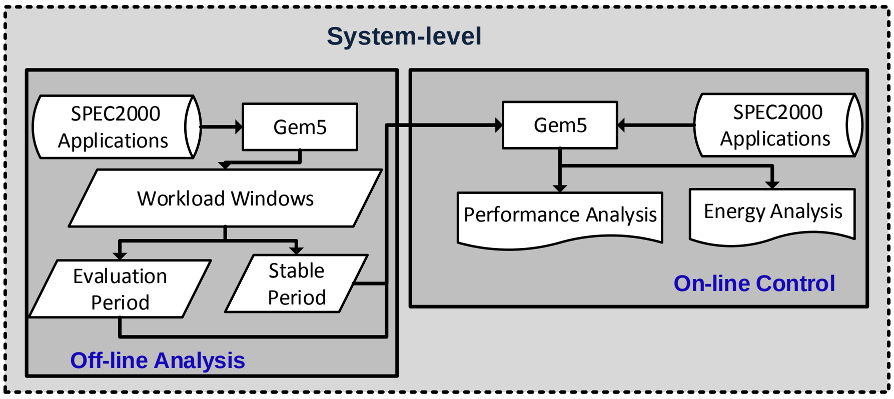


Figure 6.6: Proposed cross-layer tool flow

Table 6.3: Simulation setup

Device model	Perpendicular STT-MRAM model with radius of 20 nm [120]
gem5 confiquration	ISA ALPHA
Processor	Single-core, 2 GHz, Out-of-order, 4-issue
L1 cache	64 KB, 4-way set associative, 64B line size STT-MRAM, $\Delta = 30$, different write latencies
L2-cache	512 KB, 4-way set associative, 64B line size STT-MRAM, $\Delta = 40$, different write latencies
SPEC Applications	gzip, bzip2, mcf, twolf, vpr, sjeng, lbm

6.2 Simulation Results

6.2.1 Experimental Setup

Figure 1.4 demonstrates the framework adopted at device and architecture levels to analyze the efficiency of the proposed idea. However, at system-level, as seen in Figure 6.6, there are two sub-components: off-line analysis and on-line control. In Off-line analysis, we simulated various applications of the SPEC2000 benchmark suite for the entire execution time with extended gem5 simulator [118] . On the other hand, in on-line control, performance and energy analysis of the proposed approach are done for L1- and L2-caches. For each SPEC2000 executed application, the simulation was conducted over the entire execution time and over 0.5 ms for off-line analysis and on-line control estimations, respectively. Table 6.3 summarizes the set-up for our experiments.

6.2.2 SPEC2000 Behavior Analysis

From our off-line analysis, it can be noticed that the write rates of the SPEC2000 workloads for both L1- and L2-cache levels are repetitive and predictable over the run-time. The phases can be clustered into three levels of the write rate, high, medium and low, see Table 6.4. The greatest common divisor of all stable off-line phases is 350 ms. This identifies the length of the run-time window. Whereas, the tracing period is determined by observing the longest time of the unstable write current value, which is less than 0.75 ms, as shown in Figure 6.4.

Table 6.4: Write rate phases of SPEC2000

Write rate	L1-cache	L2-cache	Application
Low	$WR < 0.1$	$WR < 0.05$	bzip2,lbm,vpr
Medium	$0.1 < WR$ $WR < 0.2$	$0.05 < WR$ $WR < 0.1$	gzip,bzip2 twolf,sjeng
High	$0.2 < WR$	$0.1 < WR$	gzip,mcf

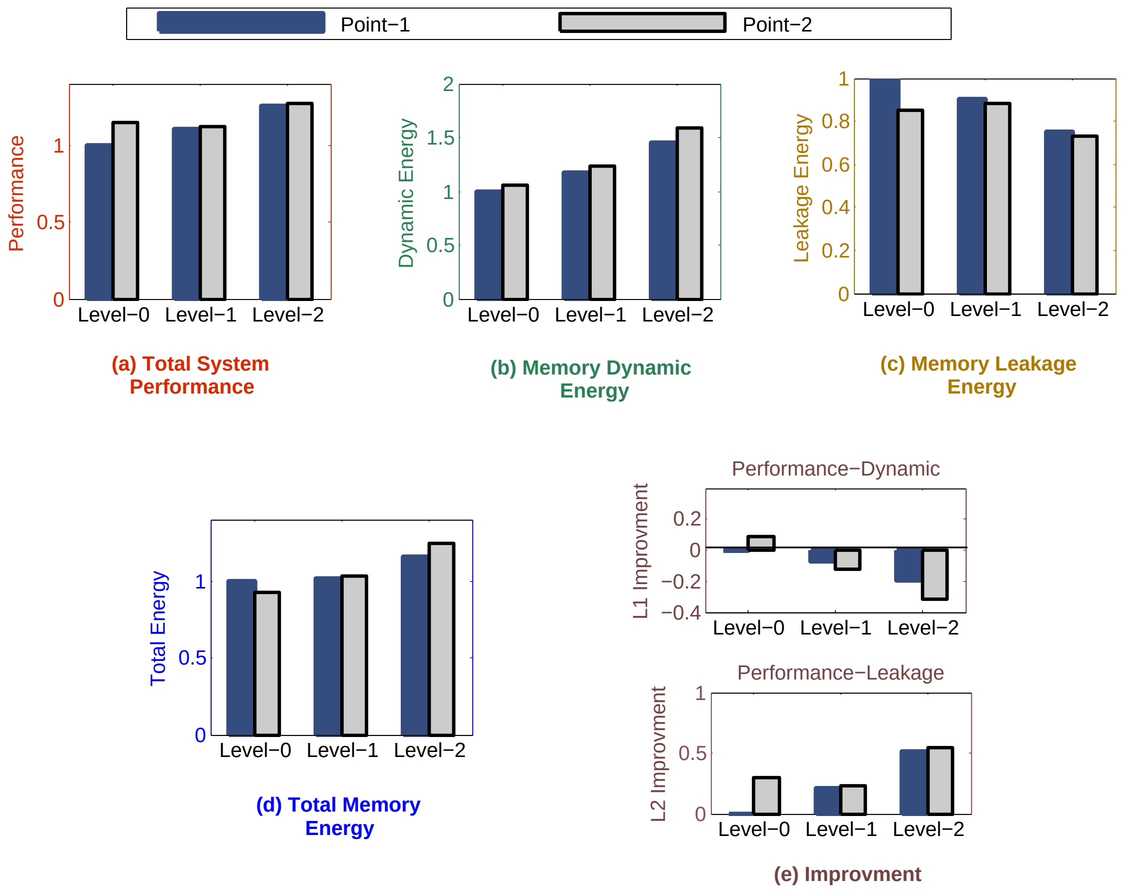


Figure 6.7: Performance and total energy overhead (Dynamic+Leakage) for L1 ad L2 caches of the proposed approach normalized to standard STT-MRAM design without including (Dynamic+Leackage) overhead of the additional circuitry

6.2.3 Performance and Energy Analysis

For a direct comparison between a standard STT-MRAM design and our adaptive write current scaling approach, we normalized all performance and energy results to the extracted results of the minimum energy point (default current value), which refers to Point-1 of Level-0 (see Table 6.1). In fact, as the L1-cache filters most accesses, the access rate for higher level cache (i.e., L2-cache) is low. Therefore, the leakage energy is responsible for 60% of the overall energy consumption for L2-cache. In contrast, for an L1-cache, the access energy is the major contributor. Figure 6.7 illustrates the effects of the write current levels on the IPC and energy consumption for different SPEC2000 applications.

For the energy-efficient mode (i.e., Level-0), the total IPC of the low write rate phase is optimized with the second candidate (i.e., Point-2) by 14%, compared to the baseline (i.e., Point-1). This in turn results in reducing the leakage energy, which is the dominant portion of the total energy for L2-cache, by 15%. However, the dynamic energy is increased by 5%. Consequently, the total energy consumption is reduced by 8%.

For the balanced-energy-performance mode (Level-1), the performance (i.e., IPC) is improved by 10% and 11%, and the leakage is reduced by 10% and 12% for Point-1 and Point-2, respectively. However, the dynamic energy is increased by 17% and 23% with Point-1 and Point-2 compared to the standard design (i.e., Point-1 of Level-0). This leads to increase the total energy consumption by 1.5% and 3.5%.

In the high-performance mode (Level-2), the performance is optimized by 25% and 27% along with increasing the dynamic energy by 45% and 58% for Point-1 and Point-2, respectively. Whereas, leakage dissipation is reduced by 15% and 17%. Therefore, total energy dissipation increases by 15.3% and 24.3% for Point-1 and Point-2, respectively.

As illustrated in Figure 6.7b (e), Point-2 of level-0, Point-1 of level-1 and level-2 are considered as optimal current points for L1 cache, since they offer maximum IPC with minimum dynamic energy consumption. Whereas, for L2 cache, where leakage energy is dominant, Point-2 of level-0, level-1 and level-2 are considered as optimal points. Table 6.5 summarizes the percentage of IPC and energy overhead improvements per phase of the proposed write approach over the standard one.

In order to estimate the overall improvements of the proposed adaptive approach, the energy overhead and performance improvements over each phase and the total breakdown of the execution over different phases should be considered. Our results show that the effective write latency for L1 is reduced by 52.4% (from 21 cycles to 10 cycles), while for L2, it is reduced by 55.7% (from 40 cycles to 18 cycles). Figure 6.8 illustrates the overall performance improvement and energy overhead for different SPEC2000 applications for the proposed approach. It can be seen, that our approach improves the IPC compared to standard design, on average, by 15.5% at the cost of increasing total memory energy dissipation by only 1.7%. It is worth mentioning that the proposed approach does not impair the MTJ reliability due to TDDB. Firstly, the increased current in the highest level is within the device tolerance. Secondly, this current increase happens only during write intensive phases of application execution, and the overall impact is limited.

It is important to note that the amount of IPC improvement and energy overhead due to the proposed approach are application dependent. For instance, for the applications with a low write access rate (e.g., lbm and vpr), there is no energy overhead compared to the standard design, as the energy-efficient mode has been considered. However, the total energy consumption is reduced by 8% because of improving the performance by 14%, which in turn reduces the leakage energy significantly, while the dynamic energy negligibly increases. However, for write intensive applications (e.g. mcf and gzip) with several high performance phases (level-2), the differences of the dynamic and the total energy dissipations between the proposed and the standard design become significant, up to 45% and 15.3%, respectively (see Figure 6.8).

Table 6.5: Percentage of IPC improvement and energy increase of the proposed approach over standard STT-MRAM design

Power mode	Write Rate	Point	Increase of write current %		Reduction of write latency %		Increase of write energy %	Optimization of total performance %	Overhead of total energy %
			L1	L2	L1	L2			
Energy-efficient	Low	Point1	0%	0%	0%	0%	0%	0%	0%
		Point2	24%	29%	32.8%	27.8%	3%	14%	-8%
Balanced-energy-performance	Medium	Point1	70%	79%	58%	67%	38.8%	10%	1.5%
		Point2	100%	135%	62.8%	69.8%	45,7%	11%	3.5%
Fast-performance	High	Point1	140%	169%	76.6%	75%	64.6%	25%	15.3%
		Point2	200%	200%	81.3%	77.4%	76,6%	27%	24.3%

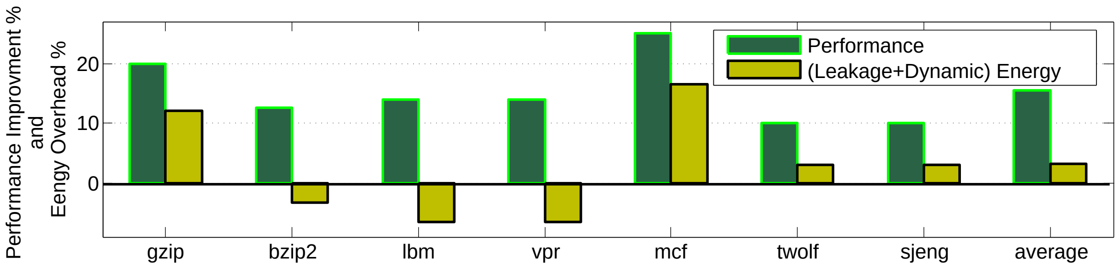


Figure 6.8: Performance improvement and energy overhead (Dynamic+Leakage) of the proposed approach for various SPEC2000 workloads compared to standard approach including (Dynamic+Leackage) overhead of the additional circuitry

Table 6.6: Area and energy overheads of circuit-level implementation of proposed approach for L1 and L2 caches

Cache	Area	Leakage	Dynamic energy
L1	7.6%	1.4%	1%
L2	1.4%	0.1%	0.5%

6.2.4 Area-overhead Evaluation

We need to organize the write circuit components in such a way that it is able to facilitate the highest required current in our proposed technique. This is handled by adding some additional drivers and its controlling circuitry as described in Fig 6.5. These additional circuitry imposes area overheads of 7.6% and 1.4% for L1 and L2 caches, respectively. This additional circuitry is employed column-wise and due to the small size of bit-cell array in L1 cache, it incurs a relatively high area overhead ratio compared to the L2 cache. Furthermore, the dynamic energy and leakage for our proposed circuit are also increased as mentioned in Table 6.6, due to addition of the extra components. On the other hand, the hardware cost associated with our control algorithm consists of a 16-bit counter and a simple look-up table for adjusting the write current based on the write rate of each level.

6.3 Summary

The write operation in STT-MRAM is the main bottleneck for the performance, energy and reliability of STT-MRAM. Traditionally, the write current is set to the minimum energy point. In this chapter, by analyzing the behavior of the workloads, we observe that based on the write access rate of the application, the sensitivity of the overall performance to write latency changes. Based on this observation, we have developed a cross-layer adaptive write current level scaling scheme, in which a runtime predictor is used to predict the write access rate, and the write circuitry is modified to allow adjusting the write current levels at runtime. A detailed cross-layer analysis is performed to obtain the optimal write current level for the corresponding write access rate phases. Our results demonstrate that we can improve the overall system performance by 15.5% for a maximum energy overhead of 3.2% with 7.6% and 1.4% area overheads for L1 and L2- cache levels, respectively.

Chapter 7

Leveraging Systematic Unidirectional Error-Detecting Codes

Recent studies show that a robust ECC with multiple bit correction capability could be used to correct various types of errors, such as read disturb failures, retention failures [78, 134] and write errors that occur in the tail of the stochastic write distribution [94, 96].

However, the decoding latency of robust ECC is relatively high (i.e., multiple cycles) [98], causing read access to become the performance bottleneck for fast caches. Furthermore, as the correction capability of ECC grows, the storage overhead of the check bits significantly increases, which in turn erodes the overall performance. Therefore, the conventional multiple error correction approach is quite ineffective for fast caches.

In general, the MTJ cell has asymmetric switching, which means that one of the switching takes longer time than the other one. Moreover, the one that takes longer switching time also has a wider distribution that eventually represents the overall worst case switching scenario. Hence, the total write latency has to be evaluated based on this worst case switching.

Additionally, since the read current flows uni-directionally through the cell, the unwanted switching that may occur during the read operation (*read disturb failure*) also affects only one type of switching.

Such unidirectional mode of error patterns can be exploited using an efficient asymmetric coding, which, however, can only be used for error detections. By adopting an appropriate cache access policy, since the (correct) copy of the data will be available in the lower level caches, error detection in the first level caches would be sufficient. Considering these facts in order to find the opportunity of using STT-MRAM as a universal memory technology including fast caches, we propose to use effective *Systematic Unidirectional Error-Detecting Code* (SEDC) [75], which is an optimal code for detecting asymmetric errors. Together with the necessary adjustments to the cache access policies, we can significantly reduce the decoding/encoding latency as well as the check-bit storage overhead.

Our proposed technique results in: i) performance and energy improvements of STT-MRAM in on-chip memories (i.e., L1 and L2 caches of high performance processors) up to 12% and 3%, respectively, ii) significantly reducing the error rate of various error sources in STT-MRAM and iii) lower storage overhead from 87% to 96% compared to various ECC schemes.

7.1 Proposed Approach

In this chapter, We leverage the asymmetric nature of errors in STT-MRAM, and using an asymmetric error detection method, the erroneous data, due to unfinished switching, is detected. Furthermore, with an adjustable cache policy, the correct data can be fetched from the lower cache level, upon error detection.

7.1.1 Problem Statement

The main drawbacks of STT-MRAM are high write latency and poor reliability. In fact, there is a big gap between STT and SRAM write latencies which can reach up to 17 times for L1 caches (0.7 ns for SRAM and 11.6 ns for STT-MRAM, based on the setup outlined in Section 7.2.1), and hence, it will be relatively hard for STT-MRAM to replace SRAM in fast memory applications. As shown in Section 2.3.4, lowering Δ or using robust ECC cannot resolve both the latency and reliability challenges in STT-MRAM. Thus, we propose a solution to facilitate STT-MRAM for the fast cache applications. We leverage the physical nature of the MTJ device characteristics in order to reduce the costs of the adopted error codes. It is based on the fact that the 'AP' switching has a wider distribution with a longer tail compared to the 'P' switching, thus it determines the write latency, energy and error rate. This means that any scaling in the write margins affects only one type of switchings (i.e.,'AP'). Furthermore, the high resistance value of the MTJ (i.e., 'AP' magnetization) is more likely to be unstable during retention time. Consequently, errors in STT-MRAM are uni-directional ('AP' switching). This feature can be exploited to generate efficient error coding schemes.

7.1.2 Optimal Error Detecting Code For STT-MRAM

An extensive theory of symmetric error code has been developed [70, 135, 136], where symmetric errors '0' and '1' have exactly the same probability to appear in the reserved data. However, according to the previous discussion, errors due to unfinished switching or other failures (retention, read disturb, etc) in the MTJ device are of asymmetric nature. For this purpose, various asymmetric error codes have emerged, which can be used in STT-MRAM to address the reliability and performance concerns very efficiently. Unfortunately, asymmetric error codes can only be used for error detections. Among all codes, *Systematic Unidirectional Error-Detecting Code* (SEDC), which is proposed in [75], is the optimal one for detecting asymmetric errors, because of the following reasons: i) this code is systematic where data bits are separately identified from the check bits, resulting in parallel performing of decoding/encoding and data manipulation and ii) this code is ideal when a fixed number of errors at run time has to be detected. This can be decided at design time based on the target error rate and the expected number of unfinished switching bits per word. Furthermore, this code needs very simple encoder/decoder circuits, and it is capable to detect a large number of errors up to;

Table 7.1: Maximum number of errors detected by SEDC

No. of check bits (r)	No. of errors detected (e)	
5	11	
6	20	
7	37	
8	**70**	**Adopted SEDC in our work**
9	135	
10	264	

$e = 5.2^{(r-4)} + r - 4$ errors with r check bits. The error-detection capability of SEDC (r, e) is presented in Table 7.1. Since the number of errors in one line-cache of size 512 bit will not exceed 70, SEDC (8,70) accommodates STT-MRAM to meet L1 cache needs.

As SEDC code is able to only detect the errors, the correct copy of data has to be provided. Fortunately, as our target is the fast caches (e.g., L1), a correct copy of the data exists in the lower level caches. The details of the adopted cache policies to guarantee this feature are explained next.

7.1.3 Cache Write Policy

When the processor executes a write request, the modified contents of the accessed addresses will be eventually reflected to all different levels of the memory hierarchy in two general policies, namely *Write Through* (WT) and *Write Back* (WB) [137].

If WT policy is adopted between two particular levels (e.g., L1 and L2), the data will be written on the higher level (i.e., L1) and immediately passed to the next level (i.e., L2). This ensures that the contents of the two mentioned levels are consistent with each other, and hence, all data in the higher level is clean. On the other hand, with WB policy, the data is written only to a particular level of memory system, and consequently, all modified data at that level is then labeled as dirty copies, which will be updated in the lower level at replacement time. This means that at some particular time intervals, there is only one valid copy of the data in the cache hierarchy and if that data becomes erroneous, there is no alternative. In such case, SEDC is not enough and robust ECC is necessary.

There are issues of performance, reliability and costs in order to choose the optimal policy for write requests. The WB policy has better performance than WT policy because it reduces the number of write operations to the lower level of memory (reducing memory traffic). However, with this performance improvement, there is a risk of losing the dirty data in case of errors. The above observation motivates us to use WT policy in case of employing SEDC code to support STT-MRAM memory.

7.1.4 STT-MRAM Improvement With SEDC Code And WT Policy

To fulfill our main goals of improving STT-MRAM performance and reliability, the combination of SEDC code and WT policy can be exploited to reduce the write margin to a large extent. With this way, detection will be carried out for unfinished switching bits in the tail of the stochastic write distribution by SEDC code. The error-free copy,

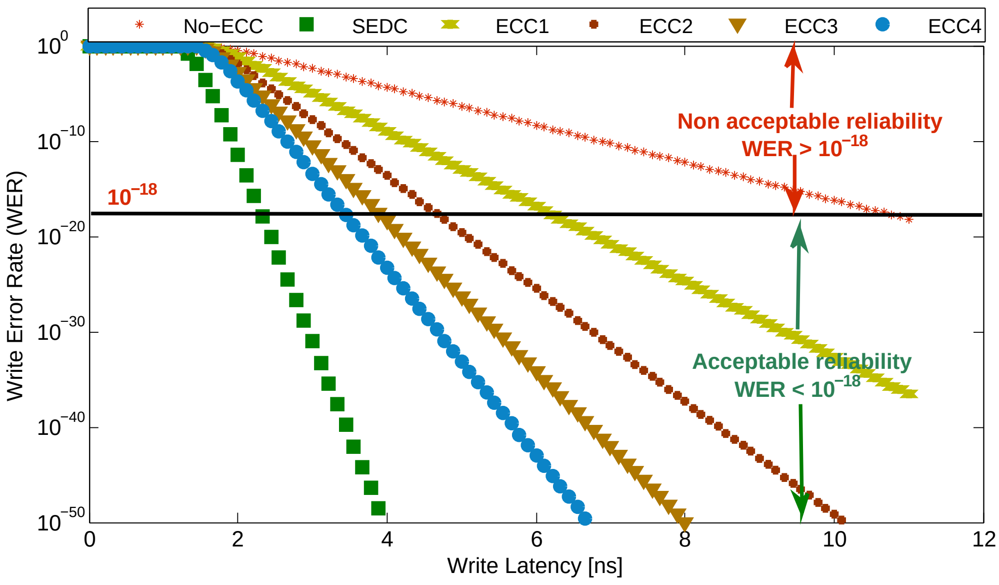


Figure 7.1: WER versus write latency for 64-bit data with different ECC schemes [excluding ECC encoding, $\Delta = 30$, $I_w/I_c = 3$, see Section 7.2.1 for setup]

which is already available due to chosen WT policy, can be obtained from the lower cache level, when error is detected. In addition to reducing the write margin to the acceptable level for fast caches, SEDC code also increases the overall reliability significantly, thanks to its capability to detect far more errors than ECC. The relationship between write latency and write error rate for various ECC schemes and SEDC is shown in Figure 7.1. As seen, compared to ECC, SEDC is able to reduce the write latency for a given WER value.

7.1.5 Optimal Retention Time And Write Latency Estimations

As previously discussed, reducing the write margin can be achieved by reducing Δ which in turns reduces the retention time as well. However, we have to keep the associated retention time reasonable according to the maximum possible time that data resides in the particular cache level. According to [48], $\Delta = 30$ provides a retention period of hours on average which is suitable for L1 caches. However, the estimation of the actual retention time with corresponding probability of the retention failure is crucial to ensure the integrity of the system. According to our experimental results (see Section 7.2.1 for detailed setup), the probability of having write event in each cycle is $Write_{rate} = 0.08$, and hence, probability of refresh for the entire content of L1 can be extracted as follows:

$$Refresh_{rate} = Write_{rate} * Word_{rate} \tag{7.1}$$

which is equal to 1 each 250 ms, namely *Actual Preserved Period.* In order to evaluate the corresponding retention failure rate, we use a binomial distribution modeled in Equation 5.2. The results show that the retention failure rate ranges between 10^{-5} to 10^{-3} which is significantly high and thus un-acceptable. Whereas, once an error code with ability to detect 3 errors is employed , the retention failure rate sharply decreases to 10^{-16}, as shown in Figure 7.2. Our adopted SEDC is able to detect up to 70 errors per cache line of size 512 bits, which means 9 errors per 64-bit word, in which 6 of them are

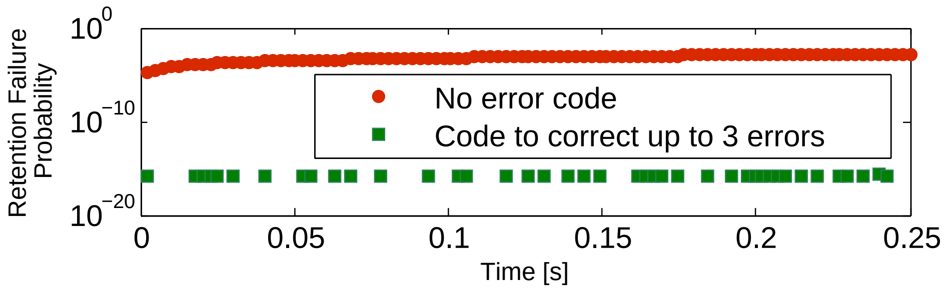


Figure 7.2: Retention failure versus retention time for 64-bit data [$\Delta = 30$, $I_w/I_c = 3$, see Section 7.2.1 for setup]

Table 7.2: raw WER (before coding) for target WER of 10^{-18}

	raw WER	Write latency [ns]
No-ECC	10^{-18}	11.6
ECC1	10^{-9}	6.5
ECC2	10^{-6}	4.9
ECC3	10^{-4}	4.1
ECC4	10^{-3}	3.6
SEDC	$\mathbf{10^{-1}}$	**2.2**

reserved for WER to reduce write latency, and the remaining targets retention failures that correlate with lowering Δ for fast caches. However, the proposed idea is based on reducing Δ as long as the read disturb is not an issue (i.e., $\Delta > 25$). Error detecting codes are considered in STT-MRAM in order to reduce write margins for a target WER (10^{-18} [66]). In fact, the raw WER, before detection or correction, is much higher ($>> 10^{-18}$). Table 7.2 lists the raw WER for various error codes with corresponding write latency for the target WER of 10^{-18}. However, to estimate the impact of the optimized write latency on the overall performance, we need to evaluate the impact of the raw WER and the imposed overhead of the data correction (data movements from L_{i+1} to L_i) as well. We use *Effective Write Latency*, which relies primarily on the raw WER, and is modeled as:

$$t'_w = t_w + WER * (t_w + t_e + t_{L2}) \tag{7.2}$$

where t_e is encoding penalty and t_{L2} is read access time of L2. Therefore, with high raw WER (e.g., 10^{-1} such as with SEDC), the effective write latency will dramatically increase, which impairs the overall performance and increases the write energy. Figure 7.3 shows the system-level impact of the effective write latency and energy according to the adopted write latency. Based on this analysis, the optimal write latency is 2.5 ns for our setup (latency of 5 clock cycles for 2 GHz processor).

7.2 Simulation Results

This section demonstrates the impact of our approach applied on the STT-MRAM for L1 cache of a high-performance processor in a detailed comparison with the conventional

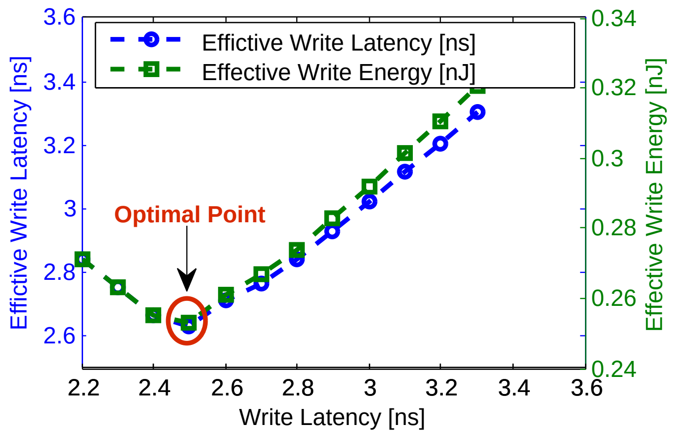


Figure 7.3: Adopted write latency versus effective write latency for 64-bit data according to WER [$\Delta = 30$, $I_w/I_c = 3$, see Section 7.2.1 for setup]

Table 7.3: Simulation setup in gem5

Processor	Single-core, 2 GHz, Out-of-order, 4-issue
L1-cache (Data and Instruction)	16/16 KB, 4-way set associative, 64B line size STT-MRAM, $\Delta = 30$, $I = 3$ different read/write latencies Table 7.4
L2-cache	512 KB, 8-way set associative, 64B line size STT-MRAM, $\Delta = 35$, $I = 2$ read/write latencies as 1.12ns/14.00ns
SPEC Applications	gzip, bzip2, mcf, twolf, vpr

Table 7.4: Read and write latencies of 64-bit STT-MRAM memory with 16 KB capacity

		No-ECC	ECC1 SECDED	ECC2 BCH	ECC3 BCH	ECC4 BCH	**SEDC**
Write [ns]	ECC Encoding	—	0.400	0.525	0.530	0.545	**0.600**
	Memory Write	11.610	6.456	4.909	4.100	3.628	**2.500**
	Overall	11.610	6.856	5.434	4.630	4.173	**3.100**
Read [ns]	ECC Decoding	—	0.580	2.459	3.698	4.714	**0.700**
	Memory Read	0.898	0.899	0.899	0.899	0.900	**0.898**
	Overall	0.898	1.479	3.358	4.597	5.614	**1.598**

approach of STT-MRAM and SRAM in terms of performance, energy consumption, and reliability.

7.2.1 Simulation Setup

We simulated five applications from the SPEC2000 [123] benchmark suite on this processor. For each one, the Simpoint tool [138] is used to find the representative fast-forward distances. Afterwards the simulation was conducted over 100 millions instructions. Table 7.3 summarizes the set up for our experiments.

Table 7.5: SEDC Versus ECCn

	Encoder		Decoder		Storage	Dynamic write
	Latency [Cycle]	Area [μm]	Latency [Cycle]	Area [μm]	overhead	energy ratio ECCn/SEDC
SEDC	2	3,875	2	4,088	1.5%	—
ECC1	1	2,862	2	3,456	12.5%	1.08X
ECC2	2	7,728	5	23,560	23.4%	1.22X
ECC3	2	13,356	8	36,382	34.3%	1.32X
ECC4	2	20,496	10	50,885	45.3%	1.43X

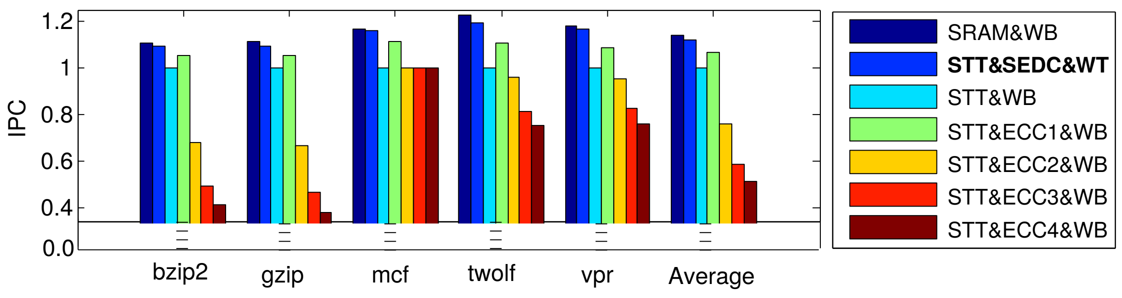


Figure 7.4: Performance of the proposed and conventional approaches for various SPEC workloads

7.2.2 SEDC Versus ECCn

To evaluate the efficacy of SEDC code, a detailed comparison with ECCn codes for word size of 64-bits synthesized with TSMC 65 nm standard cell library and for a 2 GHz processor is done in terms of area, encoding/decoding latencies, storage overhead and dynamic write energy. The results are shown in Table 7.5. It is obvious that the decoding latencies are quite high for ECCn codes (except n=1). This means that the utilization of ECC will adversely impact the read latency. Furthermore, ECC code not only increases the read latency, but also imposes a considerable storage overhead.

7.2.3 Performance Analysis

Figure 7.4 illustrates the *Instruction Per Cycle* (IPC) extracted for different implementations of L1 cache. As shown, the IPC obtained by ”SRAM&WB” is significantly high (on average 14%) compared to ”STT&WB” with no ECC capabilities. This is mainly due to the high write latency in STT-MRAM.

As shown, exploiting ECC to reduce the write margin to reach the required performance of fast caches is efficient only for ECC1. Afterwards, IPC dramatically decreases because of the impact of the added decoding penalty on read access latency (See Table 7.4), which is significantly increased by increasing the ECC robustness. This erodes the gain from write latency improvement. For instance, for ”STT&ECC4&WB”, the IPC can be as low as 55% compared to the standard ”SRAM&WB” implementation. However, with our proposed approach ”SEDC&WB” the performance gap between SRAM and STT-MRAM is reduced considerably, which becomes less than 2%. Table 7.6 summarizes the average IPC, read latency and write latency improvements of our approach over existing solutions.

Table 7.6: Improvements using our proposed approach "SEDC&WB" over existing solutions

	No-ECC	ECC1	ECC2	ECC3	ECC4
IPC improvement	12.1%	5.2%	47.8%	92.5%	120.8%
Read latency improvement	—	1X	2.62X	3.99X	5.12X
Write latency improvement	3.87 X	2.15 X	1.63 X	1.36 X	1.21 X

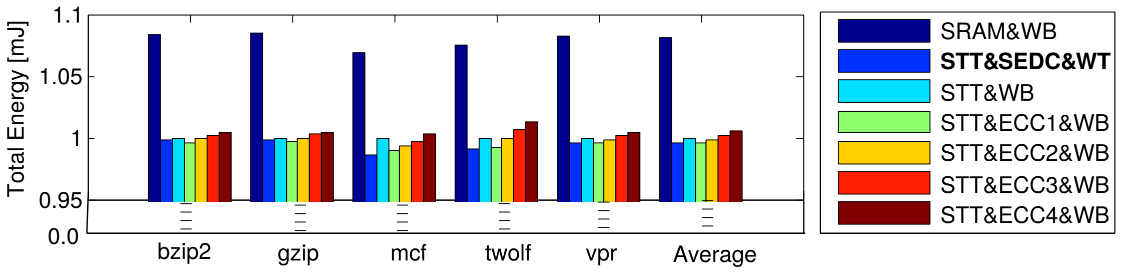


Figure 7.5: Total energy consumption (Dynamic+Leakage) for L1 and L2 of the proposed and conventional approaches for various SPEC workloads

7.2.4 Energy Analysis

In general, in L2 cache, the WT policy increases the number of write accesses, which consumes higher dynamic-energy in L2 than WB policy. However, using our approach "SEDC&WT", the write access latency goes down from 11.6 ns to 2.5 ns that in turn reduces the energy overhead of WT in L2 cache. This energy reduction in L2 is because of two reasons: i) the reduction of the dynamic-energy in L1 and ii) the total reduction of the leakage for both L1 and L2 because of the performance optimization. The total energy consumption for L1 and L2 caches for our proposed "STT&SEDC&WB" along with various other conventional approaches are shown in Figure 7.5. This figure shows that, for all workloads, our proposed approach is the most energy efficient scheme. We have observed that using our proposed approach "STT&SEDC&WT", the total system-level energy is reduced by 9% on average compared to the standard "SRAM&WB" scheme. On the top of that, it is also beneficial to the system-level leakage as STT-MRAM technology consumes significantly less leakage compared to SRAM. Please note, in STT-MRAM technology, only peripheral circuitry contributes to leakage.

7.2.5 Reliability Analysis

A comparison of various failure rates, as shown in Table 7.7, demonstrates the robustness of our approach "SEDC&WT". Because of the high error-detection capability of SEDC, the write error rate is orders of magnitude smaller than ECC schemes. Furthermore, "SEDC&WT" approach allows us to optimize the write latency and energy by adopting low thermal stability factor value ($\Delta = 30$), while the same Δ is not acceptable to be used with ECC schemes, as the retention failure rate is very far from target 10^{-16}. The reliability of our proposed approach makes it the only viable solution to use STT-MRAM for fast caches.

Table 7.7: Reliability Analysis of our approach over existing solutions

	SEDC	No-ECC	ECC1	ECC2	ECC3	ECC4
Write error rate (WER)	10^{-21}	10^{-18}				
Retention failure rate	10^{-16}	From 10^{-5} to 10^{-3}				
Read error rate	From 10^{-23} to 10^{-21} [55]					

7.3 Summary

In this chapter, we have presented a novel approach to use STT-MRAM in fast caches by exploiting the asymmetric nature of error occurrence. Using an asymmetric error detecting code together with the write-through cache policy, the high costs of using ECC in terms of decoding latency and storage overhead are minimized to a large extent. Furthermore, this allows us to excessively reduce the write latency while improving the reliability. As a result, the proposed approach improves the overall system performance and energy, on average, by up to 12% and 3%, respectively, and reduces write errors and retention failure to virtually zero. This make it a viable solution to replace SRAM even for fast caches.

Chapter 8

Process Variation and Temperature Aware Adaptive Scrubbing scheme

Retention failure has emerged as a major reliability concern for STT-MRAM technology due to the large variations in retention time because of process variations and temperature effects. The conventional solution to mitigate retention failures is to use scrubbing at regular intervals to prevent accumulation of errors, based on the worst case retention time of the memory array. However, this leads to very frequent scrubbing, resulting in large performance and energy overheads. On the other hand, fixing the scrubbing period based on the average case could lead to accumulation of errors beyond the error correction capability and hence result in failures of the memory system.

In this chapter, we develop a cross-layer framework for retention failure analysis in STT-MRAM and propose a process variation and temperature aware adaptive scrubbing scheme to mitigate these failures. In the proposed method, we group different cache lines based on their thermal stability (or retention times). We then use different scrubbing intervals for different groups based on the worst case retention time of each group. The scrubbing interval is dynamically adjusted to take into account the temperature effects as well. This means that the cache lines in some of the groups will be scrubbed more frequently, whereas those in some other groups will be scrubbed less frequently. The performance overhead improvement depends on the number of groups the cache lines are divided into. Our results show that for a group size of 4, the performance and energy overheads of scrubbing can be reduced by 97%, compared to the uniform scrubbing interval across the entire memory array.

8.1 Proposed Approach

In this chapter, we propose an adaptive multi-level scrubbing approach to prevent the data loss in STT-MRAM due to retention failures, which is highly affected by the process variation and the operating temperature. This technique exploits the non-uniformity in the retention time of STT-MRAM to reduce the scrubbing costs.

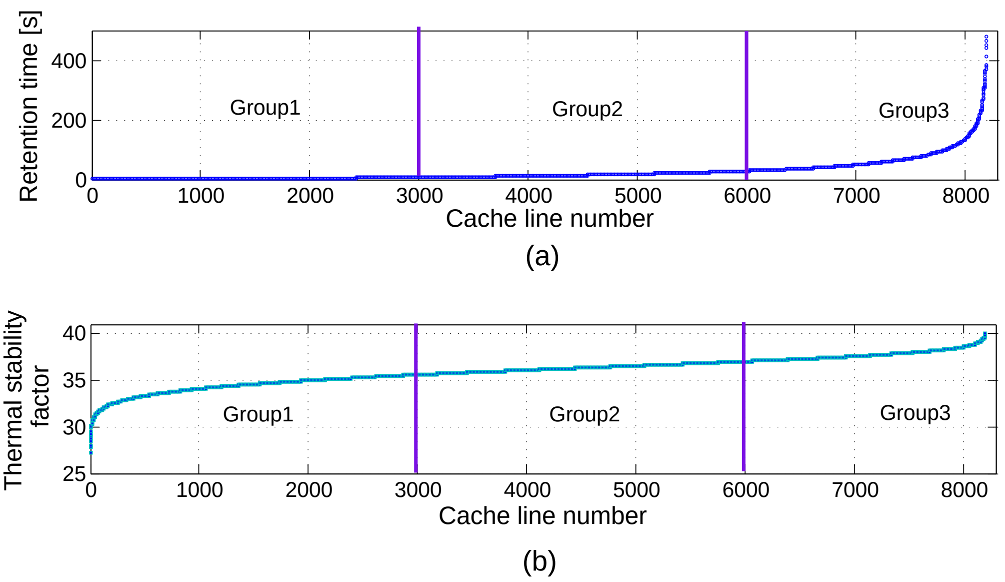
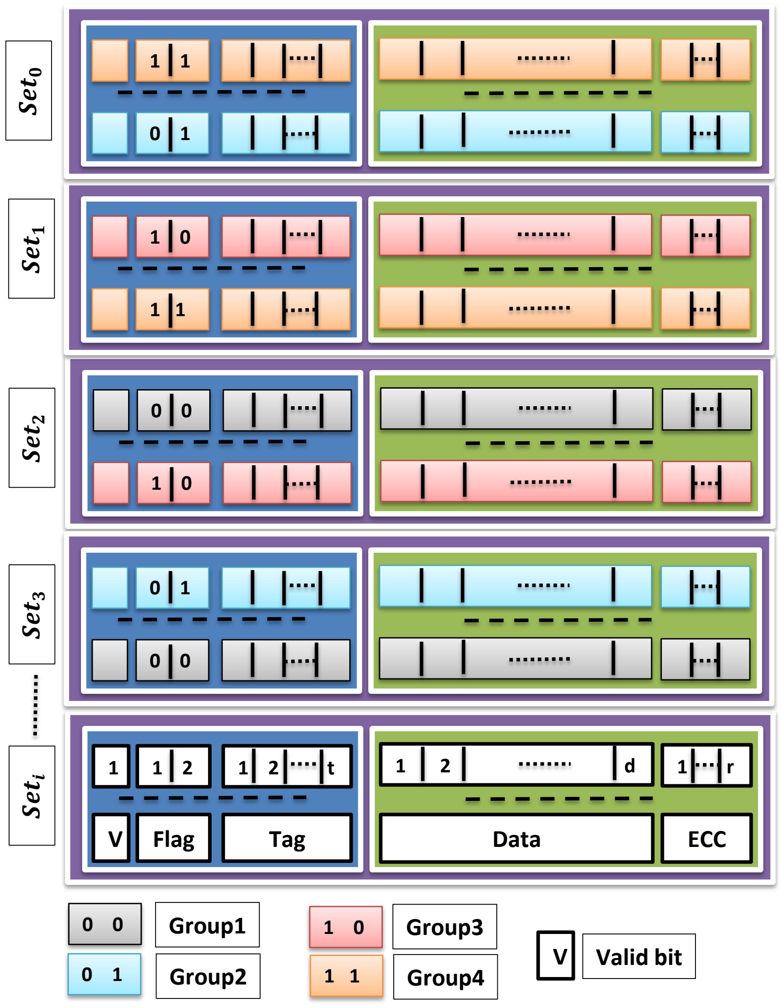


Figure 8.1: (a) Retention time variations; and, (b) Thermal stability factor variations per line with size of 64-Byte for 512 KB memory

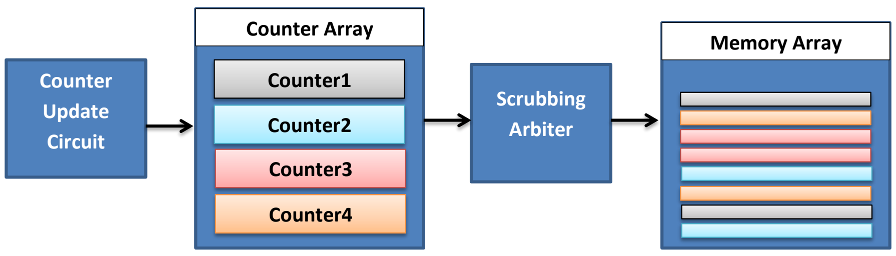
8.1.1 Methodology

As explained in Section 2.3.4, a single scrubbing interval based on the cell with the worst case retention time is highly inefficient due to the large performance and energy overheads. Hence, a multi-level scrubbing scheme is proposed. This method involves classifying the cache lines into different groups based on their thermal stability factors (or retention times) and using different scrubbing intervals for each of these groups. The scrubbing interval for each group is set based on the cell with the worst case retention time for that group. Figure 8.1 illustrates the variations of the thermal stability factor along with the variations of retention time per line. To obtain the retention times of various memory lines of STT-MRAM cache, any of the existing retention testing methods can be adopted [78, 131]. Moreover, the operating temperature may lead to large variations in the retention time. This means that the multi-level scrubbing interval should be adjusted based on the operating temperature so as to meet the target retention failure rates. Therefore, the system can be equipped with thermal sensors to obtain the operating temperature. Scrubbing intervals of groups can be tracked by assigning a counter to each group. The readouts from thermal sensor will initiate the event of updating counter values, based on the operating temperature. When the counters reach their lowest value (i.e., “0”), the scrubbing process is performed and the counters will be reset to the highest value (i.e., based on the minimum retention time of the cells in the group). The minimum retention time of groups based on the temperature can be simply saved in a lookup table. However, since the retention times of groups scale for different temperatures with the same scale, a shifting factor can be easily implemented to adjust the counters values.

The implementation of the proposed approach is divided into different steps. At design-time, the required number of groups along with their counters are specified. The lines belonging to a single group can be identified by adding the required number of flag bits to the tag array. Thus, accessing the line for the scrubbing process will be based on its flag. After manufacturing, retention testing determines bit-cell, line and memory retention times. Based on the retention testing information, classifying memory lines into groups is done using the standard k-means clustering algorithm [132]. At run time, adjusting the scrubbing intervals of different groups based on the operating temperature is performed by the counter update circuitry, as the counter values of each group are adjusted based on operating temperature intervals, using a simple look-up table. The scrubbing events of the groups are then activated based on the counter values using the scrubbing arbiter, which scans the group lines and writes back only erroneous bits. Figure 8.2 illustrates our scrubbing scheme.



(a) Memory groups



(b) Counter-controlled scheme

Figure 8.2: Demonstration of the proposed scrubbing approach at cache level

8.1.2 Grouping of Cache Lines based on Retention Time

The grouping of cache lines are done based on their thermal stability factors. Each group is defined as a collection of cache lines which exhibit very close retention capabilities that are easily distinguishable from those of the rest of the lines. In order to identify the group to which a particular line belongs, simple division of cache lines into groups of the same size will not help, as the thermal stability factor values are randomly distributed over the cells due to the process variation. Hence, it is better to partition the lines with similar retention times into the same group. Therefore, classifying the lines into groups should be done based on a classification technique from the field of machine learning such as K-mean clustering algorithm [132]. This algorithm can use the row-retention capability as the classifier metric to determine the optimal size of groups with their associated lines.

The multi-scrubbing scheme relies on determining the retention time of each bit-cell for identifying the retention time per line by testing the memory after manufacturing. The traditional approach to test the retention time of STT-MRAM technology is proposed in [78]. The retention time is determined by writing a small number of static patterns (such as “all 1s” or “all 0s”), turning off refreshes, and observing when the first bit changes [139]. There are several advanced methods to test the retention time of STT-MRAM (such as weak write test, at-burn-in and after-burn-in [131]). They are proposed to improve the test time by several orders of magnitude compared to the traditional STT-MRAM retention testing methods [78].

8.1.3 Data Residency in Cache

The on-chip memory hierarchy may consist of multiple levels of caches from the slow but high-capacity last level cache to a relatively fast but small second level cache, to an even smaller and faster first level cache. The data is stored in these levels for different retention times. In [91], it is observed that it is sufficient to store the data in the L2-cache for a few tens of ms, whereas a retention time of μs is acceptable for the L1-cache. Therefore, based on the retention testing results, if the retention time of a group is larger than the residency time of data in that cache level, no scrubbing for that group is required.

8.1.4 Hardware Implementation and Overheads

Scrubbing overheads such as the performance penalty and the energy consumption can be effectively reduced at the cost of stronger ECC or smaller fragmentation for more groups. However, overheads of ECC and scrubbing increase rapidly for stronger ECC and more groups, respectively. Table 8.1 reports the storage overhead due to the redundant bits of ECC with 1-error correcting capability (i.e., ECC1) up to ECC with 8-error correcting capability (i.e., ECC8) for a 64-Byte cache line. On the other hand, the cost associated with more groups appears in the increase in the required number of counters and the number of bits for the flag of each cache line, as shown in Table 8.1. Moreover, a set of bins is added to the memory controller, each is associated with one counter, each bin contains all of rows in the group which has to be scrubbed based on that counter, when it reaches the lowest value (“0”). At the system level, it is common to find

Table 8.1: ECC vs scrubbing overheads for 512 KB cache with 64-Byte row-size

ECC storage overhead						Scrubbing area overhead							
						Counter $\ll 0.0016\%$				Flag			
No-ECC	ECC1	ECC2	ECC3	ECC4	ECC8	1 group	2 groups	4 groups	8 groups	1 group	2 groups	4 groups	8 groups
0%	2%	4%	6%	8%	11%	16 bit	32 bit	48 bit	64 bit	0%	0.2%	0.4%	0.6%

Table 8.2: Simulation setup in gem5

Processor	Single-core, 2 GHZ, Out-of-order, 4-issue
L1-cache	32 KB, 4-way set associative, 64B line size SRAM
L2-cache	512 KB, 8-way set associative, 64B line size STT-MRAM, $\Delta = 40$ 1.6 ns/22 ns read/write latencies
SPEC2006 Applications	soplex, omnetpp, namd, libquantum, lbm hmmer, h264ref, GemsFDTD, gcc, bzip2

several thermal sensors inside modern chips, possibly one thermal sensor to monitor the operating temperature as in the Exynos 5 Octa (5422) processor [140]. The temperature values and their related retention time (i.e., the reset values of the counters) are saved in a lookup table.

8.2 Simulation Results

8.2.1 Simulation Setup and Implementation flow

In order to analyze the efficacy of the proposed idea, we assume that the radius of the MTJ has a normal distribution with a standard variation of 5% [141], and we consider the operating temperature variation stepped by $10°C$ in the range of $[-20°C, +120°C]$, in order to capture the impact of process variation and operating temperature based on Equation (2.8) and Equation (2.12). We evaluate the retention failure probability per bit based on the target FIT rate and the adopted ECC scheme. A typical FIT rate for on-chip memories can be around 10 (i.e., 10 failures in 1 billion hours) [78]. We used different ECC correction capabilities (up to 8-errors per single cache line). We implement K-mean clustering algorithm to classify the lines into groups. We then evaluate the required scrubbing intervals for all groups along with the temperature variation in Matlab. Table 8.2 summarizes the set up for our experiments.

8.2.2 Scrubbing Interval Analysis

In order to quantify the benefit of the proposed approach, we perform the scrubbing interval evaluations for different cache fragmentations of 2, 4 and 8 groups under nominal operating temperature of $+30°C$. Figure 8.3 compares size and retention capabilities of each group for dividing caches into 2, 4 and 8 groups. We note that the size of the group which has to be scrubbed most frequently is optimized by 38%, 75% and 84% for dividing cache into 2, 4 and 8 groups, respectively, compared to the conventional approach, where the entire cache is considered as one block.

Figure 8.4 illustrates the difference in the scrubbing intervals within a temperature range from $-20°C$ upto $+120°C$ between the proposed method and the baseline. Note that

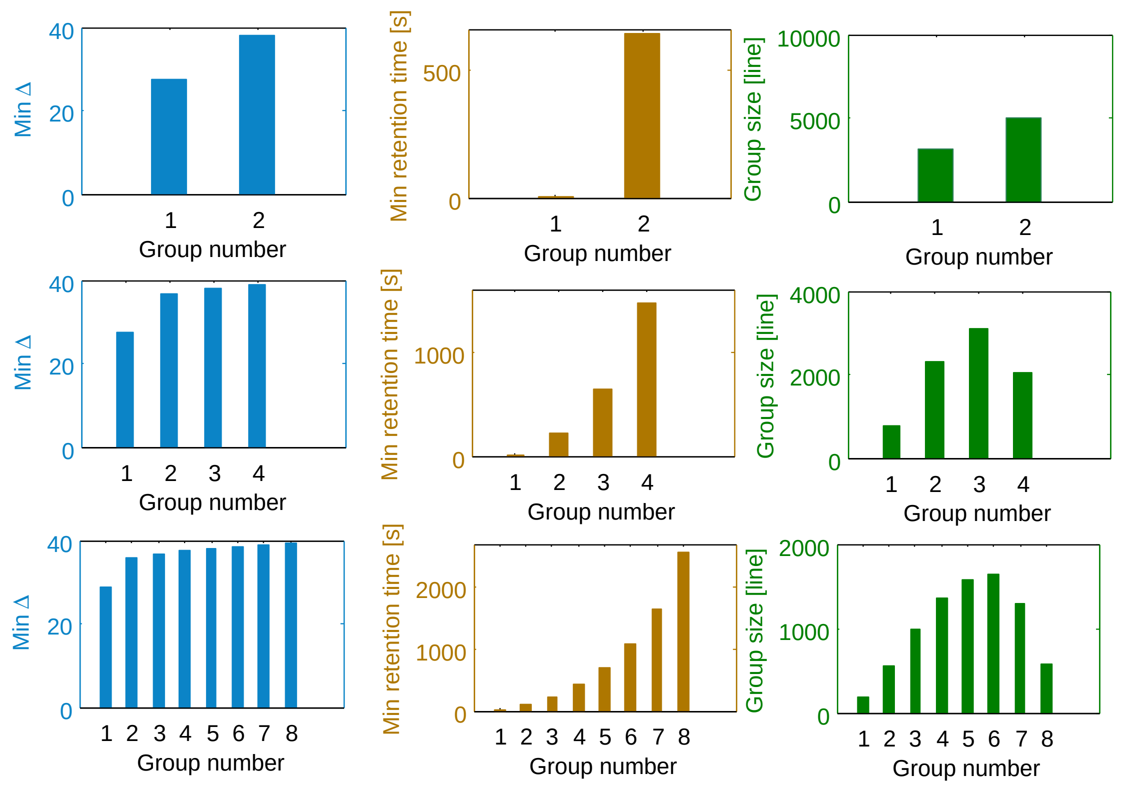


Figure 8.3: Retention capabilities, scrubbing intervals and size of groups for a 512 KB cache with line size of 64-Byte clustered into 2, 4 and 8 groups

the temperature-dependent retention times of the groups scale with the same ratio and without overlap. Increasing the temperature from $+30°C$ to $+120°C$ leads to an increase in the scrubbing frequency by 120× on average, for both baseline and the grouping schemes. On the other hand, for scaling down the operating temperature from $+30°C$ to $-20°C$, the scrubbing frequency can be reduced upto 43× on average.

Regardless of the number of groups and the operating temperature, scrubbing intervals can be significantly optimized with stronger ECCs by up to 2.9×, 8.5× and 24.7× for correction of 2, 4, and 8 errors, respectively, per line size of 512 bits in comparison to using ECC with one bit error correction.

8.2.3 Performance Analysis

For the ECC process, checking the data for errors takes less than 0.5 ns which has to be performed for each line for each scrubbing access. However, correction and write back operations, which takes around 22.5 ns for L2-cache, happen rarely, since retention failure probability is very low. Therefore, the scrubbing performance penalty is mainly due to read access, where all the lines in the group are read sequentially, and the data is checked for errors. By using a buffer cache, the penalty for checking the data can be efficiently hidden by pipelining the read and checking processes. This makes read penalty of the scrubbing process, which is 1.6 ns per line for L2, the main portion of the performance overhead.

We have performed the scrubbing experiments on SPEC2006 benchmark for different ECC correction capabilities (up to 8-errors per 64-Byte, i.e. one bit correcting per Byte)

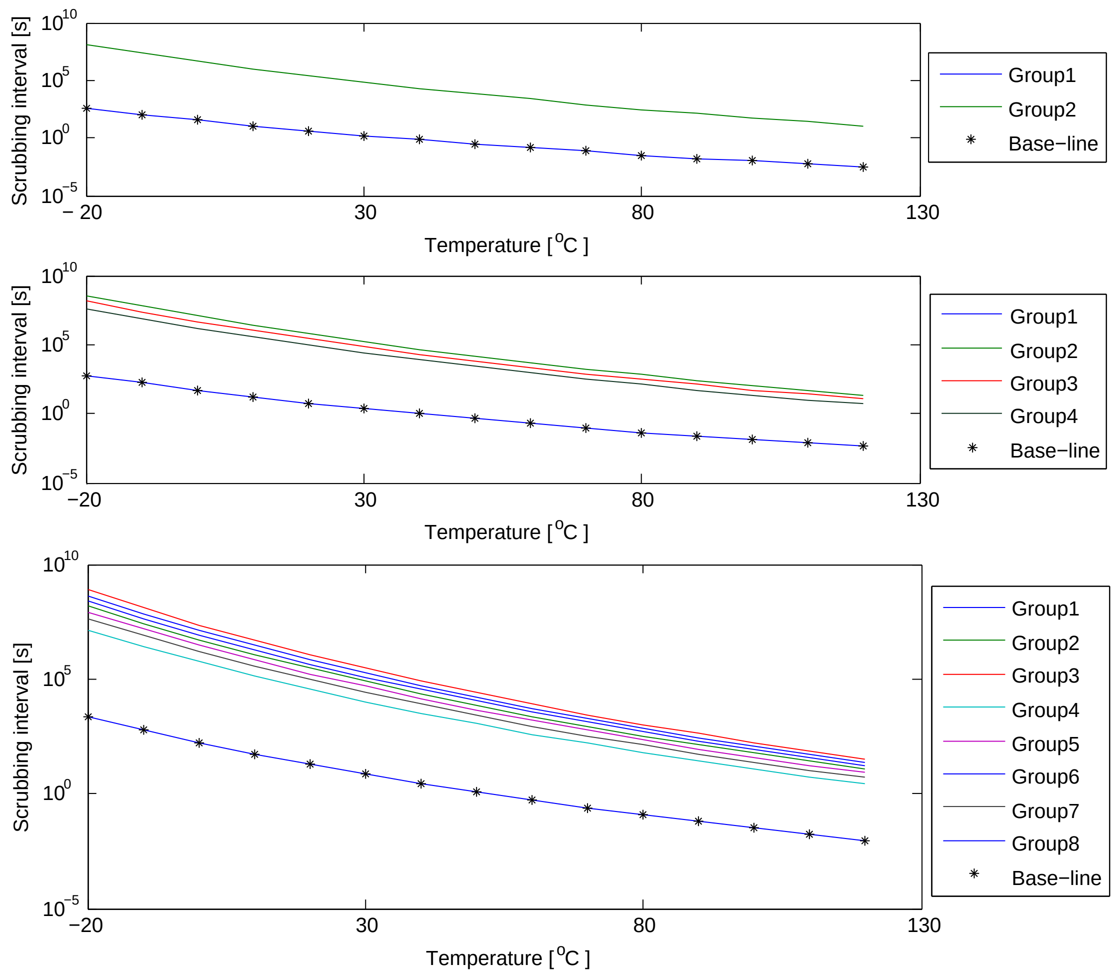


Figure 8.4: Scrubbing interval scaling trend with operating temperature scaling

and different number of groups. Figure 8.5 illustrates the scrubbing performance penalty of the proposed approach under different ECCs and grouping options compared to the baseline, which considers the entire cache as one group. The scrubbing performance overheads of the baseline and our approach are obtained by estimating the related L2-cache block time, which refers to the amount of time it takes to read the requested cache lines. We then evaluate the related processor pipeline-stall caused by blocking the L2-cache. This stall is translated into an increase in the read and write latencies of L2-caches, as shown in Table 8.3. The impact of this latency increase on the overall performance is then calculated by performing simulations in gem5.

It is seen that the scrubbing performance overhead is reduced by 60%, 97% and 99%, on average for 2, 4 and 8 groups, respectively, compared to the baseline, with zero storage overhead. On the other hand, for different ECC schemes, the reduction in the performance overhead compared to the baseline for 2 groups reach up to 60%, 97%, 98% and 99% for ECC1, ECC2, ECC4, and ECC8, respectively. However, the related storage overhead increases by 11% for ECC8, see Table 8.1.

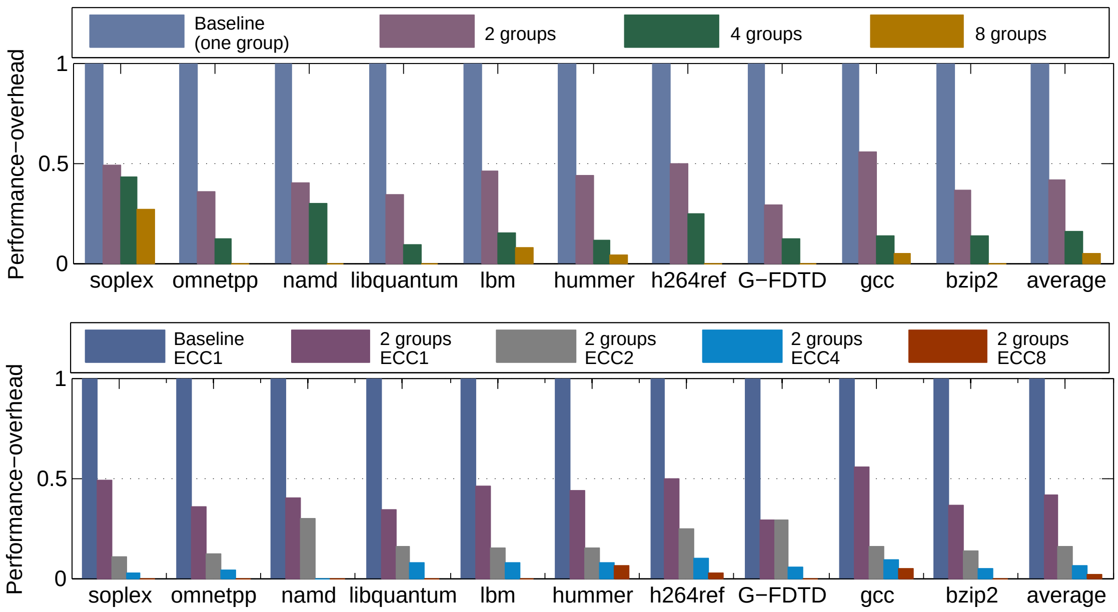


Figure 8.5: Performance overhead of scrubbing for the proposed and conventional scrubbing approaches, normalized to the overhead of conventional scrubbing, for various SPEC2006 workloads

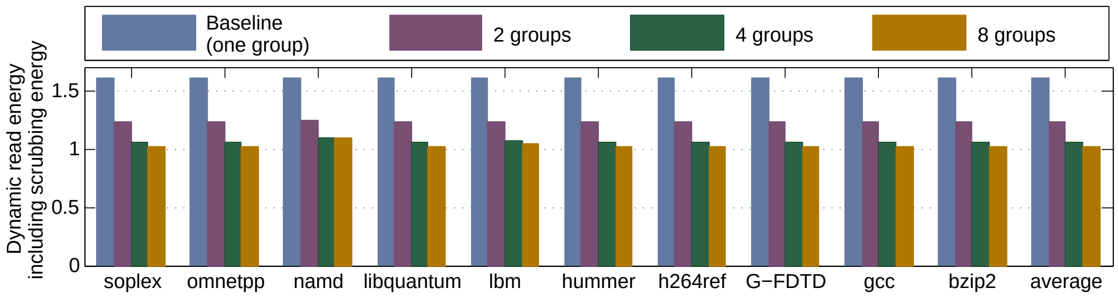


Figure 8.6: Dynamic read energy of the entire cache including scrubbing energy of the proposed and baseline approaches for various SPEC2006 workloads normalized to dynamic read energy without scrubbing

Table 8.3: Read/write latency increase based on the scrubbing block time

	Proposed approach					
Baseline	Two groups				Four groups	Eight groups
ECC1	ECC1	ECC2	ECC4	ECC4	ECC1	ECC1
60%	23%	8%	3%	0.8%	5.8%	0.8%

8.2.4 Energy Analysis

One of our major objectives is to reduce the energy overheads associated with scrubbing. This is achieved by replacing the uniform-scrubbing interval based on the worst case retention time with a multi-scrubbing interval, which is associated with the actual retention time of each line. Figure 8.6 shows a comparison of the total read energy (i.e., the energy of normal read processes and scrubbing read processes) consumed for the entire L2-cache for our proposed and the baseline approaches normalized to the read energy without any scrubbing. Note that to achieve the target error rate, the conventional scrubbing approach incurs more than 60% energy overhead. However, for 4 and 8 groups, the average increase in the total read energy is negligible and reaches up to 6% and 1%, respectively. For stronger ECC the related overhead of dynamic energy will increase because: i) scrubbing energy overhead does not change with different ECCs ii) dynamic energy overhead of the redundant bits increases up to 11% for ECC8 compared to no ECC.

It is worth noting that implementing the proposed idea for L1-cache will further reduce the scrubbing overheads by orders of magnitude compared to L2-cache results, as the target retention time decreases from a few tens of ms to a few μs, as discussed in Section 2.3.4.

8.3 Summary

In this chapter, we investigated an adaptive multi-level scrubbing scheme to reduce the overheads of the conventional scrubbing approach, to account for STT-MRAM retention failures due to process variation and operating temperatures. The simulation results reveal that this approach significantly reduces both performance and energy overheads associated with scrubbing up to 97%, while maintaining the same target error rate.

Chapter 9

Conclusions and Outlook

Energy consumption has emerged to be a major design constraint for modern integrated circuits, in particular in low-power application domains such as the "Internet of Things". The static power, which is the dominating component in the total energy consumption, is the leading roadblock in these fields [142]. Therefore, the stringent energy targets cannot be achieved using conventional SRAMs, because of the high leakage. Various techniques such as voltage scaling and power gating have been proposed as solutions for reducing the static power. However, with continuous scaling of CMOS technology, these techniques have become less efficient, primarily due to increased leakage power [17]. Semiconductor industry is actively searching for alternative memory technologies [143], including non-volatile memories which have zero leakage power for the bit-cell.

Among the emerging non-volatile memory technologies, *Spin Transfer Torque Magnetic Random Access Memory (STT-MRAM)* is one of the most promising technologies for future embedded memories. STT-MRAM is a novel and unique storage technology based on a magnetic device (called Magnetic Tunnel Junction or simply MTJ) that exploits not only the charge of electrons but also their spin to store digital content. As a result, STT-MRAM promises almost zero static power, fast access latencies (almost as fast as SRAM) and high integration density (as good as DRAM) [84, 144]. Thus, STT-MRAM has the potential to replace SRAM in the memory hierarchy of the computing system. However, in order to employ STT-MRAM as a promising low-power solution, several fundamental challenges still have to be resolved. In particular, it is of decisive importance to reduce the write energy as well as the write latency of STT-MRAM [142, 143].

9.1 Conclusions

In this dissertation, we provide design-time, compile-time and run-time solutions to address the challenges associated with STT-MRAM-based cache designs using a cross-layer approach. The contributions of this thesis improve the state-of-the-art by taking higher abstraction layer (i.e., application-level) requirements into consideration, which have often been neglected before. At application-level, the interdependencies among performance, energy, variability, reliability and temperature can be predicted and estimated with higher accuracy and lower complexity.

In the first part of this dissertation, STT-MRAM energy dissipation has been minimized under performance and reliability constraints. Multi-retention regions and multi-level write currents approaches have been adopted. Switching between these regions or levels relies mainly on the performance and reliability requirements, which in turn are driven directly from the application behavior. Therefore, static and dynamic approaches are proposed in order to predict and monitor the run-time application behavior (Chapter 3).

Moreover, for the energy reduction and performance optimization, relaxing the retention, write and read parameters has been discussed, resulting in run-time tradeoffs between reliability and energy/performance properties. However, the reduction in the target reliability has been successfully addressed at application-level by leveraging the approximate computing techniques (Chapter 4).

With respect to the power budget and reliability constraints, the performance has been dramatically optimized at system-level by either fast write termination or adaptive write current scaling technique at circuit-level. In fast write termination, the write margin has been reduced, where the unfinished bits are processed by adaptive error correcting code (Chapter 5). On the other hand, boosting the performance according to the application-level requirements is done by adjusting the write current levels on-the-fly (Chapter 6).

Finally, target reliability for our proposed energy-efficient or high-performance STT-MRAM design has been respected and met in two ways: i) the SEDC which is proposed to exploit the asymmetric switching nature of MTJ (Chapter 7), and ii) clustering the memory array cells based on the process variation to eliminate the overheads of conventional scrubbing technique (Chapter 8).

9.2 Outlook

The main focus in this dissertation is on optimizing the STT-MRAM performance in order to meet the on-chip requirements along with minimizing the energy dissipation due to write operation. This optimizations have been done under reliability constraints. However, our contributions can be extended in order to replace COMS technology with STT-MRAM not only in the memory system but also in the processor by designing normally-off/instant-on processor with non-volatile latches, flip-flops and registers. Furthermore, our contributions increase the potential of leveraging the STT-MRAM technology features for the future applications (e.g., IOT and artificial neural network applications) and future architecture (e.g.,in-memory computing).

Bibliography

[1] Gordon E Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[2] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[3] Intel.[Online], November 2016. Available: http://www.intel.com/content/www/us/en/silicon-innovations/ moores-law-technology.html, [accessed November 2016].

[4] S. Bob. The Origin, Nature, and Implications of "MOORE'S LAW", November 2016. Available: http://research.microsoft.com/en-us/um/people/gray/Moore-Law.html, [accessed November 2016].

[5] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[6] Karl Rupp. 40 Years of Microprocessor Trend Data, February 15th 2018. [Online]. Available: https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/.

[7] Annie Zeng, Kenneth Rose, and Ronald J Gutmann. Memory performance prediction for high-performance microprocessors at deep submicrometer technologies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(9):1705–1718, 2006.

[8] Kunle Olukotun and Lance Hammond. The future of microprocessors. *Queue*, 3(7):26–29, 2005.

[9] Wm A Wulf and Sally A McKee. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.

[10] Philip Machanick. Approaches to addressing the memory wall. *School of IT and Electrical Engineering, University of Queensland*, 2002.

[11] A Kagi, James R Goodman, and Doug Burger. Memory bandwidth limitations of future microprocessors. In *23rd Annual International Symposium on Computer Architecture (ISCA'96)*, pages 78–78. IEEE, 1996.

[12] Maurice V Wilkes. The memory gap and the future of high performance memories. *ACM SIGARCH Computer Architecture News*, 29(1):2–7, 2001.

[13] Li Zhao, Ravi Iyer, Srihari Makineni, Jaideep Moses, Ramesh Illikkal, and Donald Newell. Performance, area and bandwidth implications on large-scale cmp cache design. In *Proceedings of the Workshop on Chip Multiprocessor Memory Systems and Interconnect*, 2007.

[14] James R. Goodman. Using cache memory to reduce processor-memory traffic. *SIGARCH Comput. Archit. News*, 11(3):124–131, June 1983.

[15] N. P. Jouppi and S. J. E. Wilton. Tradeoffs in two-level on-chip caching. *SIGARCH Comput. Archit. News*, 22(2):34–45, April 1994.

[16] S. Przybylski, M. Horowitz, and J. Hennessy. Characteristics of performance-optimal multi-level cache hierarchies. In *The 16th Annual International Symposium on Computer Architecture*, pages 114–121, May 1989.

[17] Bernd Hoefflinger. Itrs: The international technology roadmap for semiconductors. In *Chips 2020*, pages 161–174. Springer, 2011.

[18] Robert Chau, Brian Doyle, Suman Datta, Jack Kavalieros, and Kevin Zhang. Integrated nanoelectronics for the future. *Nature materials*, 6(11):810, 2007.

[19] Massimo Alioto. *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems*. Springer, 2017.

[20] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[21] Shaoxiong Hua and Gang Qu. Approaching the maximum energy saving on embedded systems with multiple voltages. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 26. IEEE Computer Society, 2003.

[22] Takashi Nakada and Hiroshi Nakamura, editors. *Normally-Off Computing*. Springer, Tokyo, Springer, 2017.

[23] Kai Hwang and Naresh Jotwani. *Advanced Computer Architecture, 3e*. McGraw-Hill Education, 2011.

[24] Steven Przybylski, Mark Horowitz, and John Hennessy. Characteristics of performance-optimal multi-level cache hierarchies. In *The 16th Annual International Symposium on Computer Architecture*, pages 114–121. IEEE, 1989.

[25] Norman P Jouppi and Steven JE Wilton. Tradeoffs in two-level on-chip caching. In *ACM SIGARCH Computer Architecture News*, volume 22, pages 34–45. IEEE Computer Society Press, 1994.

[26] Shinobu Fujita, Kumiko Nomura, Hiroki Noguchi, Susumu Takeda, and Keiko Abe. Novel nonvolatile memory hierarchies to realize" normally-off mobile processors". In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 6–11. IEEE, 2014.

[27] Koji Ando, S Fujita, J Ito, S Yuasa, Y Suzuki, Y Nakatani, T Miyazaki, and H Yoda. Spin-transfer torque magnetoresistive random-access memory technologies for normally off computing. *Journal of Applied Physics*, 115(17):172607, 2014.

[28] Hiroshi Nakamura, Takashi Nakada, and Shinobu Miwa. Normally-off computing project: Challenges and opportunities. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 1–5. IEEE, 2014.

[29] Takayuki Kawahara. Scalable spin-transfer torque ram technology for normally-off computing. *IEEE Design & Test of Computers*, (1):52–63, 2010.

[30] Kumiko Nomura, Keiko Abe, Hiroaki Yoda, and Shinobu Fujita. Ultra low power processor using perpendicular-stt-mram/sram based hybrid cache toward next generation normally-off computers. *Journal of Applied Physics*, 111(7):07E330, 2012.

[31] Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, and Jude A. Rivers. Scalable high performance main memory system using phase-change memory technology. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 24–33, New York, NY, USA, 2009. ACM.

[32] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. Architecting phase change memory as a scalable dram alternative. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 2–13, New York, NY, USA, 2009. ACM.

[33] S. Lai. Current status of the phase change memory and its future. In *IEEE International Electron Devices Meeting 2003*, pages 10.1.1–10.1.4, Dec 2003.

[34] J Joshua Yang, Dmitri B Strukov, and Duncan R Stewart. Memristive devices for computing. *Nature nanotechnology*, 8(1):13, 2013.

[35] IG Baek, MS Lee, S Seo, MJ Lee, DH Seo, D-S Suh, JC Park, SO Park, HS Kim, IK Yoo, et al. Highly scalable nonvolatile resistive memory using simple binary oxide driven by asymmetric unipolar voltage pulses. In *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.*, pages 587–590. IEEE, 2004.

[36] Leon Chua. Resistance switching memories are memristors. *Applied Physics A*, 102(4):765–783, 2011.

[37] Ricardo C Sousa and I Lucian Prejbeanu. Non-volatile magnetic random access memories (mram). *Comptes Rendus Physique*, 6(9):1013–1021, 2005.

[38] S. Tehrani, J. M. Slaughter, E. Chen, M. Durlam, J. Shi, and M. DeHerren. Progress and outlook for mram technology. *IEEE Transactions on Magnetics*, 35(5):2814–2819, Sep. 1999.

[39] Fabian Oboril, Fazal Hameed, Rajendra Bishnoi, Ali Ahari, Helia Naeimi, and Mehdi Tahoori. Normally-off stt-mram cache with zero-byte compression for energy efficient last-level caches. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 236–241. ACM, 2016.

[40] Xiaochen Guo, Engin Ipek, and Tolga Soyata. Resistive computation: Avoiding the power wall with low-leakage, stt-mram based computing. *SIGARCH Comput. Archit. News*, 38(3):371–382, June 2010.

[41] Mitchelle Rasquinha, Dhruv Choudhary, Subho Chatterjee, Saibal Mukhopadhyay, and Sudhakar Yalamanchili. An energy efficient cache design using spin torque transfer (stt) ram. In *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, pages 389–394. ACM, 2010.

[42] Guillaume Prenat, Kotb Jabeur, Pierre Vanhauwaert, Gregory Di Pendina, Fabian Oboril, Rajendra Bishnoi, Mojtaba Ebrahimi, Nathalie Lamard, Olivier Boulle, Kevin Garello, et al. Ultra-fast and high-reliability sot-mram: From cache replacement to normally-off computing. *IEEE Transactions on Multi-Scale Computing Systems*, 2(1):49–60, 2016.

[43] Fabian Oboril, Rajendra Bishnoi, Mojtaba Ebrahimi, and Mehdi B Tahoori. Evaluation of hybrid memory technologies using sot-mram for on-chip cache hierarchy. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(3):367–380, 2015.

[44] R. Bishnoi, M. Ebrahimi, F. Oboril, and M. B. Tahoori. Architectural aspects in design and analysis of sot-based memories. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 700–707, Jan 2014.

[45] F. Oboril, R. Bishnoi, M. Ebrahimi, M. Tahoori, G. Di Pendina, K. Jabeur, and G. Prenat. Spin orbit torque memory for non-volatile microprocessor caches. 2016.

[46] Ping Chi, Shuangchen Li, Yuanqing Cheng, Yu Lu, Seung H Kang, and Yuan Xie. Architecture design with stt-ram: Opportunities and challenges. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 109–114. IEEE, 2016.

[47] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. A novel architecture of the 3d stacked mram l2 cache for cmps. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, pages 239–249, Feb 2009.

[48] Clinton W Smullen, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R Stan. Relaxing non-volatility for fast and energy-efficient stt-ram caches. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 50–61. IEEE, 2011.

[49] 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)*, pages 1313–1318, March 2012.

[50] Guangyu Sun, Yaojun Zhang, Yu Wang, and Yiran Chen. Improving energy efficiency of write-asymmetric memories by log style write. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '12, pages 173–178, New York, NY, USA, 2012. ACM.

[51] S. Motaman, S. Ghosh, and N. Rathi. Impact of process-variations in sttram and adaptive boosting for robustness. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1431–1436, March 2015.

[52] K. Munira, W. H. Butler, and A. W. Ghosh. A quasi-analytical model for energy-delay-reliability tradeoff studies during write operations in a perpendicular stt-ram cell. *IEEE Transactions on Electron Devices*, 59(8):2221–2226, Aug 2012.

[53] X. Fong, S. H. Choday, and K. Roy. Bit-cell level optimization for non-volatile memories using magnetic tunnel junctions and spin-transfer torque switching. *IEEE Transactions on Nanotechnology*, 11(1):172–181, Jan 2012.

[54] Weisheng Zhao and Guillaume Prenat. *Spintronics-based Computing.* Springer Publishing Company, Incorporated, 2015.

[55] Dmytro Apalkov, Alexey Khvalkovskiy, Steven Watts, Vladimir Nikitin, Xueti Tang, Daniel Lottis, Kiseok Moon, Xiao Luo, Eugene Chen, Adrian Ong, et al. Spin-transfer torque magnetic random access memory (stt-mram). *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 9(2):13, 2013.

[56] E. Chen, D. Apalkov, Z. Diao, A. Driskill-Smith, D. Druist, D. Lottis, V. Nikitin, X. Tang, S. Watts, S. Wang, S. A. Wolf, A. W. Ghosh, J. W. Lu, S. J. Poon, M. Stan, W. H. Butler, S. Gupta, C. K. A. Mewes, T. Mewes, and P. B. Visscher. Advances and future prospects of spin-transfer torque random access memory. *IEEE Transactions on Magnetics*, 46(6):1873–1878, June 2010.

[57] A. Raychowdhury. Pulsed read in spin transfer torque (stt) memory bitcell for lower read disturb. In *2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 34–35, July 2013.

[58] Zhitao Diao, Zhanjie Li, Shengyuang Wang, Yunfei Ding, Alex Panchula, Eugene Chen, Lien-Chang Wang, and Yiming Huai. Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory. *J. Phys.: Condens. Matter*, 19:165209–13, 04 2007.

[59] Nour Sayed, Mojtaba Ebrahimi, Rajendra Bishnoi, and Mehdi B Tahoori. Opportunistic write for fast and reliable stt-mram. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 554–559. European Design and Automation Association, 2017.

[60] Nour. Sayed and Rajendra Bishnoi and Mehdi B. Tahoori. Fast and reliable stt-mram using nonuniform and adaptive error detecting and correcting scheme. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 1–14, 2019.

[61] Nour Sayed, Fabian Oboril, Rajendra Bishnoi, and Mehdi B Tahoori. Leveraging systematic unidirectional error-detecting codes for fast stt-mram cache. In *2017 IEEE 35th VLSI Test Symposium (VTS)*, pages 1–6. IEEE, 2017.

[62] Nour Sayed, Rajendra Bishnoi, Fabian Oboril, and Mehdi B Tahoori. A cross-layer adaptive approach for performance and power optimization in stt-mram. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 791–796. IEEE, 2018.

[63] Nour Sayed, Fabian Oboril, Azadeh Shirvanian, Rajendra Bishnoi, and Mehdi B Tahoori. Exploiting stt-mram for approximate computing. In *2017 22nd IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2017.

[64] Nour. Sayed and Longfei Mao and Rajendra Bishnoi and Mehdi B. Tahoori. Compiler-Assisted and Profiling-Based Analysis for Fast and Efficient STT-MRAM On-Chip Cache Design. *ACM Transaction on Design Automation of Electronic Systems (TODAES)*, 2019.

[65] Nour Sayed, Sarath Mohanachandran Nair, Rajendra Bishnoi, and Mehdi B Tahoori. Process variation and temperature aware adaptive scrubbing for retention failures in stt-mram. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, pages 203–208. IEEE Press, 2018.

[66] D. Apalkov, A. Khvalkovskiy, S. Watts, V. Nikitin, X. Tang, D. Lottis, K. Moon, X. Luo, E. Chen, A. Ong, A. Driskill-Smith, and M. Krounbi. Spin-transfer Torque

Magnetic Random Access Memory (STT-MRAM). *J. Emerg. Technol. Comput. Syst.*, 9(2):1–35, May 2013.

[67] Jenny Preece, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland, and Tom Carey. *Human-Computer Interaction.* Addison-Wesley Longman Ltd., Essex, UK, UK, 1994.

[68] Mark D Hill and Alan Jay Smith. Evaluating associativity in cpu caches. *IEEE Transactions on Computers*, 38(12):1612–1630, 1989.

[69] Daniel Page. *A practical introduction to computer architecture.* Springer Science & Business Media, 2009.

[70] William Wesley Peterson, Wesley Peterson, EJ Weldon, and EJ Weldon. *Error-correcting codes.* MIT press, 1972.

[71] Todd K Moon. Error correction coding. *Mathematical Methods and Algorithms. Jhon Wiley and Son*, pages 2001–2006, 2005.

[72] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[73] Robert Chien. Cyclic decoding procedures for bose-chaudhuri-hocquenghem codes. *IEEE Transactions on information theory*, 10(4):357–363, 1964.

[74] James Massey. Step-by-step decoding of the bose-chaudhuri-hocquenghem codes. *IEEE Transactions on Information Theory*, 11(4):580–585, 1965.

[75] Der Jei Lin et al. Systematic unidirectional error-detecting codes. *IEEE Transactions on Computers*, 100(11):1026–1032, 1985.

[76] Wang Kang, Liuyang Zhang, Jacques-Olivier Klein, Youguang Zhang, Dafiné Ravelosona, and Weisheng Zhao. Reconfigurable codesign of stt-mram under process variations in deeply scaled technology. *IEEE Transactions on Electron Devices*, 62(6):1769–1777, 2015.

[77] Youngbin Jin, Mustafa Shihab, and Myoungsoo Jung. Area, power, and latency considerations of stt-mram to substitute for main memory. In *Proc. ISCA*, 2014.

[78] Helia Naeimi, Charles Augustine, Arijit Raychowdhury, Shih-Lien Lu, and James Tschanz. Sttram scaling and retention failure. *Intel Technology Journal*, 17(1), 2013.

[79] Xuanyao Fong, Sri Harsha Choday, and Kaushik Roy. Bit-cell level optimization for non-volatile memories using magnetic tunnel junctions and spin-transfer torque switching. *IEEE Transactions on Nanotechnology*, 11(1):172–181, 2012.

[80] Elena I Vatajelu, Rosa Rodríguez-Montañés, Stefano Di Carlo, Marco Indaco, Michel Renovell, Paolo Prinetto, and Joan Figueras. Power-aware voltage tuning for stt-mram reliability. In *2015 20th IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2015.

[81] Wang Kang, Zheng Li, Jacques-Olivier Klein, Yuanqing Chen, Youguang Zhang, Dafiné Ravelosona, Claude Chappert, and Weisheng Zhao. Variation-tolerant and disturbance-free sensing circuit for deep nanometer stt-mram. *IEEE Transactions on Nanotechnology*, 13(6):1088–1092, 2014.

[82] Liuyang Zhang, Yuanqing Cheng, Wang Kang, Lionel Torres, Youguang Zhang, Aida Todri-Sanial, and Weisheng Zhao. Addressing the thermal issues of stt-mram from compact modeling to design techniques. *IEEE Transactions on Nanotechnology*, 17(2):345–352, 2018.

[83] Kamaram Munira, William H Butler, and Avik W Ghosh. A quasi-analytical model for energy-delay-reliability tradeoff studies during write operations in a perpendicular stt-ram cell. *IEEE Transactions on Electron Devices*, 59(8):2221–2226, 2012.

[84] A Driskill-Smith, D Apalkov, V Nikitin, X Tang, S Watts, D Lottis, K Moon, A Khvalkovskiy, R Kawakami, X Luo, et al. Latest advances and roadmap for in-plane and perpendicular stt-ram. In *2011 3rd IEEE International Memory Workshop (IMW)*, pages 1–3. IEEE, 2011.

[85] Junwhan Ahn, Sungjoo Yoo, and Kiyoung Choi. Dasca: Dead write prediction assisted stt-ram cache architecture. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 25–36. IEEE, 2014.

[86] Kon-Woo Kwon, Sri Harsha Choday, Yusung Kim, and Kaushik Roy. Aware (asymmetric write architecture with redundant blocks): A high write speed stt-mram cache architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(4):712–720, 2014.

[87] Tianhao Zheng, Jaeyoung Park, Michael Orshansky, and Mattan Erez. Variable-energy write stt-ram architecture with bit-wise write-completion monitoring. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*, pages 229–234. IEEE Press, 2013.

[88] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. Energy reduction for STT-RAM using early write termination. In *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pages 264–268, Nov 2009.

[89] R. Bishnoi, F. Oboril, M. Ebrahimi, and M. B. Tahoori. Avoiding unnecessary write operations in STT-MRAM for low power implementation. In *Fifteenth International Symposium on Quality Electronic Design*, pages 548–553, Mar 2014.

[90] Zhenyu Sun, Xiuyuan Bi, Hai Helen Li, Weng-Fai Wong, Zhong-Liang Ong, Xiaochun Zhu, and Wenqing Wu. Multi retention level stt-ram cache designs with a dynamic refresh scheme. In *proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*, pages 329–338. ACM, 2011.

[91] Adwait Jog, Asit K Mishra, Cong Xu, Yuan Xie, Vijaykrishnan Narayanan, Ravishankar Iyer, and Chita R Das. Cache revive: architecting volatile stt-ram caches for enhanced performance in cmps. In *DAC Design Automation Conference 2012*, pages 243–252. IEEE, 2012.

[92] Daisuke Suzuki, Masanori Natsui, Akira Mochizuki, and Takahiro Hanyu. Cost-efficient self-terminated write driver for spin-transfer-torque ram and logic. *IEEE Transactions on Magnetics*, 50(11):1–4, 2014.

[93] Rajendra Bishnoi, Fabian Oboril, Mojtaba Ebrahimi, and Mehdi B Tahoori. Self-timed read and write operations in stt-mram. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(5):1783–1793, 2016.

[94] C. H. Kim B. Del Bel, J. Kim and S. S. Sapatnekar. Improving STT-MRAM Density Through Multibit Error Correction. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, Mar 2014.

[95] M. S. Golanbari, N. Sayed, M. Ebrahimi, M. H. M. Esfahany, S. Kiamehr, and M. B. Tahoori. Aging-aware coding scheme for memory arrays. In *2017 22nd IEEE European Test Symposium (ETS)*, pages 1–6, May 2017.

[96] X. Bi, Z. Sun, H. Li, and W. Wu. Probabilistic Design Methodology to Improve Run-time Stability and Performance of STT-RAM Caches. In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 88–94, Nov 2012.

[97] 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*, pages 125–130, Oct 2012.

[98] D. Strukov. The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories. In *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, pages 1183–1187, Oct 2006.

[99] J. Lucas, et al. Sparkk: Quality-scalable approximate storage in DRAM. In *The Memory Forum*, pages 1–9, 2014.

[100] S. Liu, et al. Flikker: saving DRAM refresh-power through critical data partitioning. *ACM SIGPLAN Notices*, 47(4):213–224, 2012.

[101] A. Sampson, et al. Approximate storage in solid-state memories. *TOCS*, 32(3):9, 2014.

[102] A. Ranjan, S. Venkataramani, X. Fong, K. Roy, and A. Raghunathan. Approximate storage for energy efficient spintronic memories. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015.

[103] Xiaoxia Wu, Jian Li, Lixin Zhang, E. Speight, and Yuan Xie. Power and performance of read-write aware hybrid caches with non-volatile memories. In *2009 Design, Automation Test in Europe Conference Exhibition*, pages 737–742, April 2009.

[104] J. Li, C. J. Xue, and Yinlong Xu. Stt-ram based energy-efficiency hybrid cache for cmps. In *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, pages 31–36, Oct 2011.

[105] Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, and Yuan Xie. Hybrid cache architecture with disparate memory technologies. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 34–45, New York, NY, USA, 2009. ACM.

[106] J. Hu, C. J. Xue, W. C. Tseng, Y. He, M. Qiu, and E. H. M. Sha. Reducing write activities on non-volatile memories in embedded cmps via data migration and recomputation. In *Design Automation Conference*, pages 350–355, June 2010.

[107] Q. Li, J. Li, L. Shi, M. Zhao, C. J. Xue, and Y. He. Compiler-assisted stt-ram-based hybrid cache for energy efficient embedded systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(8):1829–1840, Aug 2014.

[108] Amin Jadidi, Mohammad Arjomand, and Hamid Sarbazi-Azad. High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement. In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, ISLPED '11, pages 79–84, Piscataway, NJ, USA, 2011. IEEE Press.

[109] Yu-Ting Chen, Jason Cong, Hui Huang, Chunyue Liu, Raghu Prabhakar, and Glenn Reinman. Static and dynamic co-optimizations for blocks mapping in hybrid caches. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '12, pages 237–242, New York, NY, USA, 2012. ACM.

[110] Gabriel Rodríguez, Juan Touriño, and Mahmut T. Kandemir. Volatile stt-ram scratchpad design and data allocation for low energy. *ACM Trans. Archit. Code Optim.*, 11(4):38:1–38:26, December 2014.

[111] Ren-Shuo Liu, Chia-Lin Yang, and Wei Wu. Optimizing nand flash-based ssds via retention relaxation. pages 11–11, 02 2012.

[112] L. Shi, K. Wu, M. Zhao, C. J. Xue, D. Liu, and E. H. . Sha. Retention trimming for lifetime improvement of flash memory storage systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):58–71, Jan 2016.

[113] M. K. Qureshi, D. Kim, S. Khan, P. J. Nair, and O. Mutlu. Avatar: A variable-retention-time (vrt) aware refresh for dram systems. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 427–437, June 2015.

[114] C. Lin, D. Shen, Y. Chen, C. Yang, and M. Wang. Secret: Selective error correction for refresh energy reduction in drams. In *2012 IEEE 30th International Conference on Computer Design (ICCD)*, pages 67–74, Sep. 2012.

[115] Prashant J. Nair, Dae-Hyun Kim, and Moinuddin K. Qureshi. Archshield: Architectural framework for assisting dram scaling by tolerating high error rates. *SIGARCH Comput. Archit. News*, 41(3):72–83, June 2013.

[116] Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu. Raidr: Retention-aware intelligent dram refresh. In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society.

[117] Anurag Nigam, Clinton W. Smullen, IV, Vidyabhushan Mohan, Eugene Chen, Sudhanva Gurumurthi, and Mircea R. Stan. Delivering on the promise of universal memory for spin-transfer torque ram (stt-ram). In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, ISLPED '11, pages 121–126, Piscataway, NJ, USA, 2011. IEEE Press.

[118] N. Binkert, B. Beckmann andbose G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The Gem5 Simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug 2011.

[119] Chris Lattner and Vikram Adve. Llvm: A compilation framework for lifelong program analysis & transformation. In *Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization*, CGO '04, pages 75–, Washington, DC, USA, 2004. IEEE Computer Society.

[120] A. Mejdoubi, G. Prenat, and B. Dieny. A compact model of precessional spin-transfer switching for mtj with a perpendicular polarizer. In *2012 28th International Conference on Microelectronics Proceedings*, pages 225–228, May 2012.

[121] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, July 2012.

[122] Susan L. Graham, Peter B. Kessler, and Marshall K. Mckusick. Gprof: A call graph execution profiler. In *Proceedings of the 1982 SIGPLAN Symposium on Compiler Construction*, SIGPLAN '82, pages 120–126, New York, NY, USA, 1982. ACM.

[123] John L Henning. Spec cpu2000: Measuring cpu performance in the new millennium. *Computer*, 33(7):28–35, 2000.

[124] Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, and Yuan Xie. Power and performance of read-write aware hybrid caches with non-volatile memories. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 737–742. European Design and Automation Association, 2009.

[125] Johnny KF Lee and Alan Jay Smith. Branch prediction strategies and branch target buffer design. *Computer*, (1):6–22, 1984.

[126] Tse-Yu Yeh and Yale N Patt. Two-level adaptive training branch predict ion. 1991.

[127] M. R. Guthaus, et al. MiBench: A free, commercially representative embedded benchmark suite. In *WWC-4. 2001 International Workshop on*, pages 3–14.

[128] 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 *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, Nov 2013.

[129] Hanho Lee. High-speed vlsi architecture for parallel reed-solomon decoder. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(2):288–294, April 2003.

[130] S. M. Nair, R. Bishnoi, and M. B. Tahoori. Parametric failure modeling and yield analysis for STT-MRAM. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 265–268, Mar 2018.

[131] Anirudh Iyengar, Swaroop Ghosh, and Srikant Srinivasan. Retention testing methodology for sttram. *IEEE Design & Test*, 33(5):7–15, 2016.

[132] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Dover Books on Mathematics Series. Dover Publications, 2001.

[133] M. Nicolaidis, T. Bonnoit, and N. E. Zergainoh. Eliminating speed penalty in ECC protected memories. In *2011 Design, Automation Test in Europe*, pages 1–6, Mar 2011.

[134] Clinton Wills Smullen. *Designing giga-scale memory systems with STT-RAM*. University of Virginia, 2011.

[135] Elwyn R Berlekamp. *Algebraic coding theory*, volume 129. McGraw-Hill New York, 1968.

[136] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*. Elsevier, 1977.

[137] S.N. Trika, J.I. Garney, and M.K. Eschmann. Merging write-back and write-through cache policies, June 12 2007. US Patent 7,231,497.

[138] Rakesh Kumar, Dean M. Tullsen, Parthasarathy Ranganathan, Norman P. Jouppi, and Keith I. Farkas. Single-isa heterogeneous multi-core architectures for multi-threaded workload performance. *SIGARCH Comput. Archit. News*, 32(2):64–, March 2004.

[139] Ravi K Venkatesan, Stephen Herr, and Eric Rotenberg. Retention-aware placement in dram (rapid): Software methods for quasi-non-volatile dram. In *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, pages 155–165. IEEE, 2006.

[140] *Samsung Electronics*.

[141] Sarath Mohanachandran Nair, Rajendra Bishnoi, Mohammad Saber Golanbari, Fabian Oboril, and Mehdi B Tahoori. Vaet-stt: A variation aware estimator tool for stt-mram based memories. In *Proceedings of the Conference on Design, Automation & Test in Europe*, pages 1460–1465. European Design and Automation Association, 2017.

[142] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75, Dec 2003.

[143] International Technology Roadmap for Semiconductors. http://www.itrs.net, 2013.

[144] KL Wang, JG Alzate, and P Khalili Amiri. Low-power non-volatile spintronic memory: Stt-ram and beyond. *Journal of Physics D: Applied Physics*, 46(7):074003, 2013.

List of Figures

List of Tables

Glossary

Symbols | A | B | C | D | E | F | G | H | I | L | M | N | P | R | S | T | V | W

Symbols

C Technology Dependent Parameter.

GER_{th} Group Error Rate.

$GF(2)$ Binary Field.

$GF(2^n)$ Vector space of Error Code.

H_k Effective Anisotropy Field.

I Write current (I_w) to the critical current (I_c) ratio.

I_c Critical Current.

$I_k = (k \times k)$ Identity Matrix.

I_r Read Current.

I_w Write Current.

K_B Boltzmann Constant.

M_s Saturation Magnetization.

$P = (k \times (n - k))$ Parity Matrix.

P_{RD} Read Disturb Probability.

P_{RF} Retention Failure Probability.

$P_{fail-line}$ failure probability of a array line.

T Temperature in Kelvin.

TER_{th} Threshold Total Error Rate.

T_{Lazy} Lazy-ECC Penalty.

WER Write Error Rate.

$WP_{word}(t_w)$ Word Write Probability.

Δ Thermal Stability Factor type.

α Damping Constant.

η STT-MRAM Efficiency Parameter.

τ Intrinsic Attempt Time Constant of to 1 ns.

h Planks Constant.

p_{hit} Cache Hit Probability.

r MTJ Cell Radius.

t Retention Time.

t_c Cache Access Time.

t_c Main Memory Access Time.

t_D Decoding Latency.

t_E Encoding Latency.

t_{avg} Average Access Time of Memory.

t_r Read Time of STT-MRAM Bit-Cell Content.

t_w Write Time of STT-MRAM Bit-Cell Content.

A

AC Approximate Computing.

AP Anti-parallel Magnetic Orientation of MTJ Ferromagnetic Layers.

B

BCH Bose-Chaudhuri-Hocquenhgem.

C

C Error Code.

c Coded Messages.

c' Corrupted Message.

CMOS Complementary Metal-Oxide Semiconductor.

D

DRAM Dynamic Random Access Memories.

DVFS Dynamic Voltage and Frequency Scaling.

E

e Error Vector.

ECC Error Correcting Code which its decoding and encoding penalties are added to the read and write pulses, respectively.

F

f(x) Minimal Polynomial.

FIR Fault Injection Rate.

FIT Failures In Time.

G

G Generator Matrix.

g(x) Generator Polynomial.

gem5 Multicore simulator.

Gprof Performance Analysis Tool.

H

h(x) Parity-Check Polynomial.

I

IC Integrated Circuit.

IOT Internet of Things.

IPC Instructions per Cycle.

L

L1 First Cache Level.

L2 Second Cache Level.

Lazy-ECC Lazy-Error Correcting Code which its penalty is excluded from the read/write access.

LLVM Low Level Virtual Machine that written in C++ and designed for compile-time optimization.

M

m Original Messages.

MOSFET Metal Oxide Semiconductor Field-Effect Transistor.

MRAM Magnetic Random Access Memory.

MTJ Magnetic Tunnel Junction.

MTTF Mean Time To Failure.

N

NVM Non-Volatile Memory.

NVSim Emerging Non-Volatile Memory Simulator used for performance, energy and area estimation.

P

P Parallel Magnetic Orientation of MTJ Ferromagnetic Layers.

PCM Phase-Change Memory.

PG Power Gating.

PV Process Variation.

R

r Redundant Bits.

Read Disturb Undesired switching of a cell content at read access.

Retention Failure Undesired switching of a cell content during retention time.

RR Read Rate.

RRAM Resistive Random Access Memories.

S

s Code Syndrome.

SECDED Single Error Correction Double Error Detection.

SEDC Systematic Unidirectional Error-Detecting Code.

SNR Signal to Noise Ratiog.

SP Stable Period Of Cache Write Behavior.

SPEC2000 Standard Performance Evaluation Corporation CPU benchmark 2000.

SPEC2006 Standard Performance Evaluation Corporation CPU benchmark 2006.

SPICE Simulation Program with Integrated Circuit, an open-source analog electronic circuit simulator used to test and predict circuit behavior.

SRAM Static Random Access Memories.

STT-MRAM Spin Transfer Torque Magnetic Random Access Memory.

SVM Semi-Volatile Memory.

T

t Hamming Distance.

TDDB Time Dependent Dielectric Breakdown.

TER Total Error Rate.

TP Tracing Period Of Cache Write Access.

V

V Free Layer Volume of MTJ cell.

W

WB Write Back Policy.

Write Error Undesired switching of a cell content at write access.

WT Write Through Policy.