HIGH-PERFORMANCE AND LOW-POWER MAGNETIC MATERIAL MEMORY BASED CACHE DESIGN

by

Zhenyu Sun

M.S. in Electrical Engineering,

Polytechnic Institute of New York University, 2011

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2013

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Zhenyu Sun

It was defended on

November 15, 2013

Committee members

Hai Li, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Yiran Chen, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Alex K. Jones, Ph.D., Associate Professor, Department of Electrical and Computer Engineering

Rami Melhem, Ph.D., Professor, Department of Computer Science

Yang Jun, Ph.D., Associate Professor, Department of Electrical and Computer Engineering

Dissertation Director: Hai Li, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

HIGH-PERFORMANCE AND LOW-POWER MAGNETIC MATERIAL MEMORY BASED CACHE DESIGN

Zhenyu Sun, PhD

University of Pittsburgh, 2013

Magnetic memory technologies are very promising candidates to be universal memory due to its good scalability, zero standby power and radiation hardness. Having a cell area much smaller than SRAM, magnetic memory can be used to construct much larger cache with the same die footprint, leading to significant improvement of overall system performance and power consumption especially in this multi-core era. However, magnetic memories have their own drawbacks such as slow write, read disturbance and scaling limitation, making its usage as caches challenging.

This dissertation comprehensively studied these two most popular magnetic memory technologies. Design exploration and optimization for the cache design from different design layers including the memory devices, peripheral circuit, memory array structure and micro-architecture are presented. By leveraging device features, two major micro-architectures - multi-retention cache hierarchy and process-variation-aware cache are presented to improve the write performance of STT-RAM. The enhancement in write performance results in the degradation of read operations, in terms of both speed and data reliability. This dissertation also presents an architecture to resolve STT-RAM read disturbance issue. Furthermore, the scaling of STT-RAM is hindered due to the required size of switching transistor. To break the cell area limitation of STT-RAM, racetrack memory is studied to achieve an even higher memory density and better performance and lower energy consumption. With dedicated elaboration, racetrack memory based cache design can achieve a significant area reduction and energy saving when compared to optimized STT-RAM.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to thank my academic and thesis advisor, Dr. Hai (Helen) Li, for her support, encouragement and guidance during the entire duration of my Ph.D study in the past years. She taught me a lot about my research and guided me through every step of my studies. I also appreciate my co-advisor, Dr. Yiran Chen for his guidance for my Ph. D dissertation. I also extend my sincere thanks to my committee members for their careful guidance on my Ph. D dissertation.

Special thanks are also given to my group members and colleagues Xiuyuan Bi, Yi-Chung Chen, Miao Hu. They are my best friends in University of Pittsburgh and Polytechnic Institute of New York University. Without their moral and technical supports and helps, this project would not have gone forward.

Special thanks to my family for their support, especially for my mother who is the person raised me up and enlightened me at the beginning of my life. I want to express my gratitude to my wife for her encouragement, support and understanding throughout the period of my studies.

1.0 INTRODUCTION

Increasing capacity and cell leakage have caused the standby power of SRAM on-chip caches to dominate the overall power consumption of the latest microprocessors. Many circuit design and architectural solutions, such as V_{DD} scaling [1], power-gating [2], and body-biasing [3], have been proposed to reduce the standby power of caches. However, these techniques are becoming less effective as technology scaling has caused the transistor's leakage current to increase exponentially. Researchers have been prompted to look into the alternatives of SRAM technology. One possibility is the embedded DRAM (eDRAM) which is denser than SRAM. Unfortunately, it suffers from serious process variation issues [4]. Another alternative technology is the embedded phase change memory (PCM) [5], a new nonvolatile memory that can achieve very high density. However, its slow access speed makes PCM unsuitable as a replacement for SRAM.

1.1 MAGNETIC MEMORY TECHNOLOGIES DEVELOPMENT

Development of magnetic storage media began with the HDD from 1956. But it has been widely used as hard drive instead of on-chip embedded memory due to its storage limitation. Until 2003, a 128Kbit MRAM chip was manufactured with 0.18 technology [6], the development of STT-RAM draw increasing attention. The second generation magnetic memory, the *spin-transfer torque RAM* (STT-RAM) receives even more attention because it offers almost all the desirable features: the fast (read) access speed of SRAM, the high integration density of DRAM, and the nonvolatility of Flash memory. The unique programming mechanism of STT-RAM – changing the MTJ resistance by passing a spin-polarized current [7] – ensures

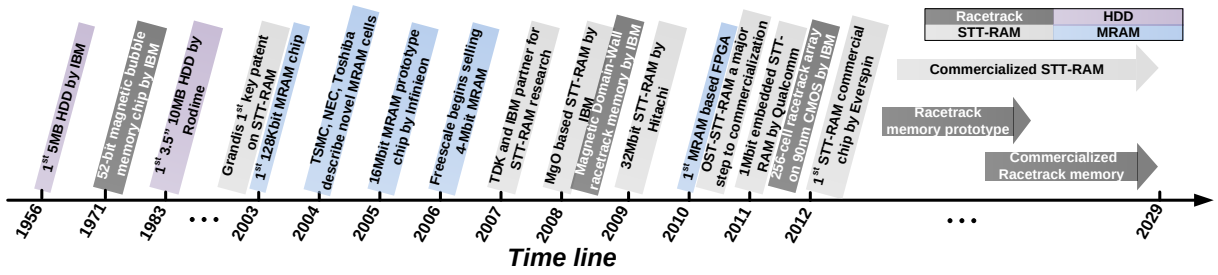


Figure 1: Development history of magnetic memory history.

fabrication feasibility down to the 22nm technology node [8]. Also, the compatibility with the CMOS fabrication process and similarities in the peripheral circuitries makes the STT-RAM an easy replacement for SRAM. Very recently, ***Everspin began shipping working samples of 64MB STT-RAM*** [9], announcing the commercialization era after many years of joint effort from both academia and industry [10][11][12].

To offer a "faster-than-Moore's law" scaling path, a team led by Dr. Parkin in IBM proposed *racetrack* memory that uses a spin-coherent electric current to move magnetic domains along a nanoscopic permalloy wire for data storage [13]. The racetrack memory is regarded as the third generation of MRAM. It inherits all the promising features of STT-RAM including fast access speed, nonvolatility, similar write/read mechanism, CMOS compatibility, hardness to soft error and even higher density. The cell area is expected to be as small as $2F^2$. Moreover, the continuous progress in device physics [14][15][16] and the recent successes in fabrication process [17][18][19] promise the feasibility of racetrack memory.

Table 1: Memory technologies comparison [ITRS2011].

	Retention	Cell size	Read time	Write time	Endurance	Dynamic pwr	Leakage pwr
SRAM	No	$50-200$ F^2	1 ns	1 ns	10^{16}	Low	High
PCM	Yes	$6-12$ F^2	$20-50$ ns	$50-120$ ns	10^{10}	High	None
MRAM	Yes	$16-40$ F^2	$3-20$ ns	$3-20$ ns	10^{15}	Medium	None
STT-RAM	Yes	$4-20$ F^2	$2-20$ ns	$2-20$ ns	10^{15}	Low	None

1.2 CONTRIBUTION OF THE DISSERTATION

STT-RAM is a very promising memory technology, but it is also subject to some inevitable challenges. The first challenge is the write performance including write speed and write energy. To address this issue, we introduce a novel cache hierarchy implemented entirely by STT-RAM with different nonvolatilities based on the fact that required data valid time in different part of the cache hierarchy differs significantly. With technology scaling, process variation, the second challenge, becomes more severe. Due to the process variation, the STT-RAM cells distributed on the whole die are supplied with non-uniform switching currents, resulting in different switching speeds. Most of the architecture explorations relevant to STT-RAM just consider single-corner scenario. We leverage such switching speed variation to further improve overall write performance of STT-RAM cache. With the write-ability enhanced, read performance including read delay and read disturbance issue of STT-RAM can no longer be neglected in STT-RAM design. In this dissertation, we study the impact of read performance of STT-RAM and evaluate possible architecture to overcome the read disturbance issue. To break the STT-RAM scaling limitation, another magnetic memory - racetrack memory is investigated in this dissertation. We exploit the potential of racetrack memory with a cross-layer design consideration. Solutions to realize ultra high density of racetrack memory are introduced. During design optimization, minimizing shift cost of racetrack memory is the main object. By leveraging the unique access feature of racetrack memory, we propose three architecture solutions, namely, *history based way reorder*, *split cache architecture* and *resizable cache* to minimize the shift cost of racetrack memory based cache design.

1.2.1 Improve write performance of STT-RAM

The major obstacles to use STT-RAM for on-chip caches are its longer write latency and higher write energy. During a STT-RAM write operation around 10ns region, the MTJ resistance switching mechanism is dominated by *spin precession*. The required switching current rises exponentially as the MTJ switching time is reduced. As a consequence, the

driving transistor's size must increase accordingly, leading to a larger memory cell area. The lifetime of memory cell also degrades exponentially as the voltage across the oxide barrier of the MTJ increases. As a result, a 10ns programming time is widely accepted as the performance limit of STT-RAM designs.

To improve the write performance, a detailed discussion on the tradeoff between the MTJ's write performance and its nonvolatility is presented in Chapter 3. In the same chapter, a multi retention level cache hierarchy implemented entirely with STT-RAM is proposed to deliver the optimal power saving and performance improvement based on the write access patterns at each level. This is the first time to use ultra-low retention STT-RAM as L1 cache which even can outperform SRAM in terms of both performance and power. In order to guarantee the data integrity of the ultra-low retention STT-RAM, a dynamic data refresh scheme is introduced. By simple leveraging the L1 cache assess characteristics, the dynamic data refresh scheme is much energy efficient than the DRAM-style periodic refresh in [20]. Inspired from previous hybrid SRAM/STT lower level, a mixed retention STT-RAM based lower level cache is invented to further maximize the energy/performance improvement and reduce design difficulty of mixing different memory technologies. The data migration is triggered by a queue based data profiling mechanism instead of previous counter based scheme. Considering observation time window, the proposed queue based profiling mechanism can classify data access pattern more efficient.

1.2.2 Leverage process variation of STT-RAM

Besides the slow and costly write operation, another challenge is introduced by the process variation. As technology scales down to 45nm or below, both CMOS and magnetic devices become more subject to process variations. In a STT-RAM cell, the current to switch the resistance state of its data storage element, MTJ, is determined by MTJ resistance and the NMOS selection transistor. When programming a STT-RAM cell in the sub-10ns region, the required MTJ switching time rises exponentially as the switching current reduces [21]. The process variations which result in a large distribution of the MTJ switching time that is in the same order of that of interconnect latencies and hence cannot be ignored.

Almost all of the architecture level work consider the single-corner device parameters. In other words, device variation is ignored in most previous architectural works. The device variation actually includes systematic and local components. The combination of the systematic and local components results in clustering of STT-RAM cells with similar switching performance. Also, most of the architecture related works have demonstrated uneven distribution of the cache accesses that can be also leveraged to mitigate the STT-RAM variation issue.

In Chapter 4, we thoroughly analyze the impact of process variations on STT-RAM key design parameters and demonstrate its spatial distribution by using multi-level qual-tree approach [22]. Inspired by NUCA algorithm [23], *process variation aware NUCA* (PVA-NUCA) techniques for large STT-RAM cache design are proposed, which include the non-uniformity of interconnect latencies and adaptively change programming duration. It is simple for DNUCA algorithm to just consider relation between the latency and memory bank location. But the STT-RAM cell variation information is not as explicit as the bank location which can be simply implied by the bank number. Post-silicon testing is required to obtain the cell variation information. And centralized storage is needed to reserve the variation information. Retrieving such information from the centralized storage is costly and inefficient. Therefore, corresponding solution is necessary. We introduced a hardware assistant scheme - sorting recording queue to resolve this issue by storing variation information of most recently accessed sets into a small queue. Such hardware assistant scheme is very efficient to minimize the access to the centralized storage. On the other hand, a conflict reduction scheme is used to minimize the data block competing issue among different processor cores to further benefit the PVA-NUCA.

Such process variation aware STT-RAM design is actually orthogonal to the multi retention cache design as well as other STT-RAM based cache design as long as there is STT-RAM with big capacity. For example, the process variation aware scheme can be used in the high retention region in the multi-retention STT-RAM cache hierarchy. It can also be used in both L2 and L3 cache of the single retention STT-RAM cache hierarchy in [20].

1.2.3 Emphasize read performance of STT-RAM

Besides the improvement of the write performance, the importance of read operation, which is not as trivial as what people previously thought of, has been brought out in the dissertation. The stored data of STT-RAM is read out by detecting the MTJ resistance state. The large process variations degrade the resistance differences between the data cell and the reference cell [24]. Consequently, the sensing delay of sense amplifier is enlarged. The situation in STT-RAM design with fast-switching devices is more severe: the improved write performance requires the read current amplitude to decrease accordingly in order to prevent the unintentional data switching, or read disturbance. The sub-nanosecond read speed under small read current and small read voltage margin that has been widely used is not true anymore.

Such read disturbance has been ignored by most of the STT-RAM designers and researchers for a while. It is really urgent to solve the read issue of STT-RAM from a different angle. There are three reasons to carry out related research. First, read operation happens much more frequent than write operation, so the overall system performance is more sensitive to read operation delay. Second, the sense amplifier, the major component of read circuit, requires very high design effort to make it more tolerable to process variation. Such accuracy requirement of sense amplifier will eventually hit bottleneck. And even more circuit design effort can not avoid read performance degradation. Third, read disturbance ratio is not trivial especially when large sensing current is required to assure large sensing margin and low sensing delay.

In Chapter 5, based on the comprehensive cross-layer (device-circuit-architecture) analysis, we first build the relationship amongst the read access latency, read disturbance probability, and the system performance. A novel STT-RAM based memory architecture is proposed, which switches between the *high accuracy* and the *low power* modes. Thus, the speed, energy, and data reliability of the overall computing system can be prioritized and balanced according to users' requirement. The high accuracy mode can guarantee zero read disturbance by rewriting each cache line after each read access, but it cost more energy. In order to minimize the energy consumption of *high accuracy* mode, hardware assistance tech-

niques are also proposed. Bit invert technology has been used in many area including bus energy reduction [25] and nonvolatile memory write energy reduction [26, 27]. More specific, [26, 27] tends to reduce energy due the asymmetric write energy of PCM and STT-RAM, respectively.

Based on the fact that RD errors happen unidirectionally [28] and the write energy is asymmetric, a bit invert scheme is combined with the proposed memory architecture to minimize the rewrite power cost. Moreover, sacrificing a little energy, a shadow rewrite buffer is used to minimize the performance degradation caused by rewrite under high accuracy mode.

1.2.4 Unveil the ractrack memory

However, restricted by the theoretical limit of $9F^2$, further shrinking memory cell size and hence improving performance and power consumption in STT-RAM is difficult [29]. Compared with SRAM and STT-RAM, the racetrack memory realizes the random data accesses by introducing an extra *racetrack shift* in a read/write operation. From memory array design perspective, the racetrack shifts enable the sharing of an access transistor by multiple memory bits and relieve the design constraint on the size and number of access transistors. The array density is determined only by the physical dimension of magnetic domains, which could be as small as $2F^2$. Because of the design rule of lithography, the $1F$ gap is required between two adjacent wires. So $1F^2$ as claimed in [30] and [31] can not be achieved. However, the extra delay and energy overheads induced by the shifts consistently apply to read and write operations and degrade the overall system performance. Therefore, minimizing the impact of racetrack shifts becomes the major concern in racetrack-based cache design.

Compared with array-style random access memory, integrating tape-style racetrack memory faces several unique design challenges: (1) To effectively utilize the stripe structure, new circuit layouts and optimizations are required distinct from array-based memories. (2) Stripe-based memory structures require new logical abstractions of memories. (3) Moving from random access (i.e., wordlines and bitlines) to sharing one access device (i.e., writing/reading requires shifting) requires careful design and scheduling of data access.

In Chapter 6, we comprehensively consider design requirements across different abstraction layers for racetrack memory. The main design goal is to construct a ultra-dense on-chip memory that enables high-performance and low-energy computation. First, we unveil the fact that the scaling trend will lead to enlarged pitch mismatch between racetrack memory cell and access transistor, causing space wasting above transistor layer. So, a design improvement for racetrack memory is initialized by reorganizing its physical layout which totally eliminates the access transistor area constraint. Second, thanks to the novel layout approach, a circuit structure improvement for racetrack memory array is proposed to support both read and write operation at each access port, avoiding long-distance racetrack shifting. Third, based on the physical structure, a racetrack memory architecture enables flexible physical-to-logic mapping, providing more design space at architectural level. Fourth, on the top of the proposed architecture, we leverage architectural solutions to further reduce the racetrack shifting operations during runtime.

Chapter 6 is an initial and general design exploration of racetrack memory. Chapter 7 aims at the deeper design exploration of racetrack last-level cache. First, we explore different physical layout strategies and array organizations of racetrack LLC compared with the conventional SRAM and the latest STT-RAM technologies, since the physical layout will have big effect on the architecture design. Based on the evaluations, we propose a two-step architecture optimization solution: (1) allocate array structures optimized for different types of cache requests at design time; and (2) adaptively adjust the racetrack usage upon workload's requirement during execution.

In racetrack memory, multiple storage elements correspond to one access transistor. Consequently, the cache performance greatly relies on the access port organization. A read-only port (R-port) realized with a minimum-size transistor can be shared by less memory bits, paying less shift overhead in each access. In contrast, a full-functional port supporting both read and write operations (R/W port) must be large enough to provide sufficient program current. It is shared by more magnetic domains, inducing higher shift overhead. All the possible layout strategies are studied and evaluated, as well as the impact of various access port organizations on racetrack LLC. The racetrack evaluation unveils the relationship between the array organization and the cache access patterns, which is the fundamental of the

first-step optimization at design time. The conventional SRAM LLC usually utilizes unified shared structure because of the identical cell design [32, 33, 34]. The scenario in racetrack memory design is different. The hybrid-port array with many R-ports and a small number of R/W ports is good for regular and read-intensive cache accesses, such as instruction requests, while the uniform-port array with the same R/W ports are more suitable for data accesses with random behaviors. *Therefore, we propose a mixed array organization composing of both hybrid-port and uniform-port arrays.*

The SRAM based cache resizing techniques are realized by power gating [35]: some ways and/or sets of a cache are enabled/disabled by turning on/off the corresponding power supply. Since switching power supply involve charging/discharging a large amount of capacitance, the power gating cannot be applied frequently, indicating a coarse resize granularity in time domain. The physical resize granularity is also limited because a whole sub-array under the same power supply has to be enabled/disabled at a time. Moreover, the volatile feature of SRAM technology requires to evict dirty data when disabling cache block and reload them back in the following requests. These extra data migration results in performance/energy overheads as well as the increase of miss rate. A resizable racetrack cache can dynamically adjust the allowable set number or way number upon runtime cache access requirement is introduced in chapter 7. Unlike SRAM resizing, the resizability of racetrack LLC doesn't need power gate to shut down the power supply. The resizing is achieved by simply limiting the racetrack shift distance to enable/disable a certain number of magnetic domains.

1.3 DISSERTATION ORGANIZATION

The rest of the dissertation is organized as follows: Chapter 2 describes the background and related research works. Chapter 3 discusses the STT-RAM device optimization and presents the multi retention STT-RAM cache architecture. Chapter 4 introduces the process variation aware STT-RAM cache design by leveraging both the STT-RAM process variation and cache access pattern. Chapter 5 presents the importance of STT-RAM read performance. A dual mode architecture is introduced in the same chapter. In Chapter 6, a cross-layer design

consideration for racetrack memory cache design is presented. Chapter 7 further explores the potential of racetrack memory by a two-step optimization method. Finally, Chapter 8 draws conclusions of this dissertation.

2.0 BACKGROUND

2.1 STT-RAM BASICS

2.1.1 STT-RAM cell structure

The data storage device in a STT-RAM cell is the *magnetic tunnel junction* (MTJ), as shown in Figure 2(a) and (b). A MTJ is composed of two ferromagnetic layers that are separated by an oxide barrier layer (e.g., MgO). The magnetization direction of one ferromagnetic layer (the *reference layer*) is fixed while that of the other ferromagnetic layer (the *free layer*) can be changed by passing a current that is polarized by the magnetization of the reference layer. When the magnetization directions of the free layer and the reference layer are parallel (anti-parallel), the MTJ is in its low (high) resistance state.

The most popular STT-RAM cell design is one-transistor-one-MTJ (or 1T1J) structure, where the MTJ is selected by turning on the word-line (WL) that is connected to the gate of the NMOS transistor. The MTJ is usually modeled as a current-dependent resistor in the circuit schematic, as shown in Figure 2(c). When writing "1" (high-resistance state) into the STT-RAM cell, a positive voltage is applied between the source-line (SL) and the bit-line (BL). Conversely, when writing a "0" (low resistance state) into the STT-RAM cell, a negative voltage is applied between the SL and the BL.

The common read-out scheme of STT-RAM is shown in Figure 3. Applying a current I_{read} (Figure 3(a)) to the selected memory cell; the generated voltage on the bit line is compared to a reference signal in sense amplifier. If the generated voltage is higher (lower) than the reference, the data storage device in the memory cell is in the high- (low-) resistance state. The reference signal is normally generated by applying the same read voltage (current)

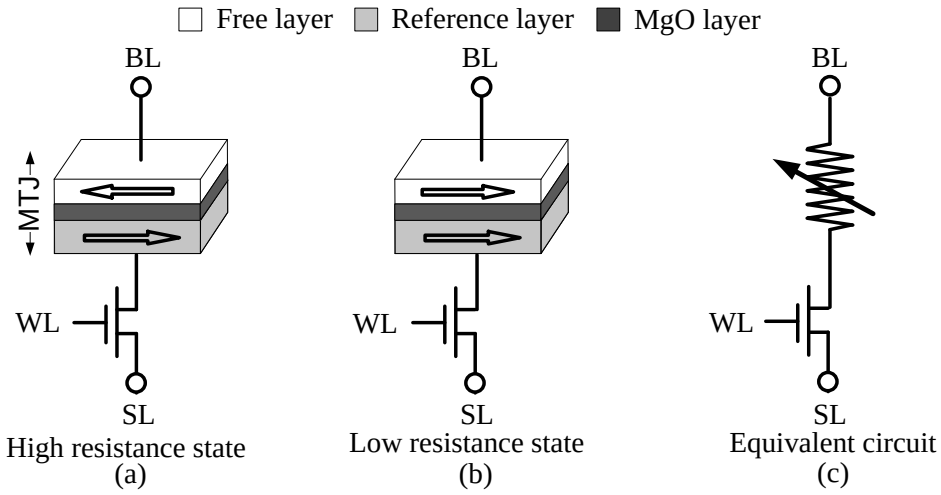


Figure 2: 1T1J STT-RAM design. (a) MTJ is in anti-parallel state; (b) MTJ is in parallel state; (c) The equivalent circuit.

on the dummy cell, whose resistance is $(R_L+R_H)/2$ ideally. The read operation can also be realized by applying read voltage on the STT-RAM cell as shown in Figure 3(b)

2.1.2 STT-RAM memory bank

As shown in Figure 3(c), similar to traditional SRAM array, several peripheral components such as word line row decoder, bit line column decoder and sense amplifier are necessary in STT-RAM memory bank design. The functionality of STT-RAM word line decoder is same with SRAM word line decoder which is used to select the target row based on the address of incoming access request. Because each row has multiple words with certain number of bits depending on the data bus width (usually 32bits or 64bits), bit line column decoder is responsible for selecting one word which is requested by the CPU. Number of sense amplifier and write driver also depends on the data bus width.

There are some differences between SRAM and STT-RAM architecture. In SRAM, pre-charge lines are connected to bit line and bit line bar to charge them up before reading the data of the cell, but pre-charge circuit is usually embedded in sense amplifier of STT-RAM design to balance the output and guarantee the accuracy of sensed result. Moreover, besides

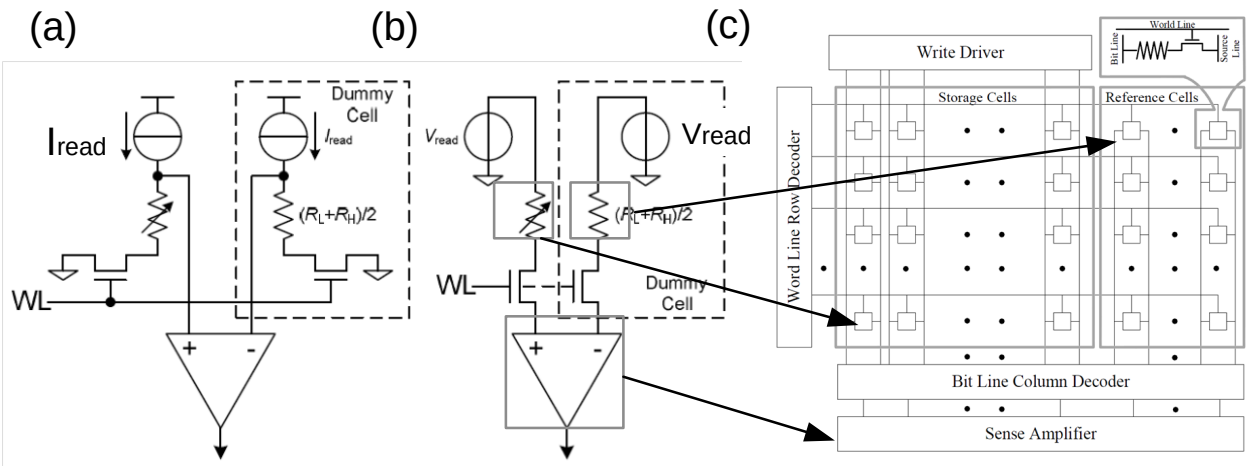


Figure 3: (a) Current driven read circuit; (b) Voltage driven read circuit; (c) STT-RAM array.

the storage cell array, there are bunch of reference cells shared by storage cells in STT-RAM architecture. The reference cell array has same number of rows as storage cell array, but each column of reference cell is shared by multiple storage cell columns. Compared to SRAM, STT-RAM is much smaller in its cell size. If we replace SRAM with STT-RAM within the same area, the memory density can be increased a lot. However, under the same CMOS technology, the peripheral circuit couldn't be shrunk accordingly. So the layout of e.g. row decoder must be redesigned with consideration to squeezed cell pitch. So is column decoder.

2.1.3 MTJ write performance vs. nonvolatility

The *data retention time*, T_{store}, of a MTJ is determined by the *magnetization stability energy height*, Δ:

$$T_{store} = \frac{1}{f_0} e^{\Delta}. \tag{2.1}$$

f_0 is the thermal attempt frequency, which is of the order of 1GHz for storage purposes [36]. Δ can be calculated by

$$\Delta = (\frac{K_u V}{k_B T}) = (\frac{M_s H_k V \cos^2(\theta)}{k_B T}), \tag{2.2}$$

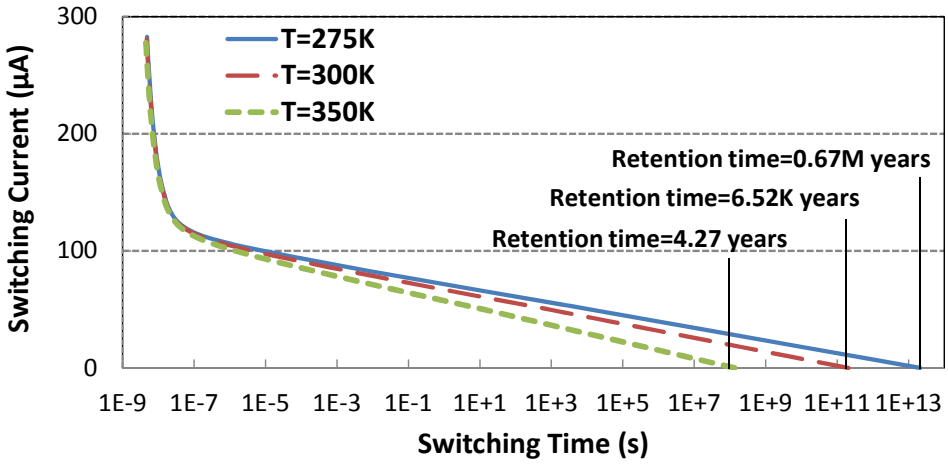


Figure 4: The relationship between the switching current and the switching time of "Base" MTJ design.

where M_s is the saturation magnetization. H_k is the effective anisotropy field including magnetocrystalline anisotropy and shape anisotropy. θ is the initial angle between the magnetization vector and the easy axis. T is working temperature. k_B is Boltzmann constant. V is the effective activation volume for the spin-transfer torque writing current. As Eq. (2.1) and (2.2) show, the data retention time of a MTJ decreases exponentially when its working temperature, T, rises.

The MTJ typically has three working regions which are identified based on the operation range of switching pulse width (T_{sw}): the thermal activation ($T_{sw} > 20ns$), the dynamic reversal ($3ns < T_{sw} < 20ns$), and the precessional switching ($T_{sw} < 3ns$). The required *switching current density*, J_C, of a MTJ operating in different working regions can be approximated as [37, 38]:

$$J_C^{\mathrm{THERM}}(T_{sw}) = J_{C0}(1 - \frac{1}{\Delta}\ln(\frac{T_{sw}}{\tau_0})) \qquad (T_{sw} > 10ns) \tag{2.3}$$

$$J_C^{\mathrm{DYN}}(T_{sw}) = \frac{J_C^{\mathrm{THERM}}(T_{sw}) + J_C^{\mathrm{PREC}}(T_{sw})e^{(-A(T_{sw}-T_{\mathrm{PIV}}))}}{1 + e^{(-A(T_{sw}-T_{\mathrm{PIV}}))}} \tag{2.4}$$

$(10ns > T_{sw} > 3ns)$

$$J_C^{\mathrm{PREC}}(T_{sw}) = J_{C0} + \frac{C\ln(\frac{\pi}{2\theta})}{T_{sw}} \qquad (T_{sw} < 3ns). \tag{2.5}$$

Here A, C and T_{PIV} are the fitting parameters. T_{sw} is the switching time of MTJ resistance. $J_C = J_C^{\mathrm{THERM}}(T_{sw})$, $J_C^{\mathrm{DYN}}(T_{sw})$ or $J_C^{\mathrm{PREC}}(T_{sw})$ are the required switching currents at T_{sw} in different working regions, respectively. The switching threshold current density J_{C0}, which causes a spin flip in the absence of any external magnetic filed at 0K, is given by:

$$J_{C0} = (\frac{2e}{\hbar})(\frac{\alpha}{\eta})(t_F M_s)(H_k \pm H_{ext} + 2\pi M_s). \tag{2.6}$$

Here e is the electron charge, α is the damping constant, τ_0 is the relaxation time, t_F is the free layer thickness, $\hbar$ is the reduced Planck's constant, H_{ext} is the external field, and η is the spin transfer efficiency.

2.2 RACETRACK MEMORY BASICS

2.2.1 Racetrack memory cell structure

As shown in Figure 5(a), an STT-RAM cell consists of a MTJ for data storage and an NMOS transistor. The NMOS transistor controls the access to the corresponding MTJ. It shall be large enough to provide sufficient current to switch the MTJ state. The STT-RAM density, therefore, is mainly determined by the access transistor size.

As the descendant of STT-RAM, the racetrack technology leverages the same physical fundamental for data storage. Figure 5(b) depicts the basic structure of planar racetrack [30]: a racetrack nanowire piles up many magnetic domains separated by ultra narrow domain walls. Each domain has its own magnetization direction, representing one-bit binary data. An access transistor is shared by multiple magnetic domains to conduct read and write operations. The density of racetrack array is not constrained by access transistors, but determined solely by the physical dimension of magnetic domains, which could be only $2F^2$. Since the storage elements and access devices in racetrack memory do not follow one-to-one correspondence relationship, a random access in Figure 5(b) requires two steps to complete. *Step 1–shift* the target magnetic domain and *align* it to an access transistor; *Step 2*–apply an appropriate voltage/current to *read* or *write* the target bit.

2.2.2 Racetrack memory bank design

Figure 6 depicts a general structure of racetrack array. A small portion of data bits in dark grey squares locate right on access ports and hence can be accessed directly. Most of data bits, however, need to be moved to an access port to enable read or write. Both ends of a racetrack strip attach some overhead bits. The number of overhead bits is determined by the *distance* (or, the number of magnetic domains) between two adjacent access ports. One or several columns of reference bits could be used to assist data detection in read operations. Similar to STT-RAM, a reference bit is required to provide a resistance value in the middle of the high- and low- resistances of data bits. The similar peripheral circuitry design of STT-RAM can be applied to racetrack memory. In addition, *racetrack status registers* and *shift drivers* are needed to record racetrack locations and control shift operations, respectively.

The racetrack technology itself is not preferable in tag array design, considering the shift-induced delay overhead in tag comparison. Previous racetrack designs [39, 40] adopted separated tag arrays in STT-RAM technology that can be accessed and updated quickly.

Figure 7 illustrates two physical-logic mapping methods supported by racetrack memory. Note that a data block of N bits corresponds to N magnetic domains on N racetrack nanowires within a single array. In Figure 7(a), a sub-array belongs to the same way but different sets. The sub-arrays in Figure 7(b) are partitioned in sets and the data bits on a racetrack nanowire fall into different associative ways. The physical-logical mapping of racetrack memory is very flexible and can be crafted to meet different design requirements. In Chapter 7, two types of resizable racetrack last-level caches are designed based on the mentioned two physical-logical mappings.

2.3 RELATED WORK

It is widely accepted that STT-RAM can save much more leakage power when compared to SRAM. Even by sacrificing some performance, using STT-RAM to directly replace SRAM is still considered to be worthy. Dong, et al. gave a comparison between the SRAM cache and

STT-RAM cache in a single-core microprocessor [41]. Desikan, et al. conducted an architectural evaluation of replacing on-chip DRAM with STT-RAM [42]. Sun, et al. extended the application of STT-RAM cache to Chip Multiprocessor (CMP) [10], and studied the impact of the costly write operation in STT-RAM on power and performance. However, all of these works have the similar conclusion that 10 % or even more performance degradation will be introduced by using STT-RAM directly as last level cache. Such performance degradation allows more design space at high performance computation area where power efficiency is still be highly emphasized. Many proposals have been made to address the slow write speed and high write energy of STT-RAM. Zhou, et al. proposed an early write termination scheme to eliminate the unnecessary writes to STT-RAM cells and save write energy [43]. A dual write speed scheme was used to improve the average access time of STT-RAM cache that distinguishes between the fast and slow cache portions [12]. In early 2011, Smullen, et al. proposed trading off the nonvolatility of STT-RAM for write performance and power improvement [20]. The corresponding DRAM-style refresh scheme to assure the data validity is not scalable for a large cache capacity. However, the single retention level cache design is lack of optimization space to maximize the benefits of STT-RAM writability and nonvolatility trad-offs. Also, the MTJ optimization technique they proposed, namely shrinking the cell surface area of the MTJ, is not efficient in the fast switching region ($< 10ns$).

Many recent works are dedicated to hybrid cache designs which promise more performance improvement by leveraging runtime cache access flow. A SRAM/STT-RAM hybrid cache hierarchy and some enhancements, such as write buffering and data migration were also proposed in [10, 44, 45]. The SRAM and STT-RAM cache ways are fabricated on the different layers in the proposed 3D integration. The hardware and communication overheads are relatively high. Chen, et al. proposed a dynamically reconfigurable cache to enhance last-level cache energy efficiency [46]. Li, et al. proposed to improve STT/SRAM hybrid cache data migration efficiency during compilation [47]. Li, et al. exploited the set-level write non-uniformity for NVM hybrid cache to improve energy efficiency [48].

Remember that the device modeling is the base of the architecture solution. MTJ parameters given in [20] is hard to be reproduced, which are overly optimistic in the fast-switching region ($<$ 3ns) in terms of write energy and performance, as well as data retention time.

Zhao, et al. reported a sub-nanosecond switching at the $45nm$ technology node for the in-plane MTJ devices [49]. The macro-magnetic model used in our work was verified by a leading magnetic recording company and calibrated with the latest in-plane MTJ measurement results of Zhao's work [49].

NUCA [23] considered access latency variation of the large on-chip cache memory and proposed interconnection-variation-aware data block rearrangement technology. Initiated by this important architecture, a lot of similar works [50, 51, 52, 53, 54, 55, 56] have been proposed to improve the efficiency of NUCA. Derived from NUCA[23], various memory technologies have different variation characteristics were optimized to deliver diverse variation-aware NUCA-like cache remapping. Some examples are SRAM [57, 58, 59, 60, 61], DRAM [62], PCM [63] and STT-RAM [64].

Most of the works related with racetrack memory are focus on the device theory or fabrication. Very recently, *TapeCache*, an early stage of estimation to utilize racetrack memory for data cache, was presented [40]. It showed 2.3× higher density, 1.4× power reduction and same performance compared to STT-RAM in last-level cache. However, the potential of the racetrack memory has not been fully explored. The *TapeCache* could be potentially improved from two aspects (1) the proposed marco cell based structure requires more racetrack overhead and peripheral circuit overhead that can be improved (2) a lot of space above the CMOS layer can be fully utilized by considering the racetrack memory cell scaling. [30] and [31] have also claimed racetrack memory has the potential to achieve $1F^2$ cell size. But without feasible design solution, it is hard to realize $1F^2$ per bit and still keep reasonable performance and energy budget.

Since the conventional SRAM based caches with the significant leakage currents are extremely energy hungry. Besides the efforts on process development and circuit design, many architecture resizing techniques have been explored to reduce leakage power and improve power efficiency of SRAM [65, 66, 67, 68]. Previously, the way-selective [69], set-selective [70], and hybrid-resizing cache designs [71] were proposed to change SRAM cache capacity to meet data requirement. Generally, there are two types of resizing policies to - static resizing and dynamic resizing. The static policy [69] adjusts the cache size before the execution of programs. This scheme can be easily implemented but it is unable to trace the

runtime data size changes very well. The dynamic resizing policy [70], on the contrary, can customize and optimize the cache size during runtime of programs by paying higher design cost.

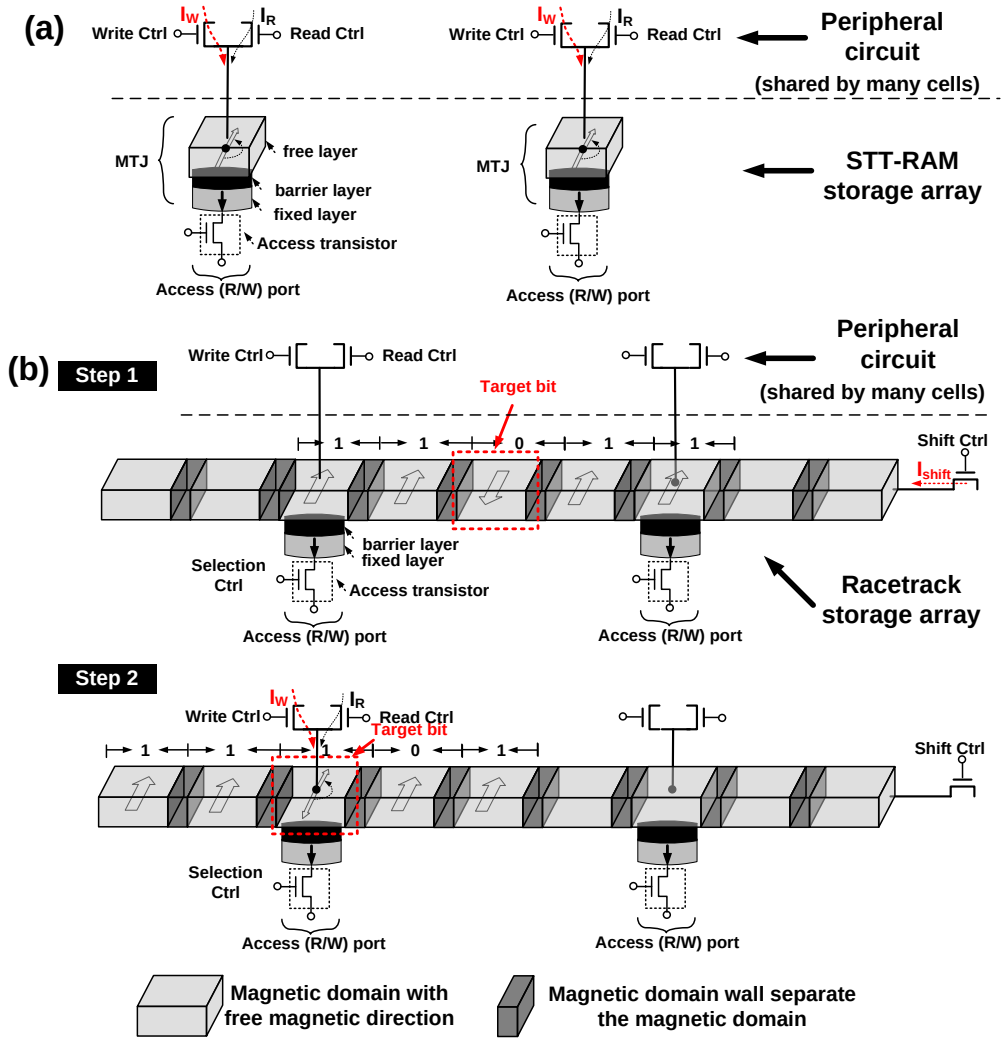


Figure 5: Cell design comparison: (a) STT-RAM; (b) Racetrack.

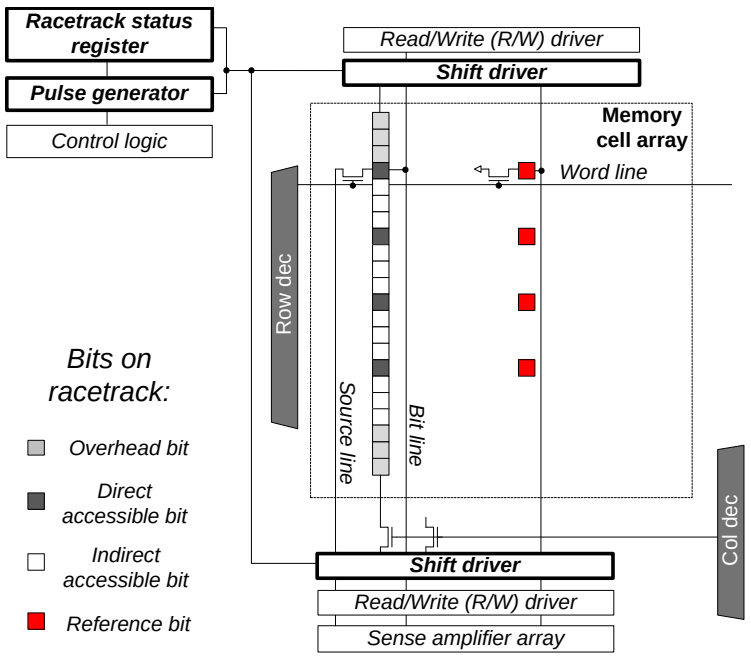


Figure 6: A general structure of racetrack array.

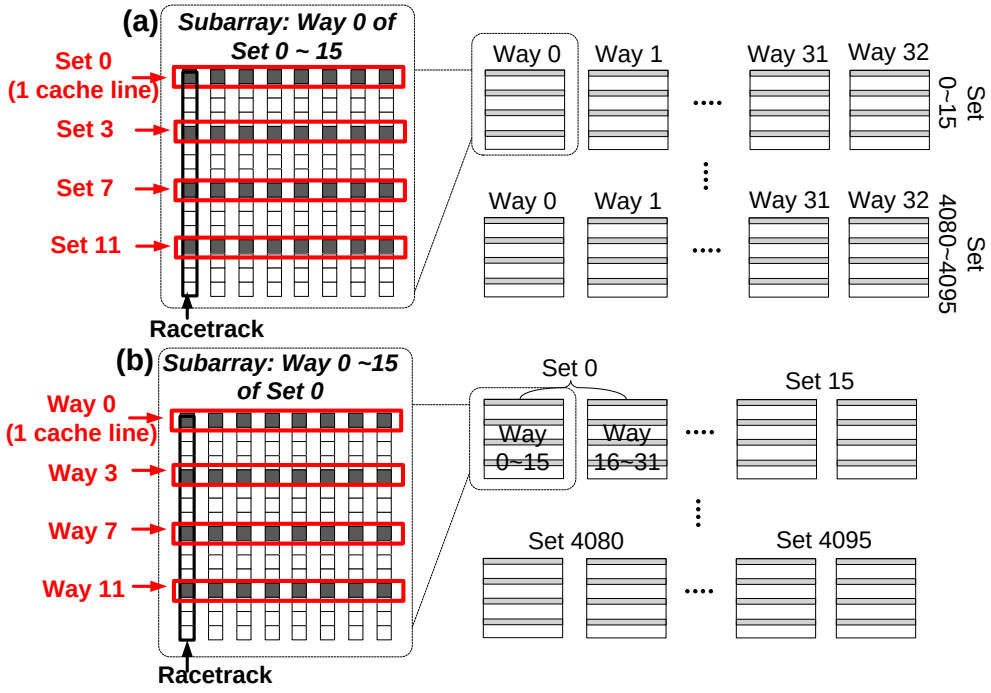


Figure 7: Two physical-logic mappings in racetrack memory.

3.0 MULTI-RETENTION STT-RAM DESIGN

In this chapter, we propose a range of cache hierarchy designs implemented entirely using STT-RAM that deliver optimal power saving and performance. In particular, our designs use STT-RAM cells with various data retention times and write performances, made possible by novel magnetic tunneling junction (MTJ) designs. For L1 caches where speed is of the utmost importance, we propose a scheme that uses fast STT-RAM cells with reduced data retention time coupled with a dynamic refresh scheme. In the dynamic refresh scheme, another emerging technology – memristor is used as the counter to monitor the data retention of the low-retention STT-RAM, achieving a higher array area efficiency than SRAM based counter. We propose the use of a hybrid lower-level STT-RAM design for cache with large capacity that simultaneously offers fast average write latency and low standby power. It has two cache partitions with different write characteristics and nonvolatility. A data migration scheme to enhance the cache response time to write accesses is also described. The proposed hybrid cache structure has been evaluated in lower level cache of both 2-level and 3-level cache hierarchies.

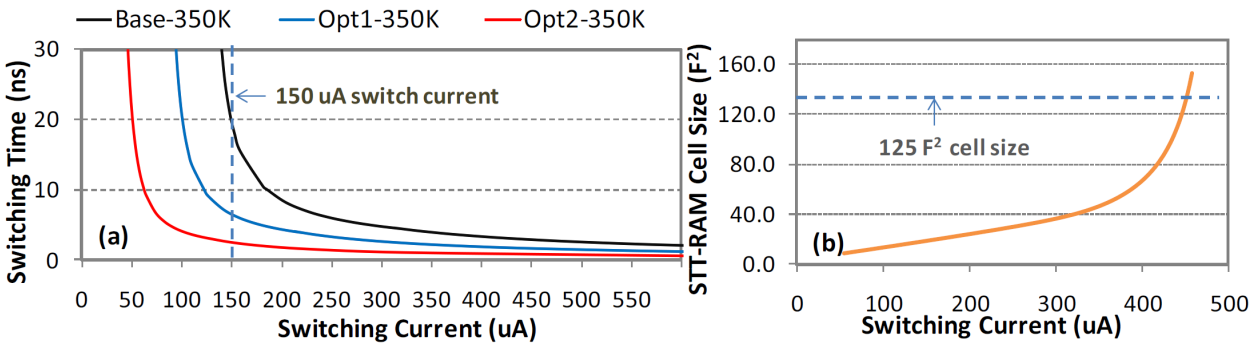


Figure 8: (a) MTJ switching performances for different MTJ designs at 350K. (b) The minimal required STT-RAM cell size at given switching current.

3.1 STT-RAM CELL DESIGN OPTIMIZATION

As proposed by [20], shrinking the cell surface area of the MTJ can reduce Δ, and consequently decreases the required switching density J_C, as shown in Eq. (2.3). However, such a design becomes less efficient in the fast switching region (T_{SW}<3ns) because the coupling between Δ and J_C is less in this region, as shown in Eq. (2.5). Based on the MTJ switching behavior, we propose to change M_s, H_k, or t_F to reduce J_c. Such a technique can lower not only Δ but also J_{c0}, offering efficient performance improvement over the entire MTJ working range.

We simulated the switching current versus the switching time of a baseline $45nm \times 90nm$ elliptical MTJ over the entire working range, as shown in Figure 4. The M_s, H_k and t_F are tuned in the finite element micro-magnetic simulations. The simulation is conducted by solving the stochastic magnetization dynamics equation describing spin torque induced magnetization motion at finite temperature [72]. The MTJ parameters are taken from [72], which are close to the measurement results recently reported in [49]. By tuning M_s, H_k and t_F, different critical switching current J_{c0} can be obtained. The MTJ data retention time is measured as the MTJ switching time when the switching current is zero. When the

working temperature rises from 275K to 350K, the MTJ's data retention time decreased from 6.7×10^6 years to 4.27 years. In the experiments reported in this work, we shall assume that the chip is working at a high temperature of 350K.

To quantitatively study the trade-offs between the write performance and nonvolatility of a MTJ, we simulated the required switching current of three different MTJ designs with the same cell surface shapes. Besides the "Base" MTJ design shown in Figure 4, two other designs ("Opt1" and "Opt2") that are optimized for better switching performance with degraded nonvolatility were studied. Compared to the "Base" MTJ design with a thermal stability (Δ) 40, the Δ of "Opt1" and "Opt2" are 22 and 8 respectively in order to achieve the target switching performance. The corresponding MTJ switching performances of these three designs at 350K are shown in Figure 8(a). The detailed comparisons of data retention times, the switching currents, the bit write energies, and the corresponding STT-RAM cell sizes of three MTJ designs at the given switching speed of 1ns, 2ns, and 10ns are given in Figure 9.

Significant write power saving is achieved if the MTJ's nonvolatility can be relaxed. For example, when the MTJ data retention time is scaled from 4.27 years ("Base") to 26.5μs ("Opt2"), the required MTJ switching current decreases from 185.2μA to 62.5μA for a 10ns switching time at 350K. Or, at a MTJ switching current of 150μA, the corresponding switching times of all three MTJ designs varied from 20ns to 2.5ns. A switching performance improvement of 8$\times$ can be obtained, as shown in Figure 8(a).

Since the switching current of a MTJ is proportional to its area, the MTJ is normally fabricated with the smallest possible dimension. The STT-RAM cell's area is mainly constrained by the NMOS transistor which needs to provide sufficient driving current to the MTJ. Figure 8(b) shows the minimal required NMOS transistor size when varying the switching current, and the corresponding STT-RAM cell area at 45nm technology node. The PTM model was used in the simulation [73] and the power supply V_{DD} is set to 1.0V. Memory cell area is measured in F^2, where F is the feature size at a certain technology node.

According to the popular cache and memory modeling software CACTI [74], the typical cell area of SRAM is about 125F^2. For a STT-RAM cell with the same area, the maximum current that can be supplied to the MTJ is 448.9μA. A MTJ switching time of less than

1ns can be obtained with the "Opt2" design under such as a switching current while the corresponding switching time for the baseline design is longer than 4.5ns. In this paper, we will not consider designs that are larger than $125F^2$.

Since "Opt1" and "Opt2" requires less switching current than the baseline design for the same write performance, they also consume less write energy. For instance, the write energies of "Base" and "Opt2" designs are $1.85pJ$ and $0.62pJ$, respectively, for a switching time of 10ns. If the switching time is reduced to 1ns, the write energy of "Opt2" design can be further reduced down to $0.32pJ$. The detailed comparisons on the write energies of different designs can be found in Figure 9(d).

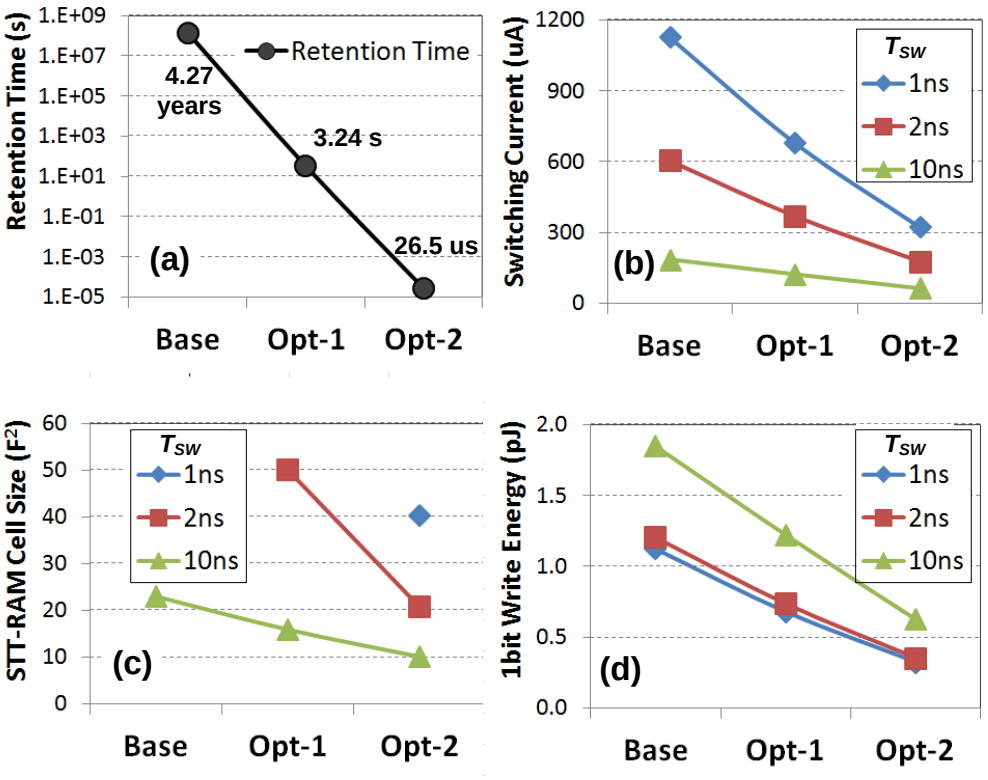


Figure 9: Comparison of different MTJ designs at 350K: (a) the retention time, (b) the switching current, (c) STT-RAM cell size, and (d) the bit write energy.

3.2 MULTI RETENTION LEVEL STT-RAM CACHE HIERARCHY

Multi retention level STT-RAM based cache hierarchy is presented in the following chapter. The multi retention level STT-RAM cache hierarchy takes into account the difference in access patterns in L1 and the lower level cache (LLC).

For L1, the overriding concern is access latency. Therefore, we propose the use of our "Opt2" nonvolatility-relaxed STT-RAM cell design as the basis of the L1 cache. In order to prevent data loss introduced by relaxing its nonvolatility, we propose a dynamic refresh scheme to monitor the lifespan of the data, and refresh cells when needed. LLC caches are much larger than L1 cache. As such, a design built with only "Opt2" STT-RAM cells will consume too much refresh energy. Using of the longer retention "Base" or "Opt1" design is more practical. However, to recover the lost performance, we propose a hybrid LLC that has a regular and a nonvolatility-relaxed STT-RAM portions. Data will be migrated from one to the other accordingly. The details of our proposed cache hierarchy will be given in the following subsections.

3.2.1 The nonvolatility-relaxed STT-RAM L1 cache design

As established earlier, using the "Opt2" STT-RAM cell design for L1 caches can significantly improve the write performance and energy. However, its data retention time of $26.5\mu s$ may not be sufficient to retain the longest living data in L1. Therefore, a refresh scheme is needed. In [20], a simple DRAM-style refreshing scheme was used. The refresh operation essentially rewrites all the memory to stabilize the data stored in memory. This scheme refresh all cache blocks in sequence regardless of its data content. Read and write accesses to memory cells that are being refreshed must be stalled. As we shall show in Section 3.3.2, this simple scheme introduces many unnecessary refreshing operations whose elimination will significantly improve performance and save energy.

Dynamic refresh scheme

To eliminate unnecessary refresh, we propose the use counters to track the lifespan of cache data blocks. Refresh is performed only on cache blocks that have reached their full

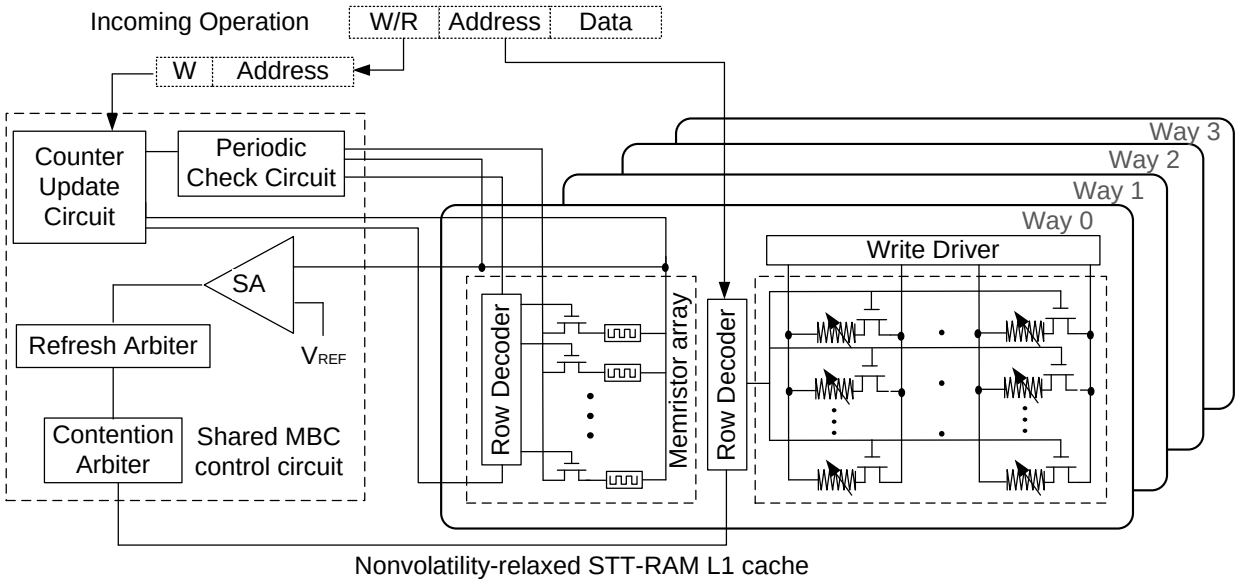


Figure 10: Memristor counter-based refreshing scheme.

lifespan. In our refresh scheme, we assign one counter to each data block in the L1 cache to monitor its data retention status. Figure 10 illustrates our dynamic refresh scheme. The operation of the counter can be summarized as the follows:

Reset: On any write access to a data block, its corresponding counter is reset to '0'.

Pushing: We divide the STT-RAM cell's retention time into N_{mem} periods, each of which is T_{period} long. A global clock is used to maintain the count-down to T_{period}. At the end of every T_{period}, the level of every counter in the cache is increased by one.

Checking: The data block corresponding to a counter would have reached the maximum retention time when the counter reaches its highest level, and hence needs to be refreshed. The overhead of such counter pushing scheme is very moderate. Take, for example, a 32KB L1 cache built using the "Opt2" STT-RAM design and a counter can represent 16 values from 0 to 15. A pushing operation happens once every $3.23ns = (26.5\mu\text{s}/512/16)$ in the entire L1 cache. This is more than 6 cycles at a $2GHz$ clock frequency. A larger cache may mean a higher pushing overhead.

Table 2: Comparison between SRAM and memristor counter

A 4-bit counter	**SRAM**	**Memristor**
Area of a cell	$100 \sim 150F^2$	$33F^2$
Number of cells	4	1
Pushing and checking eng	$0.7pJ$	$0.45pJ$
Reset energy	$0.46pJ$	$7.2pJ$
Sense margin	$50 \sim 100mV$	$46.875\ mV$

The following is some design details of the proposed dynamic refresh scheme:

Cache access during refresh: During a refresh operation, the block's data is read out into a buffer, and then saved back to the same cache block. If a read request to the same cache block comes before the refresh finishes, the data is returned from this buffer directly. There is therefore no impact on the read response time of the cache. Should a write request comes, the refresh operation is terminated immediately, and the write request is executed. Again, no penalty is introduced.

Reset threshold N_{th}: However, we observe that during the lifespan of a cache block, updates happen more frequently within a short period of time after it has been written. Many resets of the cache block data occur far from their data retention time limits, giving us an optimization opportunity. We altered the reset scheme to eliminate counter resets that happen within a short time period after data has been written. We define a threshold level, N_{th}, that is much smaller than N_{mem}. The counter is reset only when its resistance is higher than N_{th}. The larger N_{th} is, the more resets are eliminated. On the other hand, the refresh interval of the data next written into the same cache block is shortened. However, our experiments in Section 3.3.2 shall show that such cases happen very rarely and the lifetimes of most data blocks in the L1 cache are much shorter than 26.5μs.

Counter design

In the proposed scheme, the counters are used in two ways: 1) to monitor the time duration for which the data has been written into the memory cells, and 2) to monitor the

read and write intensity of the memory cells. These counters can be implemented either by the traditional SRAM or the recently discovered memristor device. The design detail of memristor as an on-chip analog counter will be introduced here. A Verilog-A model for spintronic memristor [75] was used in circuit simulations.

When the magnitude of programming pulse is fixed, the memristance (resistance) of a spintronic memristor is determined by the accumulated programming pulse width. We utilize this characteristic to implement a high density, low power, and high performance counter used in our cache refresh scheme: the memristance of a memristor can be partitioned into multiple levels, corresponding to the values the counter can record.

The maximum number of memristance levels is constrained by the minimal sense margin of the sense amplifier/comparator and the resolution of the programming pulse, i.e., the minimal pulse width. The difference between R_H and R_L of the spintronic memristor used in this work is 5000Ω (see 2), which is sufficiently large to be partitioned into 16 levels. Moreover, we use the pushing current of $150\mu A$ as the read current, further enlarging the sensing margin. The sense margin of the memristor-controlled counter $\Delta V = 46.875mV$ ($150\mu A \times 5000\Omega$ / 16 levels) is at the same level as the sense margin in nowadays SRAM design.

The area of a memristor is only $2F^2$ (refer Table 2). The total size of a memristor counter including a memristor and a control transistor is below $33F^2$. For comparison, the area of a 6T SRAM cell is about $100 \sim 150f^2$ [76]. When using SRAMs as the counter, the mismatched pitch to dense STT-RAM results in a much lower area efficiency. More importantly, the memristor counter has the same layout structure as STT-RAM and therefore can be easily integrated into STT-RAM array.

The memristance variation induced by process variations [77] is the major issue when utilizing memristors as data storage device. The counter design faces the same issue but the impact is not that critical: as a timer, the memristance variation can be overcome by giving enough design margin to guarantee the on-time refresh.

Every *pushing and checking* operation of a SRAM counter should include two actions: increase the counter value by one and read it out. In the proposed memristor counter design, the injected current can obtain the two purposes simultaneously – pushing the domain wall

to enable counter value increment and meanwhile serving as read current for data detection. The comparison between the two types of counter designs is summarized in 2. Note that the memristor counter has a larger energy consumption during a reset operation in which its domain wall moves from one end to the other.

3.2.2 Lower level cache with mixed high and low retention STT-RAM cells

The data retention time requirement in the mainstream STT-RAM development of 4∼10 years was inherited from Flash memory designs. Although such a long data retention time can save significant standby power of on-chip caches, it also entails a long write latency (∼ 10ns), and large write energy [10]. Relaxing the nonvolatility of the STT-RAM cells in the lower level cache will improve write performance as well as save more energy. However, if further reducing retention time to μs scale, *e.g.*, 26.5μs of our "Opt2" cell design, the refresh energy dominates and hence any refresh scheme becomes impractical for the large lower level cache.

The second technique we proposed is a hybrid memory system that has both high and low retention STT-RAM portions to satisfy both the power and performance targets simultaneously. We take a L2 cache with 16 ways as a case study as shown in Figure 11, way 0 of the 16-way cache is implemented with a low retention STT-RAM design ("Opt2") while ways 1 to 15 are implemented with the high retention STT-RAM ("Base" or "Opt1"). Write intensive blocks are primarily allocated from way 0 for a faster write response, while read intensive blocks are maintained in the other ways.

Like our proposed L1 cache, counters are used in way 0 to monitor the blocks' data retention status. However, unlike in L1 where we perform a refresh when a memristor counter expires, here we move the data to the high retention STT-RAM ways.

Figure 11 demonstrates the data migration scheme to move the data between the low and the high retention cache ways based on their write access patterns. A *write intensity prediction queue* (WIPQ) of 16 entries is added to record the write access history of the cache. Every entry has two parts, namely, the data address and an access counter.

During a read miss, the new cache block is loaded to the *high-retention* (HR) region (ways 1-15) following the regular LRU policy. On a write miss, the new cache block is allocated from the *low-retention* (LR) region (way 0), and its corresponding memristor counter is reset to '0'. On a write hit, we search the WIPQ first. If the address of the write hit is already in WIPQ, the corresponding access counter is incremented by one. Note that the block corresponding to this address may be in the HR- or the LR-region of the cache. Otherwise,

the hit address will be added in to the queue if any empty entry available. If the queue is full, the LRU entry will be evicted, and replaced by the current hit address. The access counters in the WIPQ are decremented periodically, for example, every $2,000$ clock cycles, so that the entries that are in the queue for too long will be evicted. Once an access counter in a WIPQ entry reaches a preset value, $N_{\mathrm{HR}\to\mathrm{LR}}$, the data stored in the corresponding address will be swapped with a cache block in the LR-region. If the corresponding address is already in the LR-region, no further action is required. A read hit does not cause any changes to the WIPQ.

Likewise, a *read intensity record queue* (RIRQ) with the same structure and number of entries is used to record the read hit history of the LR-region. Whenever there is a read hit to the LR-region, a new entry is added into the RIRQ. Or if a corresponding entry already exist in the RIRQ, the value of the access counter is increased by one. When the memristor counter of a cache block B_i in the LR-region indicates the data is about to become unstable, we check to see if this cache address is read intensive by searching the RIRQ. If B_i is read intensive, it will be moved to HR-region. The cache block being replaced by B_i in the HR-region will be selected using the LRU policy. The evicted cache block will be send to main memory. If B_i is not read intensive, it will be written back to main memory.

In a summary, our proposed scheme uses the WIRQ and RIRQ to dynamically classify cache blocks into three types:

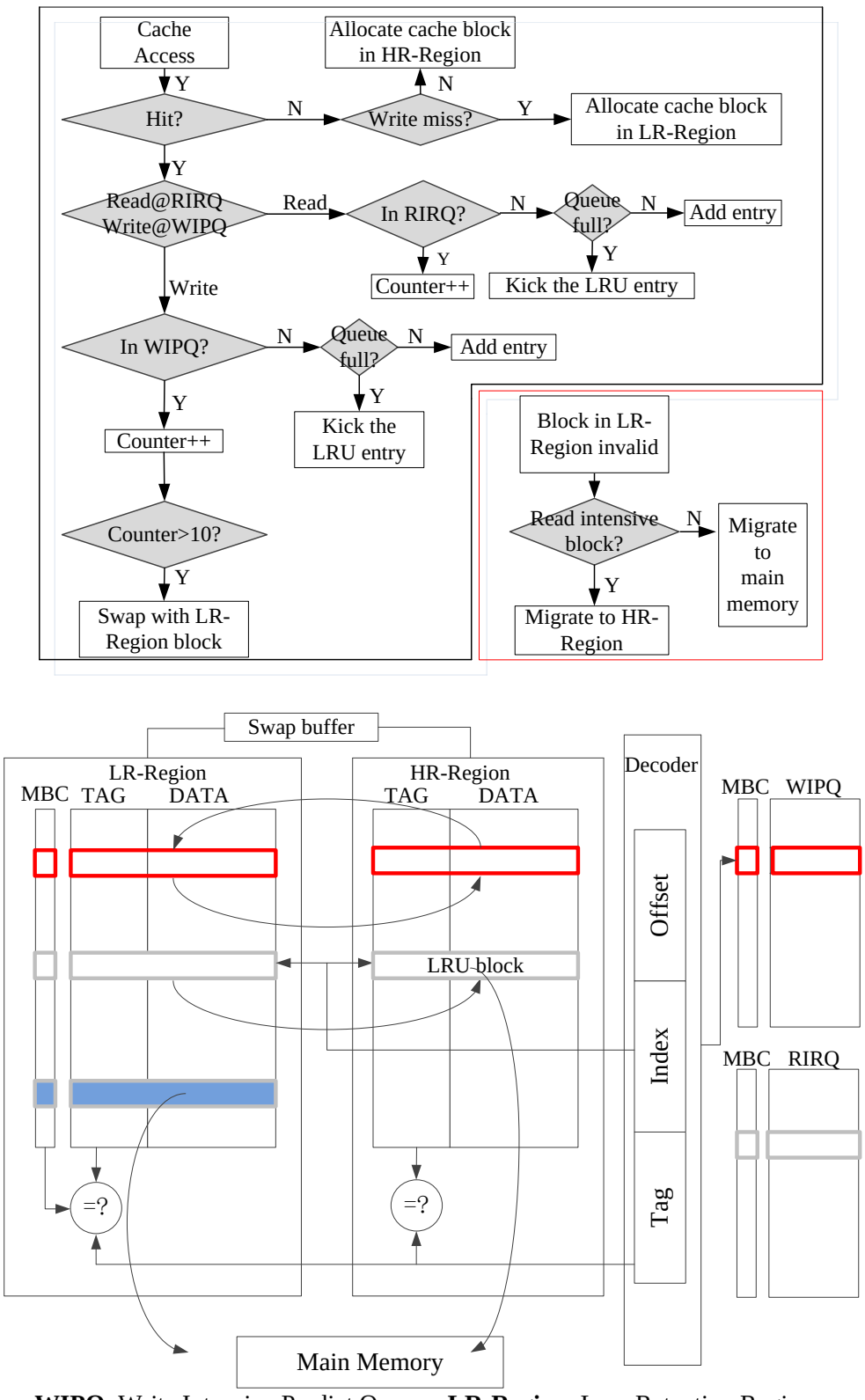
Write intensive: The addresses of such cache blocks are kept in the WIRQ. They will be moved to the LR-region once their access counters in WIRQ reach $N_{\mathrm{HR}\to\mathrm{LR}}$;

Read intensive but not write intensive: The addresses of such cache blocks are found in the RIRQ but not the WIRQ. As they approach to their data retention time limit, they will be moved to the HR-region.

Neither write nor read intensive: Neither WIRQ nor RIRQ has their addresses. They are kept in HR-region, or evicted from LR-region to main memory directly.

Identifying a *write intensive* cache blocks also appeared in some previous works. In [10], they check if two successive write accesses go to the same cache block. It is highly possible that a cache block may be accessed several times within very short time, and then becomes inactive. Our scheme is more accurate and effective as it monitors the read and write access

histories of a cache block throughout its entire lifespan. The RIRQ ensures that *read intensive* cache blocks migrate from the LR-region to HR-region in a timely manner that, at the same time, also improves energy efficiency and performance.



WIPQ: Write Intensive Predict Queue
RIRQ: Read Intensive Record Queue
LRU: Least Recent Used
LR-Region: Low-Retention Region
HR-Region: High-Retention Region
MBC: Memristor Bit Counter

Figure 11: Hybrid lower level cache migration policy: Flow graph (left). Diagram (right).

Table 3: Simulation Platform

Max issue width:	4 insts
Fetch width:	4 insts
Dispatch width:	4 insts
Write back width:	4 insts
Commit width:	4 insts
Fetch queue size:	32 insts
Reorder buffer:	64 entries
Max branch in pipeline:	24
Load store queue size:	32 entries
Functional units:	2 ALU 2 FPU
Clock cycle period:	0.5 ns
Main memory:	200 cycle latency

3.3 SIMULATION RESULTS & DISCUSSION

3.3.1 Experimental setup

We modeled a 2GHz microprocessor with 4 out-of-order cores using MARSSx86 [78]. Assume a two-level or a three-level cache configuration and a fixed 200-cycle main memory latency. The MESI cache coherency protocol is utilized in the private L1 caches to ensure consistency, and the shared lower level cache uses a write-back policy. The parameters of our simulator and cache hierarchy can be found in Table 3 and 4.

Table 5 shows the performance and energy consumptions of various designs obtained by a modified NVSim simulator [79]. All the "*-hi*", "*-md*", and "*-lo*" configurations use the "Base", "Opt1", and "Opt2" MTJ designs, respectively. Note that as shown in Figure 8, they scale differently. We simulated a subset of multi-threaded workloads from the PARSEC 2.1 and the SPEC 2006 benchmark suites so as to cover a wider spectrum of read/write and

Table 4: Cache hierarchy configuration

Baseline 2-level cache hierarchy
Local L1 Cache: 32KB 4-way, 64B cache block;
Shared L2 Cache: 4MB 16-way, 128B cache block.
3-level cache hierarchy
Local L1 Cache: 32KB 4-way, 64B cache block;
Local L2 Cache: 256KB 8-way, 64B cache block;
Shared L3 cache: 4MB 16-way, 128B cache block.

cache miss characteristics. We simulated 500 million instructions of each benchmark after their initialization.

SPICE simulations were conducted to characterize the performance and energy overheads of the memristor counter and its control circuity. The reset energy of a memristor counter is $7.2pJ$, and every pushing-checking operation consumes $0.45pJ$.

We compared the performance (in terms of instruction per cycle, IPC) and the energy consumption of different configurations for both 2-level and 3-level hybrid cache hierarchies. The conventional all SRAM cache design is used as the baseline. The optimal STT-RAM cache configuration based on our simulations is summarized as follows. The detailed experimental results will be shown and discussed in Section 3.3.2, 3.3.3, and 3.3.4.

An optimal 2-level STT-RAM cache hierarchy is the combination of (a) a L1 cache of the "L1-lo2" design, and (b) a hybrid L2 cache of using the "L2-lo" in the LR-region and "L2-md2" in the HR-region;

An optimal 3-level STT-RAM cache hierarchy is composed of (a) a L1 cache of the "L1-lo2" design, (b) a hybrid L2 cache of using the "L2-lo" in the LR-region and "L2-md1" in the HR-region and (c) a hybrid L3 cache of the "L3-lo" design in the LR-region and "L3-md2" in the HR-region.

Table 5: Cache Configuration

	32KB L1 Cache					
	SRAM	lo1	lo2	lo3	md	hi
Cell Size (F^2)	125	20.7	27.3	40.3	22	23
MTJ Switching Time (ns)	/	2	1.5	1	5	10
Retention Time	/	$26.5\mu s$			$3.24s$	$4.27yr$
Read Latency (ns)	1.113	0.778	0.843	0.951	0.792	0.802
Read Latency (cycles)	3	2	2	2	2	2
Write Latency (ns)	1.082	2.359	1.912	1.500	5.370	10.378
Write Latency (cycles)	3	5	4	4	11	21
Read Dyn. Energy (nJ)	0.075	0.031	0.035	0.043	0.032	0.083
Write Dyn. Energy (nJ)	0.059	0.174	0.187	0.198	0.466	0.958
Leakage Power (mW)	57.7	1.73	1.98	2.41	1.78	1.82
	4MB L2 or L3 Cache					
	SRAM	lo	md1	md2	md3	hi
Cell Size (F^2)	125	20.7	22	15.9	14.4	23
MTJ Switching Time (ns)	/	2	5	10	20	10
Retention Time	/	$26.5\mu s$			$3.24s$	$4.27yr$
Read Latency (ns)	4.273	2.065	2.118	1.852	1.779	2.158
Read Latency (cycles)	9	5	5	4	4	5
Write Latency (ns)	3.603	3.373	6.415	11.203	21.144	11.447
Write Latency (cycles)	8	7	13	23	43	23
Read Dyn. Energy (nJ)	0.197	0.081	0.083	0.070	0.067	0.085
Write Dyn. Energy (nJ)	0.119	0.347	0.932	1.264	2.103	1.916
Leakage Power (mW)	4107	96.1	104	69.1	61.2	110

3.3.2 Results for the proposed L1 cache design

To evaluate the impacts of using STT-RAM in L1 cache design, we implemented the L1-cache with the different STT-RAM designs listed in the L1 Cache portion of Table 5 while leaving the SRAM L2 cache unchanged. Due to the smaller STT-RAM cell size, the overall area of L1 cache is significantly reduced. The delay components of interconnect and peripheral circuits also decrease accordingly. Even considering the relatively long sensing latency, the read latency of STT-RAM L1 cache is still similar, or even slightly lower than that of a SRAM L1 cache. However, the write performance of STT-RAM L1 cache is always slower than the SRAM L1 cache for all the design configurations considered. The leakage power consumption of the STT-RAM caches come from the peripheral circuits only, and is very low. The power supply to the memory cells that are not being accessed can be safely cutoff without fear of data loss until the data retention limit is reached.

Figure 12 illustrates the ratio between read and write access numbers in L1 D-cache. Here, the read and write access numbers are normalized to the total L1 cache access number of `blackscholes`. The ratio reflects the sensitivity of the L1 cache in terms of performance, the dynamic energy toward per-read and per-write latency, and energy of the L1 cache.

Figure 13 shows the IPC performance of the simulated L1 cache designs normalized to the baseline all-SRAM cache. On average, implementing the L1 cache using the "Base" (used in "L1-hi") or "Opt1" (used in "L1-md") STT-RAM design incurs more than 32.5% and 42.5% IPC degradation, respectively, due to the long write latency. However, the performance of the L1 caches with the low retention STT-RAM design significantly improves compared to that of the SRAM L1 cache: the average normalized IPC's of 'L1-lo1', 'L1-lo2', and 'L1-lo3' are 0.998, 1.092, and 1.092, respectively. The performance improvement of 'L1-lo2' or 'L1-lo3' L1 cache w.r.t the baseline SRAM L1 cache comes from the shorter read latency even though its write latency is still longer (see Table 5). However, L1 read accesses are far more frequent than write access in most benchmarks as shown in Figure 12. In some benchmarks whose read/write ratio is pretty high, for example, `swaptions`, the 'L1-lo2' or 'L1-lo3' design achieves a better than 20% improvement in IPC. The energy consumptions of the different L1 cache designs normalized to the baseline all-SRAM cache are summarized

in Figure 14(a). The reported results includes the energy overhead of the refresh scheme and the counters, where applicable. Not surprisingly, all three low retention STT-RAM L1 cache designs achieved significant energy savings compared to the SRAM baseline. The "L1-lo3" design consumes more energy because of its larger memory cell size, and larger peripheral circuit having more leakage and dynamic power, as shown in Table 5. Figure 14(a) also shows that implementing the L1 cache with the "Base" (used in "L1-hi") or "Opt1" (used in "L1-md") STT-RAM is much less energy-efficient because (1) the MTJ switching time is longer, resulting in a higher write dynamic energy, and (2) a longer operation time due to the low IPC.

Figure 14(b) presents the breakdowns of the read dynamic energy, the write dynamic energy and the leakage energy in the baseline SRAM cache. First, the leakage occupies more than 30% of overall energy, most of which can be eliminated in STT-RAM design. Second, when comparing to Figure 12, we noticed that the dynamic read/write energy ratio is close to the read/write access ratio. The high read access ratio together with the lower per-bit read energy consumption of STT-RAM results in a much lower dynamic energy of STT-RAM L1 cache design. Therefore, "L1-lo1", "L1-lo2" and "L1-lo3" STT-RAM designs save up to 30% to 40% of overall energy compared to the baseline SRAM L1 cache.

Figure 15(a) compares the refresh energy consumptions of the 'L1-lo2' L1 cache under different refresh schemes. In each group, the three bars from left to right represent the refresh energy consumptions of DRAM style refresh scheme, refresh scheme without reset threshold N_{th}, and with $N_{\text{th}} = 10$, respectively. The refresh energy consumptions are normalized to the overall L1 energy consumptions when implementing the refresh scheme with $N_{\text{th}} = 10$. Note that the y-axis is in logarithmic scale.

The energy consumption of the simple DRAM-style refresh scheme accounts for more than 20% of the overall L1 cache energy consumption on average. In some extreme cases of low write access frequency, for example, `mcf`, this ratio is as high as 80% because of the low dynamic cache energy consumption. The total energy consumption of our proposed refresh scheme consists of the checking and pushing, the reset, and the memory cell refresh.

As we discussed in Section 3.2.1, the introduction of the reset threshold N_{th} can further reduce the refresh energy consumption by reducing the number of counter resets. This is

confirmed in Figure 15(a) and (b). The number of counter reset operations are reduced by more than 20× on average after setting a reset threshold N_{th} of 10, resulting in more than 95% of the reset energy saving. The energy consumptions for the reset and pushing&checking are marginal compared to overall refresh energy as shown in Figure 15(a). So the energy consumption on the memristor counters has less effect on the overall energy consumption. By using the dynamic refresh scheme, the refresh energy reduces from 20% to only 4.35% of the overall L1 cache energy consumption. By accurately monitoring the lifespan of the cache line data, our refresh scheme significantly reduced the refresh energy in all the benchmarks.

The refresh energy saving by utilizing the dynamic refresh scheme is determined by the cache write access distribution and intensity. Figure 16 demonstrates the distribution of average write access intervals obtained from four selected benchmarks. In each subfigure, the STT-RAM retention time is represented by the red vertical line. Therefore, the data stored in those cache lines on the right side of the red line need refreshment to maintain correctness.

We also collected the cache write access intensities with time. The results of two selected benchmarks are shown in Figure 17. For illustration purpose, we divide the overall simulation time into ten periods and partition cache lines into eight groups. Figure 17 exhibits the average write access intervals for all the cache line groups in each time period. Benchmark `hammer` has a relatively uniform cache write intensity. Its average write access interval is less than $2\mu s$, which is much shorter than the STT-RAM data retention time $26.5\mu s$. Often the cache lines are updated by regular write access without refreshed. Therefore, the dynamic refresh scheme can reduce the refresh energy of `hammer` significantly – from 30% to 1% of the total energy consumption when DRAM-style refresh is utilized. On the contrary, benchmark `gobmk` demonstrates a completely uneven write access intensities among different cache lines. Moreover, the access intervals of many cache lines are longer than the data retention time, making refresh necessary. The dynamic refresh scheme does not benefit too much in such a type of programs.

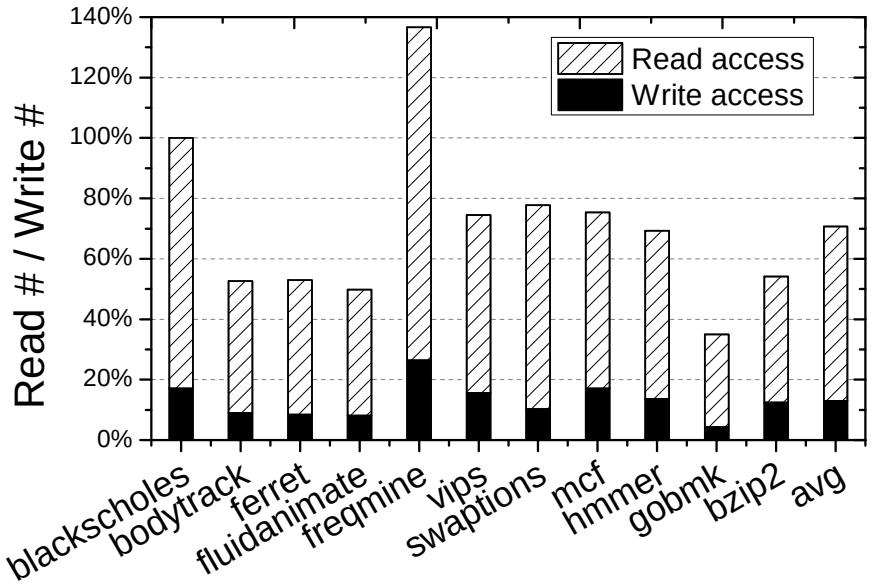


Figure 12: The normalized L1 cache read and write access numbers. The access numbers are normalized to the total L1 access number of `blackscholes` (note: "avg" means arithmetic mean).

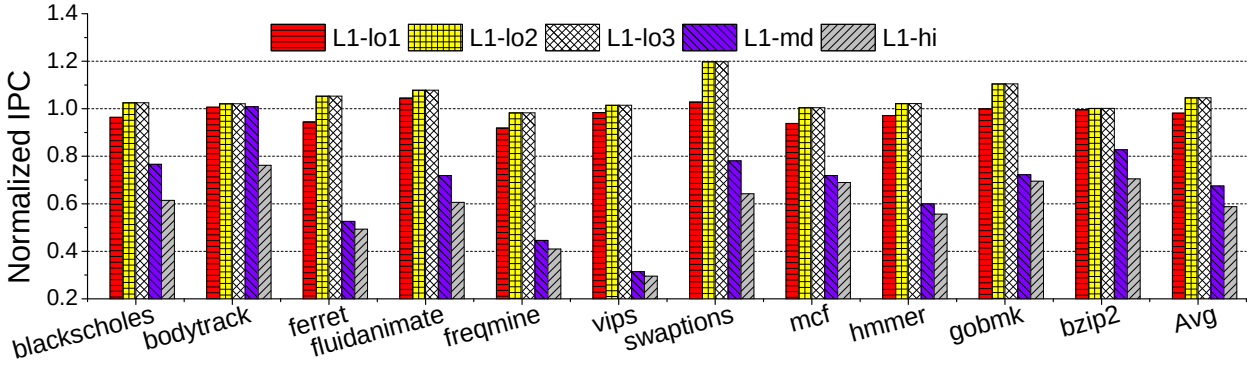


Figure 13: IPC comparison of various L1 cache designs. The IPC's are normalized to all-SRAM baseline.

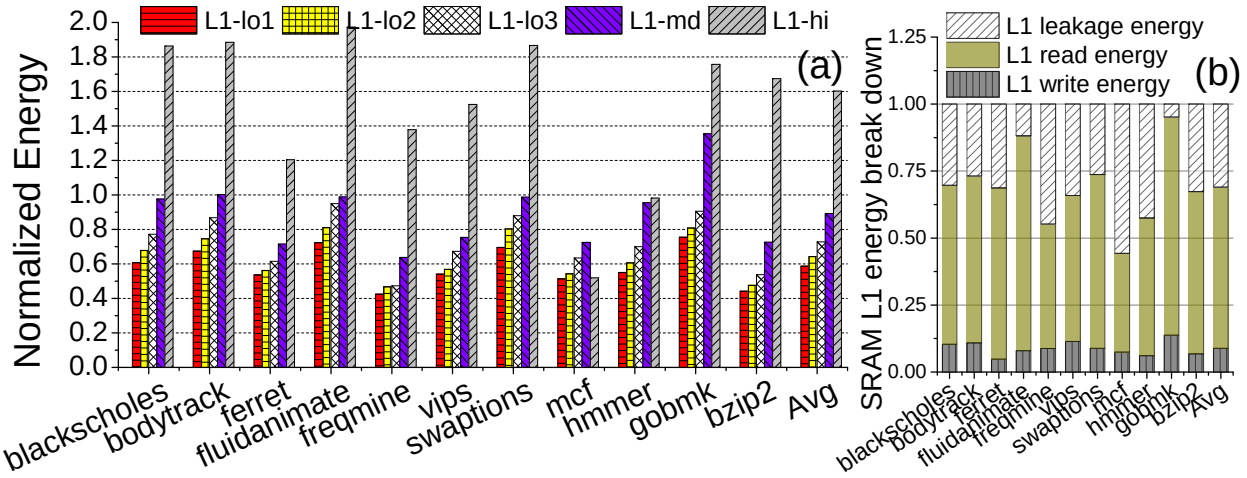


Figure 14: (a) L1 cache overall energy comparison. The energy consumptions are normalized to SRAM baseline. (b) The breakdowns of energy consumption in SRAM-based L1 cache design.

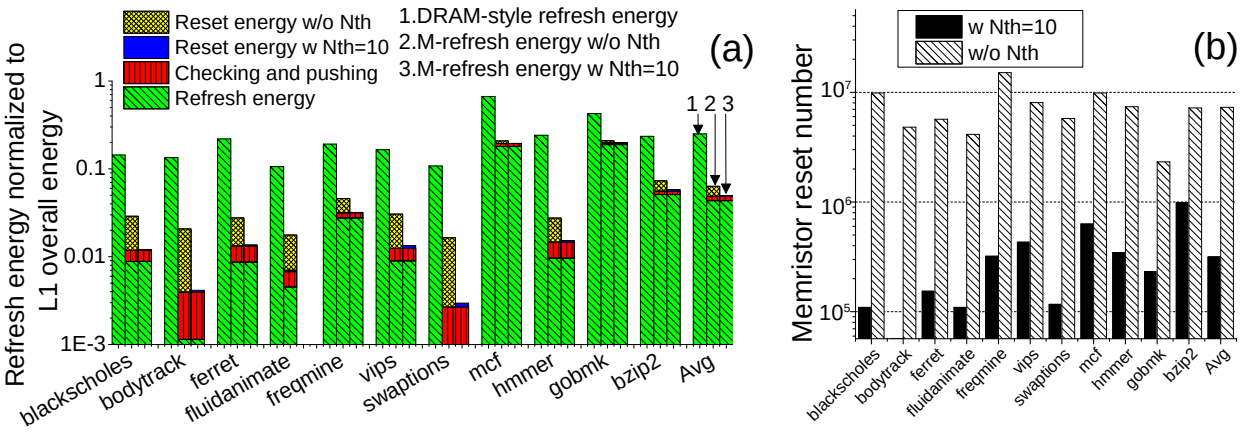


Figure 15: (a) Refresh energy comparison of the different refresh schemes. (b) The number of counter reset operations in the refresh schemes without reset threshold N_{th} and with $N_{\text{th}} = 10$.

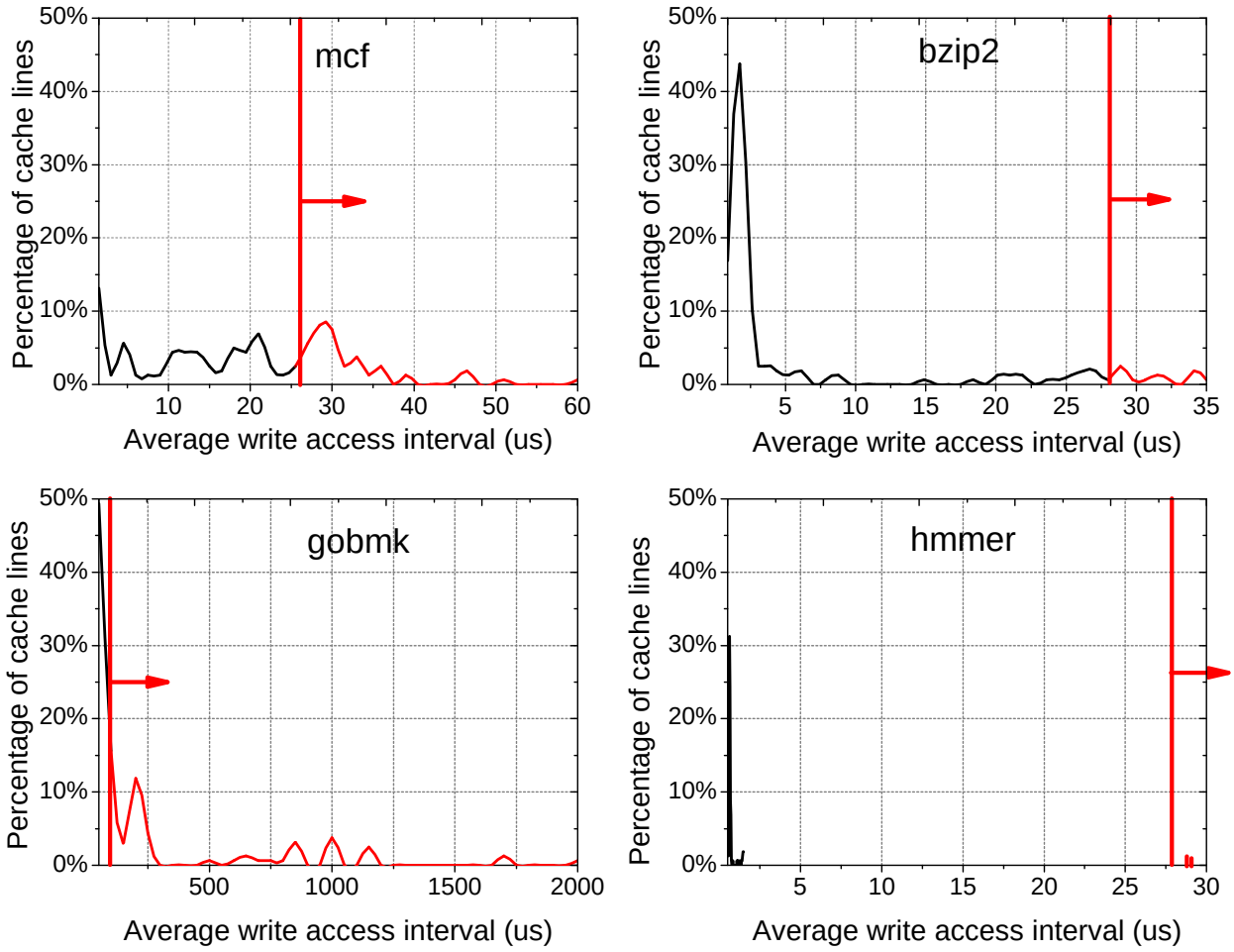


Figure 16: Cache write access distributions of the selected benchmarks.

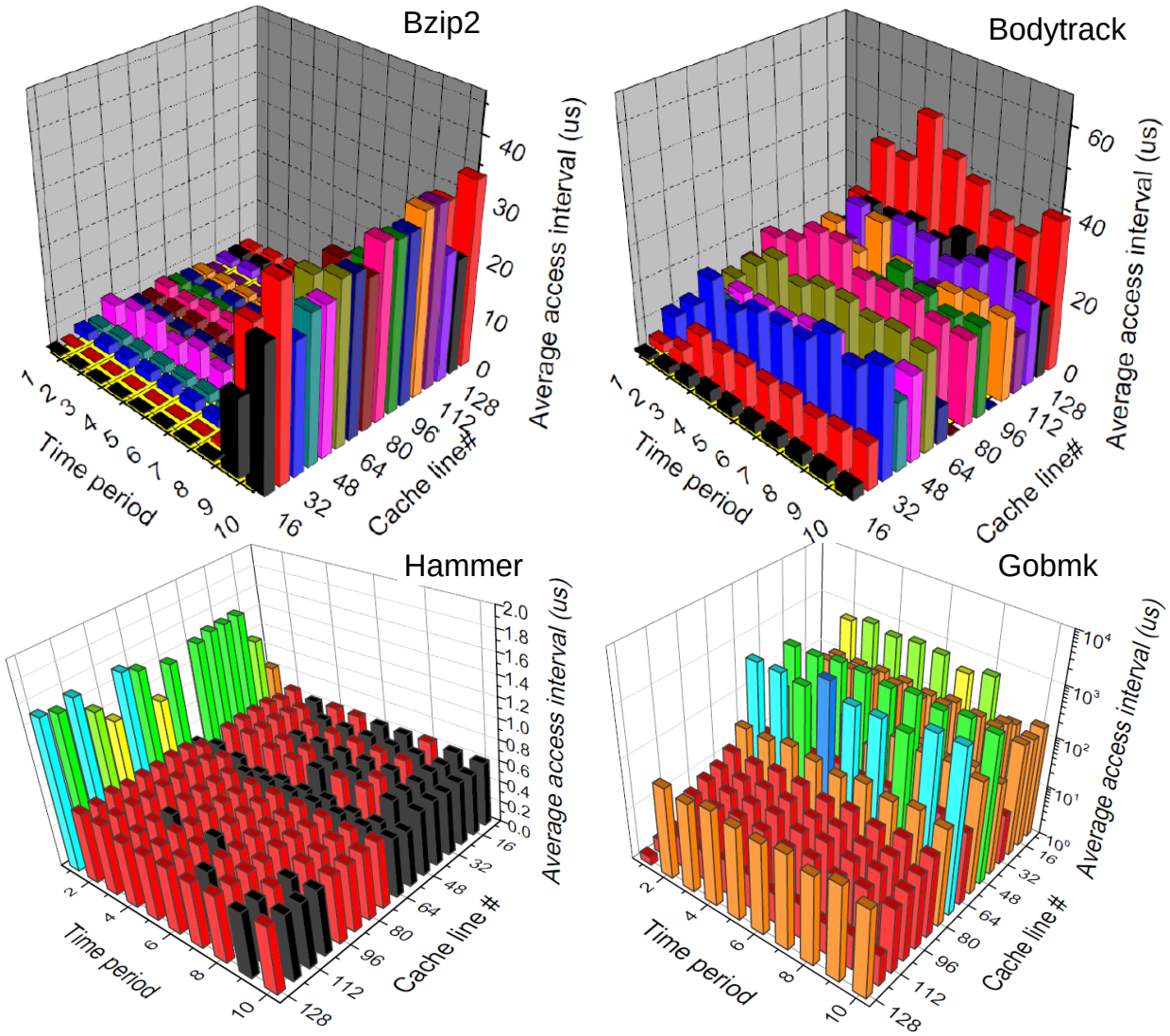


Figure 17: Cache write access intensities of different cache lines.

3.3.3 Evaluating the hybrid cache design in 2-level cache hierarchy

First, we evaluate the proposed hybrid cache design within L2 cache in 2-level cache hierarchies. In comparing the different L2 cache designs, we fixed the L1 cache to the 'L1-lo2' design. In our proposed hybrid L2 cache, way 0 assumes the 'L2-lo' design for the best read latency and the smallest leakage power among all three low retention STT-RAM designs. Ways 1 to 15 are implemented using the 'L2-md1', 'L2-md2', or 'L2-md3' (all "Opt1" MTJ designs) because a 3.24s retention time is good enough for most applications, and they have the minimal refresh overhead. The three resultant configurations are labeled as 'L2-Hyb1', 'L2-Hyb2', and 'L2-Hyb3', respectively. We compare our hybrid L2 cache with the single retention level STT-RAM design of [20] and the *read/write aware high performance architecture* (RWHCA) of [44], and label them as 'L2-SMNGS' and 'L2-RWHCA', respectively. For 'L2-SMNGS', we assumed that the L2 cache uses 'L2-md1' because its cell area of $22F^2$ is compatible to the $19F^2$ one reported in [20]. Instead of using 'L2-hi' in ways 1 to 15, 'L2-RWHCA' uses 'L2-md2' as it has an access latency that is similar to the one assumed in [44] but a much lower energy consumption. Except for Hybrid, all other L2 STT-RAM schemes use the simple DRAM refresh when refresh is needed. To be consistent with the previous section, we normalize the simulation results to the all-SRAM design.

Figure 18 compares the normalized IPC results of the different L2 cache designs. As expected, the regular STT-RAM L2 cache with 'L2-hi' design shows the worst performance among all the configurations, especially for benchmarks with high L1 miss rates, and L2 write frequencies (such as `mcf` and `swaptions`). Using relaxed retention STT-RAM design 'L2-SMNGS' improves performance but on the average it still suffers 6% degradation compared to the all-SRAM baseline due to its longer write latency. Among the three hybrid schemes we proposed, 'L2-Hyb1' is comparable in performance (99.8% on average) to the all-SRAM cache design. As we prolong the MTJ switching time by reducing STT-RAM cell size in 'L2-Hyb2' and 'L2-Hyb3', IPC performance suffers.

Figure 19(a) compares the write access numbers in HR- and LR-regions in hybrid L2 cache. Some benchmarks, such as `mcf` and `freqmine`, have a large amount of write accesses falling into HR-region, resulting in significant IPC performance degradation. In the con-

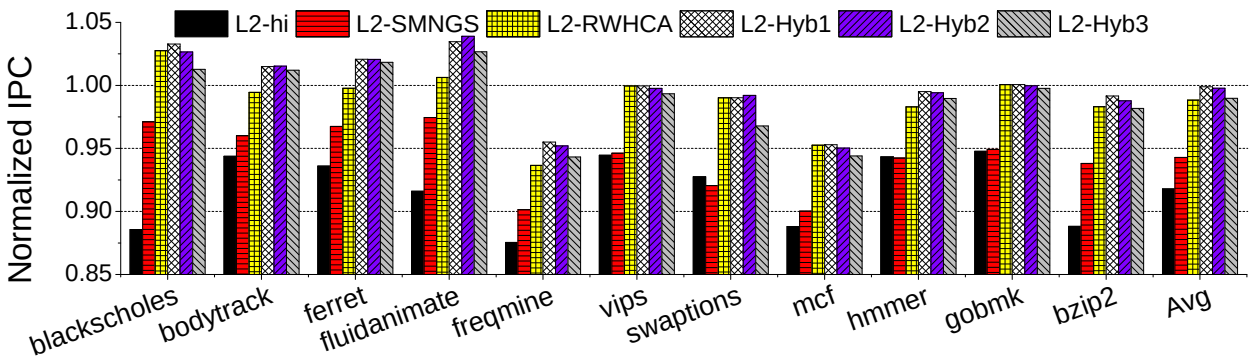


Figure 18: Performance comparison of different 2-level cache designs. The IPC's are normalized to all-SRAM baseline.

trast, other programs such as `bodytrack` and `ferret` obtain IPC improvement compared to all-SRAM baseline, which mainly benefits from the less L2 write accesses. Although `blacksholes` sends more data to HR-region than `bodytrack` and `ferret`, it has low chances to swap data between HR- and LR-regions and to write data back to main memory, as shown in Figure 19(b) and (c), respectively. So the performance of `blacksholes` also improves. In summary, all our hybrid L2 caches outperform both 'L2-SMNGS' and 'L2-RWHCA' due to their lower read latencies.

Since the savings in leakage energy by using STT-RAM designs in the L2 cache is well established, we compared the *dynamic* energy consumptions of different L2 cache designs. The energy overheads of the data refresh in LR-region, and the data migration between LR- and HR-regions in our hybrid L2 caches are included in the dynamic energy. Due to the lower write energy in the LR-region, 'L2-Hyb1' has the lowest dynamic energy consumption, as shown in Figure 20(a). As the STT-RAM cell size is reduced, the write latency and write energy consumption increased. Thus, the corresponding dynamic energy of 'L2-Hyb2' and 'L2-Hyb3' grow rapidly. Figure 20(b) shows the leakage energy comparison. Compared to 'L2-RWHCA' which is a combination of SRAM/STT-RAM [44], all the other configurations have much lower leakage energy consumptions. 'L2-hi', 'L2-SMNGS', and 'L2-Hyb1' have

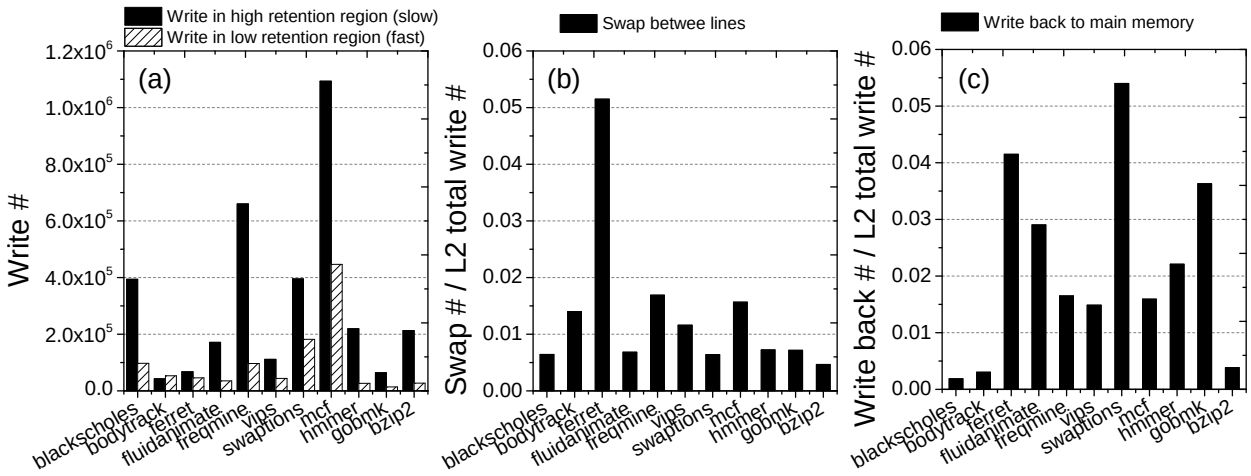


Figure 19: The hybrid L2 cache statistics. (a) The write access numbers in HR- and LR-regions. (b) The ratio of data swaps between HR- and LR-regions among all the L2 accesses. (c) The ratio of data writing back to main memory among all the L2 accesses.

similar leakage energies because their memory array sizes are quite close to each other. However, 'L2-Hyb2' and 'L2-Hyb3' benefit from their much smaller memory cell size.

The overall cache energy consumptions of all the simulated cache configurations are summarized in Figure 21. On the average, 'L2-Hyb2' and 'L2-Hyb3' consumes about 70% of the energy of 'L2-SMNGS', and 26.2% of 'L2-RWHCA'. In summary, our proposed hybrid scheme outperforms the previous techniques in [20] and [44] both in terms of performance, and (by an even bigger margin) total energy.

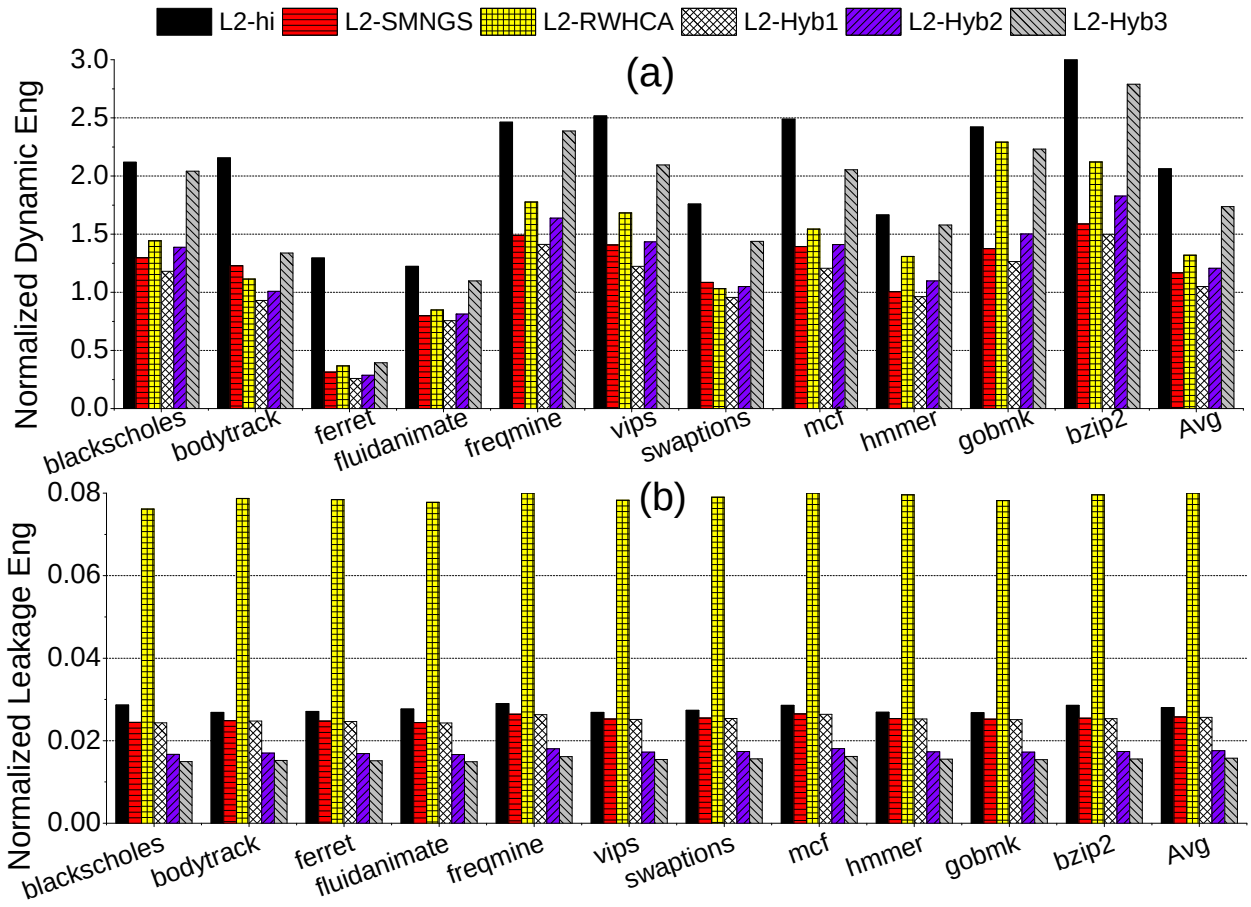


Figure 20: Dynamic and leakage energy comparison of L2 cache (normalized to SRAM baseline).

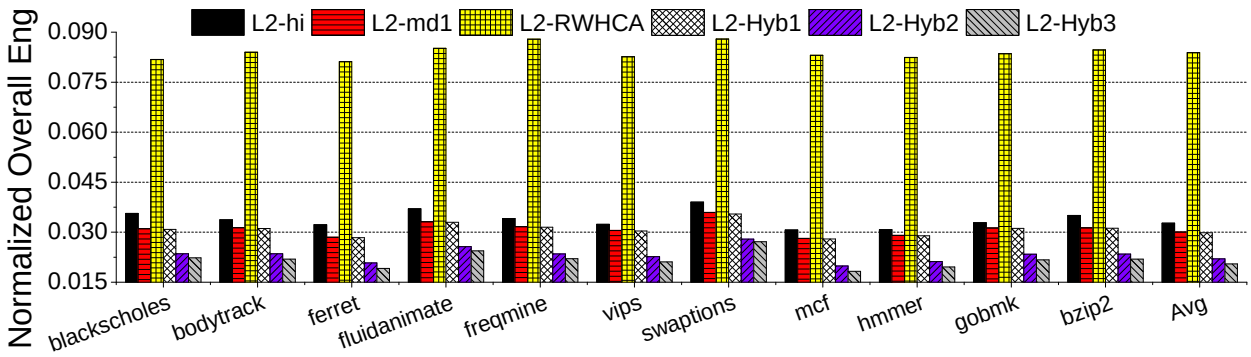


Figure 21: Overall cache energy consumption comparison of 2-L cache designs. (Normalized to the all-SRAM design).

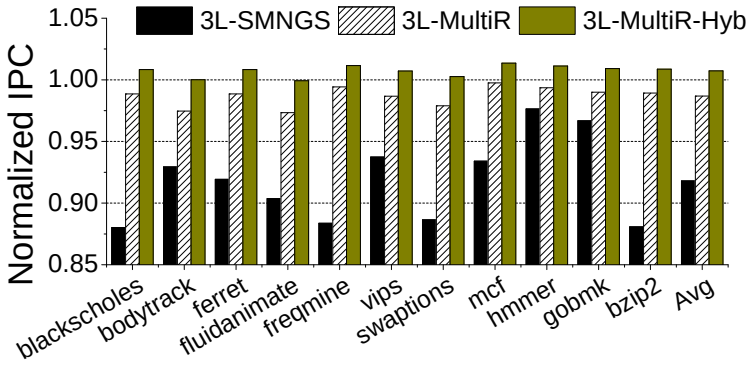


Figure 22: Performance comparison of different 3-L cache designs. The IPC's are normalized to all-SRAM baseline.

3.3.4 Deployment in 3-level Cache Hierarchies

We also evaluate four 3-level cache designs whose parameters were given in Table 5. These designs are:

The all SRAM cache hierarchy;

'3L-SMNGS' that uses the "md1" STT-RAM design in all the three level of caches, just like 'L2-SMNGS' [20];

'3L-MultiR'– a multi retention 3-level STT-RAM cache hierarchy with "L1-lo2" , "L2-md2" and "L3-hi";

'3L-MultiR-Hyb' – a multi retention 3-level STT-RAM cache hierarchy with "L1-lo2", as well as the proposed hybrid cache design used in both L2 and L3 caches. Here, 'Hyb1' is used in L2 cache for the performance purpose, while 'Hyb2' is used in L3 cache to minimize the leakage energy.

In [20], the IPC performance degradations for using the single retention STT-RAM ('md1') were from 1% to 9% when compared to an all-SRAM design. Our simulation result of '3L-SMNGS' (8% performance degradation on average) matches this well. Comparatively, the average IPC performance degradation of '3L-MultiR' is only 1.4% on average, as shown in Figure 22. The performance gain of '3L-MultiR' over '3L-SMNGS' comes mainly from "L1-

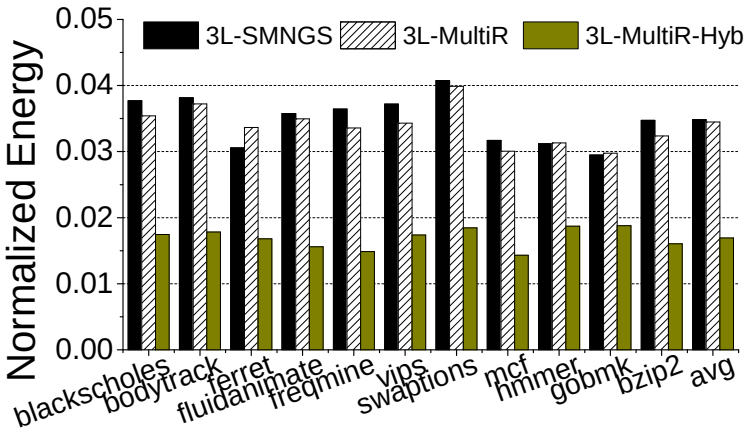


Figure 23: Overall cache energy consumption comparison 3-L cache designs. (Normalized to the all-SRAM design.)

lo2". '3L-MultiR-Hyb' has the best performance which is on average 8.8% and 2.1% better than '3L-SMNGS' and '3L-MultiR', respectively. Most of the write access in L2 and L3 cache of '3L-MultiR-Hyb' are allocated into the fast region, boosting up the system performance. Under the joint effort of "L1-lo2" and hybrid lower level cache, '3L-MultiR-Hyb' can even achieve a slightly higher IPC can all-SRAM design.

Normalized against an all-SRAM 3-level cache design, the overall energy comparison of 3-level cache hierarchy is shown in Figure 23. All three combinations with STT-RAM save significantly more energy when compared to the all-SRAM design. '3L-MultiR' saves slightly more overall energy compared to '3L-SMNGS' because the 'Lo" STT-RAM cell design has a lower per bit access dynamic energy than the "md" design. In '3L-MultiR-Hyb', shared L3 cache which embedded "md2" is much larger than local L2 cache which uses "md1". Thereby, the leakage of L3 dominates the overall energy consumption. The leakage power ratio between "md2" and "hi" is 69.1/110 (see Table 5). This is why the overall energy of '3L-MultiR-Hyb' is only 60% of '3L-MultiR' whose L3 is "hi".

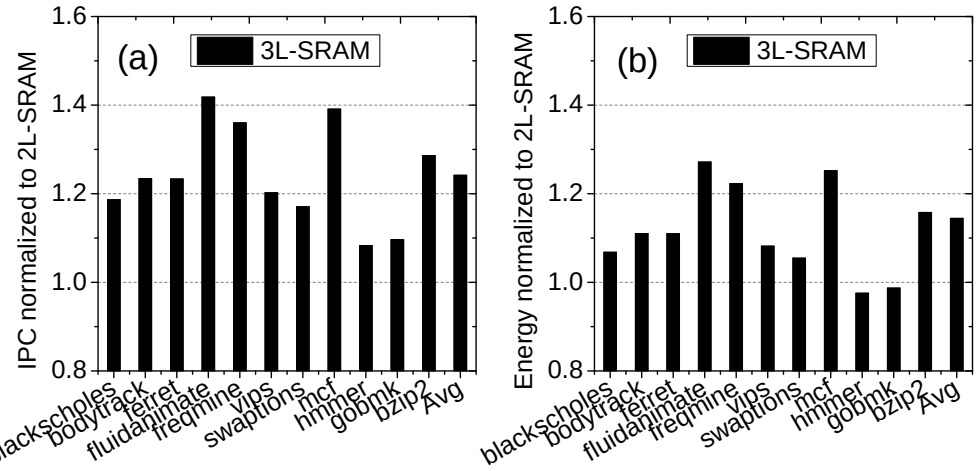


Figure 24: IPC and Overall cache energy comparison between 2-L and 3-L SRAM cache designs. (Normalized to the 2-L SRAM design.)

3.3.5 Comparison between 2-level and 3-level cache hierarchies

First, we directly compare 2-level and 3-level cache both implemented by SRAM. Figure 24(a) shows the IPC comparison. The 3-level SRAM cache outperforms the 2-level SRAM cache by 24.2% in IPC performance because the 3-level cache hierarchy include $256KB$ private L2 cache within each core, enlarging the cache capacity by 32%. Accordingly, the leakage energy increases. Figure 24(b) compares their overall energy consumptions. The total energy of 3-level SRAM cache is 14.4% greater than that of 2-level SRAM cache. The 2-level cache hierarchy with hybrid LR- and HR-regions ('2L-Hybrid') is compared with the 3-level multi-retention STT-RAM cache hierarchy ('3L-MuliR'). With regards to IPC performance, '2L-Hybrid' is 14.36% worse than '3L-MuliR', as shown in Figure 25(a). Compared to SRAM based cache, that is to say the hybrid design actually shrink the performance degradation between 2-level and 3-level cache hierarchies. On one hand, since the leakage energy of STT-RAM cell is very small, the leakage energy increasing has a much smaller scalar than the growth of cache capacity. On the other hand, the access to L3 cache is filtered by L2 cache, which induces a smaller dynamic energy in '3L-MuliR' than that of '2L-Hybrid'. So the overall energy of '3L-MuliR' is not increased as 3L SRAM does. The overall energy comparison between '2L-Hybrid' and '3L-MuliR' is shown in Figure 25(b).

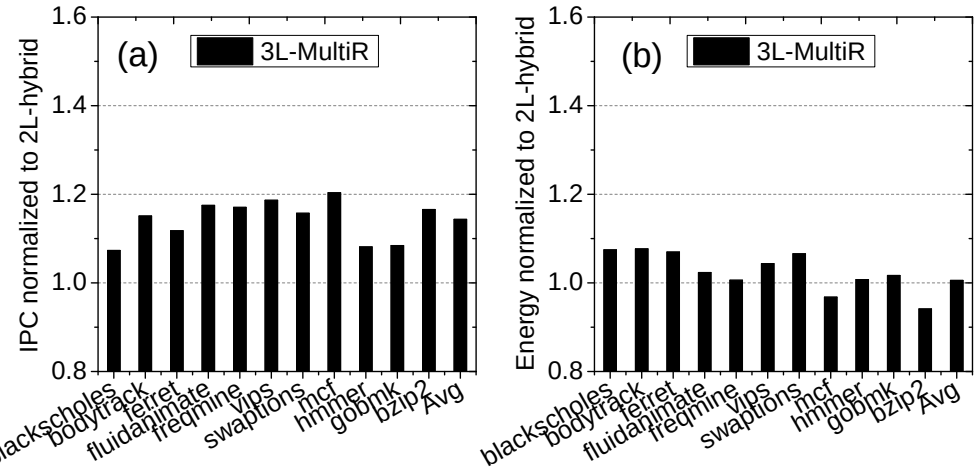


Figure 25: IPC and Overall cache energy comparison between 2-L and 3-L STT cache designs. (Normalized to the 2-L Hybrid STT design.)

3.4 SUMMARY OF CHAPTER

In this chapter, we proposed a multi retention level STT-RAM cache hierarchy that trades off the STT-RAM cell's nonvolatility for energy saving and performance improvement. Taking into consideration the differences in data access behavior, we proposed a low retention L1 cache with a counter-controlled refresh scheme, and a hybrid structure for lower level cache with both low- and high-retention portions. For L2, a data migration scheme between the low- and the high-retention portions of the cache yielded fast average write latency, and low standby power.

The experimental results show that the multi retention level STT-RAM hierarchy achieves on average a 74.2% energy reduction over the SRAM/STT-RAM mixed design, while maintaining a nearly identical IPC performance. Compared with the previous single-level relaxed retention STT-RAM design, we obtained a 5.5% performance improvement, and a 30% overall energy reduction by having multiple retention levels in 2-level hierarchy. The multi retention STT-RAM cache with proposed hybrid STT-RAM lower level cache achieves on average of 6.2% performance improvement and 41.2% energy saving compared to the previous single-level relaxed retention STT-RAM design for a 3-level cache hierarchy. Compared to traditional SRAM L1 cache, the L1 cache with a ultra low retention STT-RAM augmented by the proposed refresh scheme can achieve a 9.2% performance improvement, and a 30.3% energy saving.

The proposed refresh scheme with memristors as its retention counters can reduce 80% refresh energy compared to simple DRAM-style refresh. The memristor based counter consumes similar energy as SRAM based counter does, but can provide a much higher array area efficiency for STT-RAM array.

4.0 PROCESS VARIATION AWARE CACHE DATA MANAGEMENT FOR STT-RAM

In this chapter, we thoroughly analyze the impact of process variations on STT-RAM key design parameters and demonstrate its spatial distribution. Inspired by the *non-uniform cache access* (NUCA) [23] which has been used to compensate the difference in interconnect latencies determined by cache physical locations across the last-level on-chip cache. Two *process variation aware NUCA* (PVA-NUCA) techniques for large STT-RAM cache design are proposed, which include the non-uniformity of ***interconnect latencies*** and adaptively change ***programming duration***. In the *static PVA-NUCA*, a data block stays at a fixed location during its whole life time; while the *dynamic PVA-NUCA* can migrate a data block according to its access frequency. By considering the difference in terms of interconnect latency and programming time at the different physical locations, we propose two date migration algorithms: the first one gradually promotes a data block to a new location with faster access time, while the second algorithm aggressively places an access-intensive data directly into the fastest cache block.

4.1 PROCESS VARIATIONS OF STT-RAM

4.1.1 STT-RAM write performance variation

The CMOS process variations, such as the geometry variations of transistor channel length and width, the threshold voltage deviations, *etc.*, results in the driving strength non-uniformity of the NMOS transistor in a STT-RAM cell. The major MTJ device variations include: (1) the deviation of MTJ shape; (2) the variation in MgO thickness; and (3) the normally distributed localized fluctuation of magnetic anisotropy $K = M_s \cdot H_k$. The first two factors cause the variations of the MTJ resistances (R_L and R_H) and the MTJ switching current I_c induced by the discrepancy of the NMOS transistor bias condition. The third factor is an intrinsic feature of magnetic material that affects the MTJ switching threshold current I_{c0} and the magnetization stability barrier height Δ [37]. Table 6 summarizes the key STT-RAM design parameters and the ratio of standard deviation (σ) and mean value (μ) used in this work. We assume each parameter follows a Gaussian distribution.

To analyze the impact of process variations on the STT-RAM design, we conducted 5,000 Monte-Carlo simulations under Cadence Spectre environment. The STT-RAM cell with $24F^2$ was used. The nominal value of the highest possible switching current I_c of this design is $200\mu A$. Figure 26(a) is the distribution of I_c, which varies from $110\mu A$ to $320\mu A$ after including the variations of both NMOS transistor and MTJ. Correspondingly, the required MTJ switching time τ in Figure 26(b) demonstrates a very large distribution from $2.3ns$ to $12.4ns$. The traditional worst-corner design methodology using a uniform write period, *i.e.*, $\tau_{max} = 12.4ns$, will result in a pessimistic design and severe degradation on STT-RAM cache performance. Furthermore, ***the variation of MTJ switching time τ has exceeded the difference in the interconnect latencies***, which is less than $8ns$ for a 32MB STT-RAM based on CACTI [74].

The results in Figure 26(c) shows the minimum retention time is $0.01s$, which corresponds to 20 million clock cycles in a system with $2GHz$ clock frequency. In the following chapter, we assume a DRAM-like refreshing scheme [20] with a refresh period of $0.01s$. The overhead on system performance and energy consumption has been considered in evaluations.

4.1.2 Write performance distribution map

Figure 27 illustrates the study object of this work – an 8MB shared STT-RAM L2 cache stacked on top of an 8-core CPU layer by using 3D technology. The shared L2 cache consists of 32 banks. The signal transportation between two layers is realized by *through-silicon-vias* (TSV).

We generated a distribution map of the required write pulse periods of the 8MB LLC by using the multi-level quad-tree approach [22] and combining the statistical results from Monte-Carlo simulations. The distribution map is also shown in Figure 27. The process variations present strong spatial correlations, that is, the STT-RAM cells close to each other have smaller difference in terms of the required write pulse periods than those far apart.

Table 6: Process variations in STT-RAM design.

Parameters	Mean (μ)	σ/μ.
NMOS transistor width (W_{tx})	$180nm$	10% (1)
NMOS transistor length (L_{tx})	$45nm$	10% (1)
NMOS transistor threshold voltage (V_{th})	$0.3V$	10% (1)
MTJ resistance area product (RA)	$20\Omega \cdot \mu m^2$	8% (2)
Cross section area of MTJ free layer (A)	$90nm \times 180nm$	5% (2)
MTJ magnetization stability energy height (Δ)	22	27% (3)

Data sources: (1)[80], (2)estimation based on Matlab simulation.

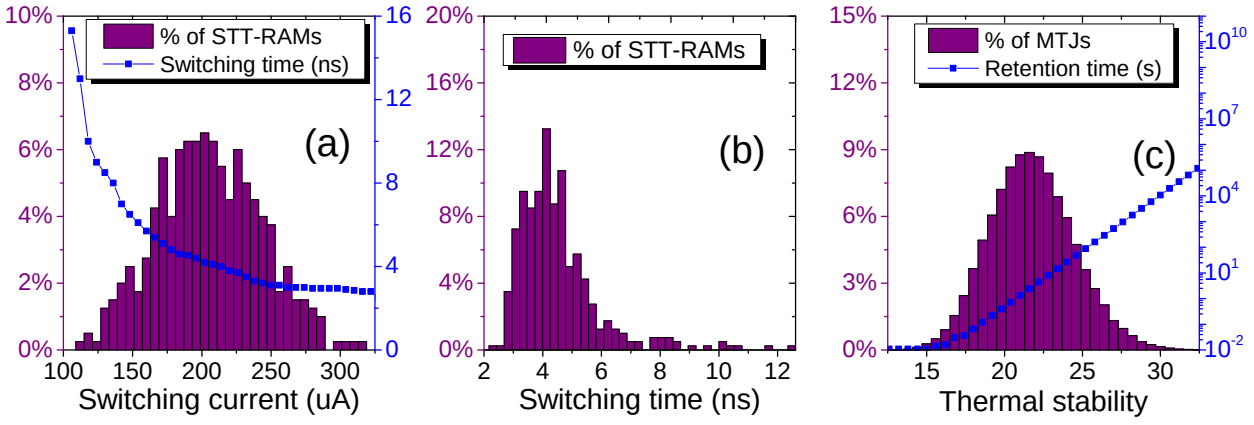


Figure 26: The distribution of (a) MTJ switching current I_c, (b) writing pulse width τ, and (c) MTJ data retention time.

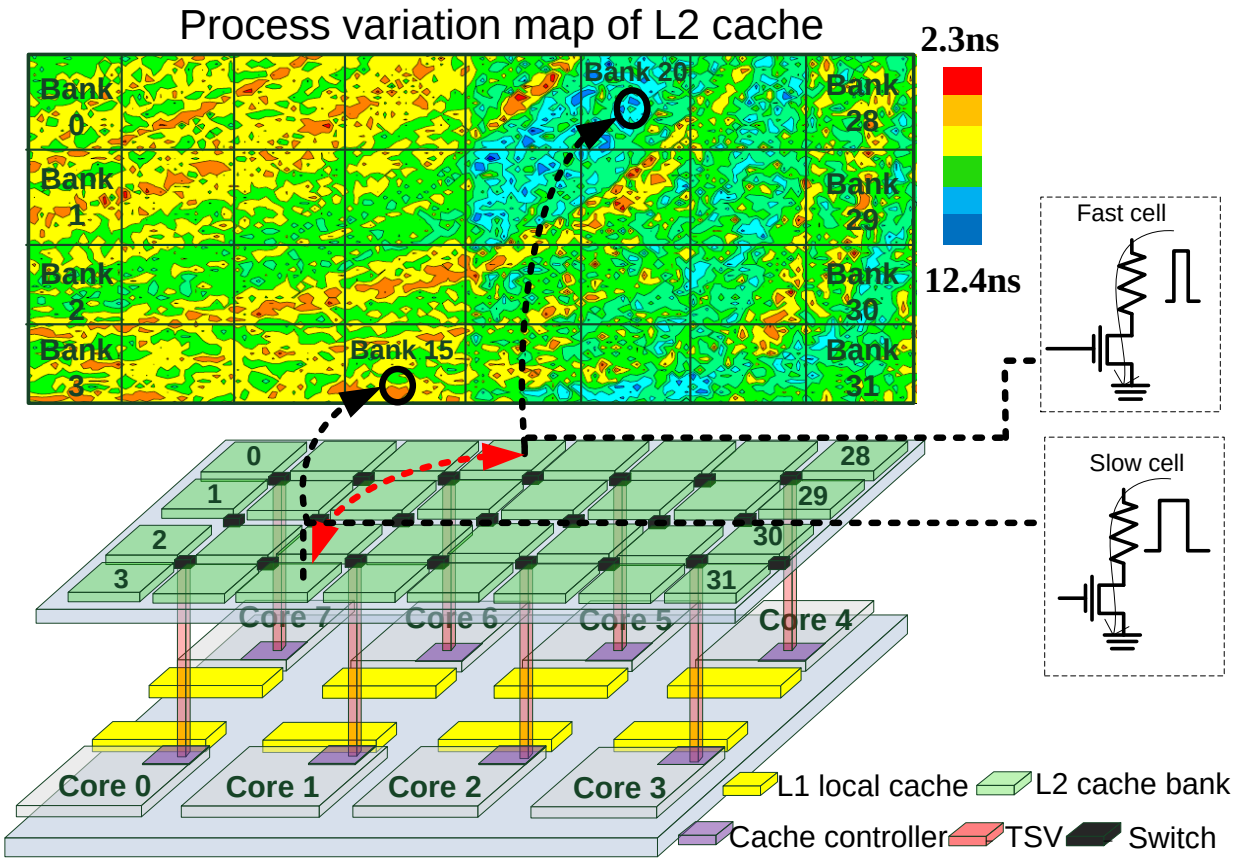


Figure 27: The write pulse period distribution map of a shared STT-RAM L2 cache in 3D stacking structure.

4.2 PROCESS VARIATION AWARE NUCA

4.2.1 DPVA-NUCA-1

The *last level caches* (LLC) occupies a big potion of on-chip area. The different core to bank distances and the induced discrepancy of the interconnect lengths make significant variations of cache access latency. Therefore, the *dynamic non-uniform cache access* (DNUCA) has been used to migrate data block from distant bank to nearby one and reduce transmission overhead [23]. Previously, Sun *et al.* extended NUCA to STT-RAM design [10] by separating and adopting the worst-case read and write latencies, *i.e.*, $lat_{c,r,max}$ and $lat_{c,w,max}$, respectively.

Although the STT-RAM read operation is small and not significantly affected by process variations, the distribution of write latencies are large and could be more severe than the discrepancy induced by the different core to bank distances. For example, Section 4.1 shows that the required STT-RAM write latency $lat_{c,w}$ changes from $2.3ns$ to $12.4ns$ across an 8MB STT-RAM cache, while the latency of peripheral circuitry and interconnect lat_{wire} of the same design varies between $0.5ns$ and $8ns$, based on CACTI [74].

Hence, a *process variation aware NUCA* (PVA-NUCA) with adaptive $lat_{c,w}$ is necessary for STT-RAM cache design to accelerate cache performance and reduce write energy consumption. In this chapter, we will present and discuss one static and two dynamic versions of PVA-NUCA, namely, SPVA-NUCA, DPVA-NUCA-1, and DPVA-NUCA-2, respectively.

4.2.2 SPVA-NUCA

SPVA-NUCA is a static process variation aware NUCA for STT-RAM design without data migration. Here, a read operation includes the different core to bank distances but uses a fixed, maximum cell read latency $lat_{c,r,max}$. On the contrary, a write access latency is determined by both the physical location and the required write time of the cache block. The overall latency of a cache access then can be expressed as:

$$\text{Read access}: \quad Lat_R = lat_{c,r,max} + lat_{wire}; \tag{4.1}$$

$$\text{Write access}: \quad Lat_W = lat_{c,w} + lat_{wire}. \tag{4.2}$$

As illustrated in Figure 27, the required write time can be obtained in post-fabrication testing and stored within the LLC data storage array. For the 8MB STT-RAM cache with a $2GHz$ clock frequency in Section 4.1, a cache block requires 5 extra bits to represent the corresponding $lat_{c,w}$. Considering the spatial correlation, a few cache blocks in close locations (*e.g.*, 64 blocks) can utilize the maximum $lat_{c,w}$ among them to reduce design overhead.

The dynamic process variation aware NUCA, DPVA-NUCA-1, uses the *conservative promotion* algorithm (see Algorithm 1) for cache block migration. The data promotion/demotion is determined by the frequencies of write hits and write misses: a write-intensive cache block will be promoted to a physical location with the minimum Lat_W.

On each write hit, the Lat_W of 32 ways within the accessing set will be sorted. Based on the sorting results, the data block will be swapped with the data block who is located in the cache block with next smaller Lat_W as the pairs of black and red arrows shown in Figure 28(a). Naturally, the non-write-intensive data blocks will be gradually demoted to cache block with larger Lat_W as shown by the red arrows. During a write miss, the *least recent used* (LRU) data block will be evicted and replaced by incoming request.

Without loss of generality, the example in Figure 28(a) demonstrates how a data block is gradually promoted. Assuming the data block is placed at *bank 31* which is the farthest cache block with longest programming time at very beginning. In this example, it takes N swaps for the the data block to promote from *bank 31* with maximum Lat_W to *bank 4* with minimal Lat_W. Cache block in *bank 28* and *bank 29* also have shortest programming time within this accessing set, but their interconnection latencies is too large. Minimum $lat_{c,w}$ doesn't mean Lat_W, so the data won't be moved to *bank 28* or *bank 29* finally. Moreover, the data block won't be swapped to the cache block in *bank 0*, which has smallest lat_{wire} but a larger $lat_{c,w}$. The sorting of Lat_W of 32 ways within the accessing set every time is too costly to be affordable. We use a **sorting recorder** to record sorting results of recent accessed set. The more hits in the **sorting recorder**, the least performance degradation caused by the sorting of Lat_W will be. The efficiency of **sorting recorder** will be evaluated in the results part.

4.2.3 DPVA-NUCA-2

The date migration in DPVA-NUCA-1 is determined by only write accesses. Note that many programs are read access dominant. Moreover, the read access latency variation with its physical location is mainly determined by the difference of lat_{wire} since $lat_{c,r}$ is much smaller and uniform in different banks. Thus, DPVA-NUCA-1 solely based on write accesses might not obtain the best system performance and the least energy consumption. We proposed another dynamic process variation aware NUCA (DPVA-NUCA-2) to balance the read and write requests. Moreover, data blocks are aggressively move to fastest cache block according to the access situation to further improve the performance.

Algorithm 2 is an *aggresive promotion* algorithm used for DPVA-NUCA-2. The data blocks are classified into three types: *read intensive*, *write intensive*, and *not access intensive*. We use two queues, namely, *read prediction queue* and *write prediction queue* to record the read and write access histories, respectively. An incoming read or write hit triggers the address searching in both read queue and write queue. (a) If the block address is in the read queue and its corresponding counter is above threshold, the block is read intensive. We will move it to the cache block with the minimal read access latency Lat_R, as the dash black arrow illustrated in Figure 28(b). (b) Otherwise, if the data is not read intensive, but we can find it in write queue with a counter value beyond threshold, the data is write intensive. We will move it toward the cache block with the minimal write access latency Lat_W, as the solid black arrow illustrated in Figure 28(b). (c) If the data is neither read intensive nor write intensive, no data migration is needed. The proposed DPVA-NUCA-2 grants higher priority to read accesses than that of write ones. This is because usually read is more frequent and more critical to system performance. So, if a data block is both read and write intensive, the data migration will be determined by Lat_R. LRU block will be replaced during write or read miss. **Sorting recorder** is needed by DPVA-NUCA-2.

Write and read prediction queues: The two prediction queues have the same hardware structure – have a repository to store recent accessed block address and associated with corresponding counter. An incoming access can trigger the increment of the corresponding counter if the accessing block is already in the queue. Otherwise, a new entry will be added

if the queue is not full. If the queue is full, the LRU entry will be evicted. As we shall show in Section 7.3, a small number of entries, *i.e.*, 32, is efficient enough for read and write queues. The queue is CAM structure, so the searching of the queue is quick and within 1 clock cycle.

Conflict reduction: For DPVA-NUCA-2, it is possible that two or more cores compete for one cache block location with minimum Lat_W. In such a situation, the write-intensive data blocks held by these two cores may take each other replaced alternatively, resulting in unexpected cache misses. To reduce those conflicts, we can simply attach a counter to record its cache miss numbers. If the miss numbers of several cores increase simultaneously, we check if there are conflict of write intensive blocks in their write queues. If it happens to be the scenario, some data will be moved to block with the second minimum Lat_W in a round robin manner. If the miss rate keep on high, the data will be further demoted to cache block with larger Lat_W.

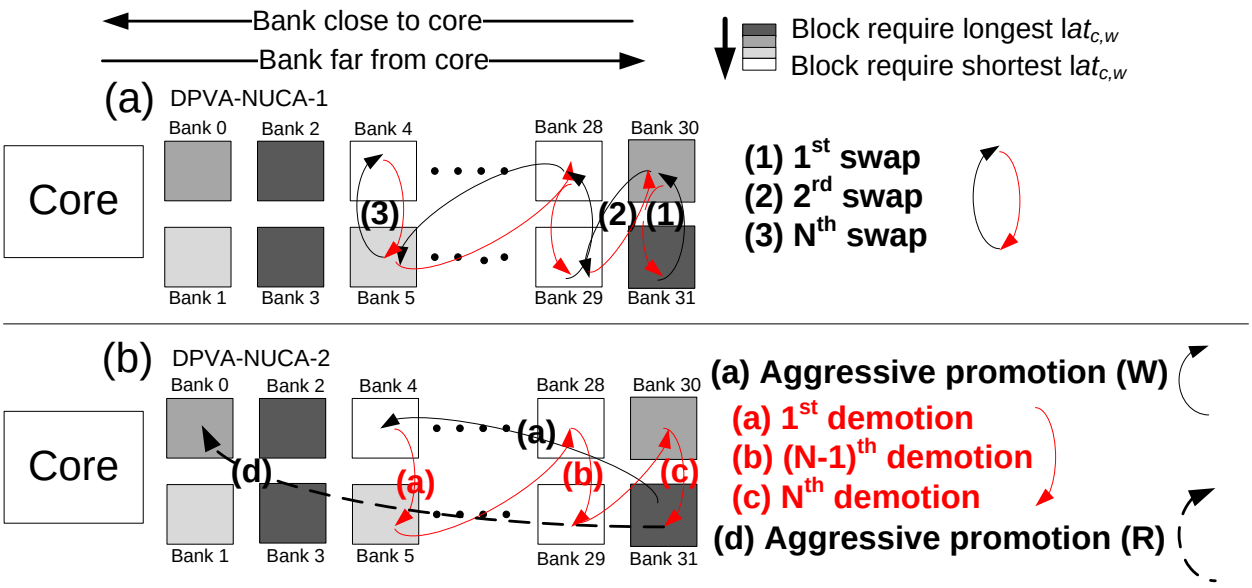


Figure 28: The illustrations of data migration in (a) DPVA-NUCA-1 (b) DPVA-NUCA-2.

Definitions:
$CB(Lat_{W,min})$: the cache block with the minimal write latency.
$CB(Lat_{R,min})$: the cache block with the minimal read latency.
$CB(Lat_{W,next})$: the cache block with the second smallest write latency.

Algorithm 1: **Conservative Promotion** used in DPVA-NUCA-1.

```
1  if write(hit)
2     if  data is not in CB(Lat_{W,min}) then
3        swap data in CB(Lat_W) with the one in CB(Lat_{W,next})
4     end if
5  end if
6  if write(miss) then
7     replace the LRU data block
8  end if
```

Algorithm 2: **Aggressive Prediction** for DPVA-NUCA-2.

```
1  if hit
2     if data is read intensive block then
3        move data into CB(Lat_{R,min})
4     else if data is write intensive block  then
5        move data into CB(Lat_{W,min})
6     end if
7  if write(hit)
8     if access address is in the write prediction queue then
9        corresponding queue counter ++
10    else then
11       if queue is full then
12        kick LRU entry and fill the entry with incoming address
13       else then
14        add new entry with incoming address
15       end if
16    end if
17 if read(hit) then
18       do exact same operation in read prediction queue
19    end if
20 end if
21 if miss then
22       replace the LRU data block
23 end if
```

4.3 SIMULATION RESULTS DISCUSSION

4.3.1 Experimental setup

Our basic architecture configuration is an 8-core in-order UltraSPARC T1 processor where the SRAM L2 cache is replaced by STT-RAM. We use the Simics toolset [81] for performance simulations. The details on microarchitecture configurations are depicted in Table 12. UCA, SUNCA, DUNCA and our proposed PVA-NUCA are implemented by modifying the cache model in Simics. We adopted the multi-threaded benchmarks from Parsec Benchmark Suite [82] in our architectural simulations. For each benchmark, we fast-forward to ROI (Region of Interest), warm up the cache for 200 million instructions, and then run 500 million instructions.

The system architecture is illustrated in Figure 27. We use an 8-core in-order UltraSPARC T1 processor where the shared SRAM L2 cache is replaced by STT-RAM cache. The shared L2 cache can be stacked on the CPU core layer, benefiting from 3D integration technology. The inter-layer signals are connected by *through-silicon-vias* (TSVs). Each CPU core is connected with the on-chip switch which has four neighboring L2 cache banks. Cache controller is used to deal with the communication and coherency between L1 and L2 cache, as well as cores.

Table 7: Processor configurations.

Processors:	8 cores, 2GHz, 4 threads/CPU core, 1-way issue (in order)
SRAM L1 Cache:	local, 16KB I/D, 2-way, 64B line, 2-cycle, write-through
STT-RAM L2 Cache:	shared, 8MB, 32 banks, 32-way, 64B line, write-back, 1 read/write port, 4 write buffers.
Main Memory:	4GB, 400-cycle latency.

4.3.2 Cache access statistics

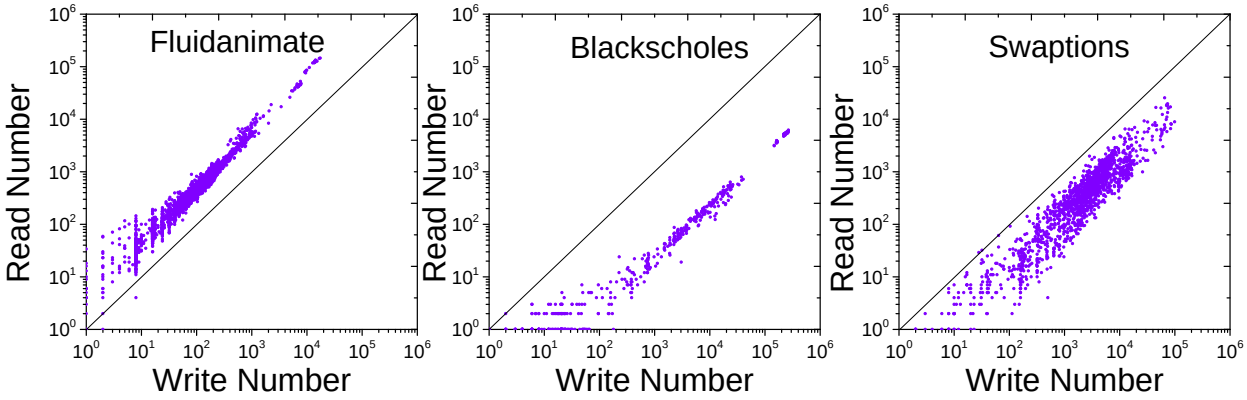


Figure 29: Read and write number of all cache blocks.

Figure 29 compares the overall read and write numbers of three selected benchmarks representing different read and write access distributions. Data was collected without applying any data migration algorithm. Comparably, `Blacksholes` is more write intensive and `Fluidanimate` is more read intensive. Write and read numbers of `Swaptions` are more decentralized. The $x-$axis and $y-$axis are in logarithmic scale. The read and write number distributions of data blocks will affect the hardware optimization efficiencies in the proposed schemes such as conflict reduction, prediction queues and sorting recorder.

Figure 30 demonstrates a more detail write and read access analysis. According to the write access numbers in 500 million instructions, we divide cache blocks into four groups. For example, the first bar in each benchmark of Figure 30(a) represents the number of cache blocks with zero write access, while the last bar shows the number of cache blocks with more than 100 write accesses. Figure 30(b) shows the percentage of writes that fall into those blocks with more than 100 write access. We conducted the similar statistical analysis for read accesses and show in Figure 30(c) and (d). Figure 30 explicitly indicates that a small number of frequently accessed cache blocks hold most of the write and read operations. The goal of the proposed schemes is to allocate these data blocks into fast cache blocks to improve system performance.

4.3.3 Conflict reduction

As discussed in Section 4.2.3, the conflicts of access-intensive data blocks from different CPU cores could severely degrade overall system performance of DPVA-NUCA-2. Figure 31(a) demonstrates the simulated conflict rate for various benchmarks. Here, the conflict rate is defined as the ratio of the conflict number over the sum up of read hits and write hits. Compared to ideal case without calculating the impact of conflict, the corresponding IPC degradation caused by those conflicts are shown in Figure 31(b). Some benchmarks, such as `Blacksholes` and `Bodytrack`, have strong data locality. The chance to have data confliction is low, which leads to close to zero IPC degradation. However, some benchmarks such as `Fluidanimate` demonstrate high conflictions from different CPU cores and unaffordable IPC degradation. Both conflict rates and IPC degradation have been drastically reduced after utilizing the conflict-reduction mechanism in DPVA-NUCA-2.

The conflict rate and the corresponding IPC degradation could be affected by the prediction threshold. The prediction threshold is the preset threshold of the counter of the prediction queue. Generally, increasing prediction threshold can reduce the number of cache blocks swapped into the fast cache blocks, and hence, decrease the conflict rate. However, the results of `Fluidanimate`, `x264` and `Swaptions` in Figure 32 indicate that merely increasing the prediction threshold will not help improving the IPC degradation, even though the conflict rate decreases. That is because less data blocks are allocated into the fast cache blocks with the leveling up of the prediction threshold. On the contrary, our proposed conflict-reduction mechanism can overcome this difficulty because it dynamically and gradually move write intensive data block away from the fastest block based on the cache miss rate.

4.3.4 Hardware exploration

In DPVA-NUCA-2, we use read and write predction queues to determine if a data block is a read or write intensive block. Intuitively, the larger the queues are, the more access-intensive data block can be allocated in the queue. Figure 33 shows the ratio of read and write hits captured by the queues. Though increasing queue entries can help allocate more accesses

in the fast cache blocks, it also generates more conflicts. So keep continuously enlarging the prediction queues does not always help benefit system performance eventually. Our simulation results show that when the entry number of the prediction queues is 32, IPC stops growing even the data access ratio in the fast cache blocks continuously increases.

Similarly we explore the optimal size of sorting recorder. Figure 34 shows the sorting recorder hit ratio and IPC degradation when varying sorting recorder entry number. When the sorting recorder has 32 entries, the hit ratio is less than 60% for all the benchmarks. The IPC degradation is significant due to the respectively reading out sorting information and sorting Lat_W. As entry number increases to 64, the sorting recorder can obtain $> 99\%$ of hit ratio for most of benchmarks. The only exception is `Swatpions`, because the recent accessed addresses are frequently replaced by new requests due to its disperse access pattern. Further increasing sorting recorder size does not much benefit on IPC improvement.

Figure 35 shows the sorting recorder hit ratio with time. A small sorting recorder demonstrates a random hit ratio pattern as shown in Figure 35(a). A more stable pattern can be observed as the size of sorting recorder increases. Hit ratio is low at beginning of the simulation when all the entries are empty. As the incoming request gradually fill the sorting recorder, the hit ratio climbing up to approach 100% except for `Swatpions` who has a decentralized read/write number distribution as shown in Figure 29.

4.3.5 Performance, energy and hardware

According to hardware exploration results in Section 4.3.4,we adopt the prediction queue with 32 entries and sorting recorder with 128 entries as the optimal configuration. In Figure 36, the IPC performance of different schemes have been evaluated. All the results are normalized against UCA with consideration of process variation. Because data migration is not introduced in SPVA-NUCA which maximize the effect of the dynamic cell programming latency, so `Bodytrack`, `Blacksholes` and `Swaptions` are more write intensive so that the gaps between SPVA-NUCA and DNUCA as well as SNUCA are enlarged. `Ferret` and `Fluidanimate` have more read intensive data blocks but less write intensive data blocks. Thereby, the their IPCs of SPVA-NUCA don't have too much improvement over DNUCA

and SNUCA. The data migration overhead has already been considered in both DPVA-NUCA-1 and DPVA-NUCA-2. The DPVA-NUCA-1 and DPVA-NUCA-2 achieve 21.70% and 29.29% IPC improvement over STT-RAM based DNUCA [10], respectively.

Figure 37 compares the dynamic write energy and the overall energy of three proposed schemes. Since UCA and SNUCA use the worst-case switching time, both dynamic energy and leakage energy of them are obviously higher than the proposed schemes. So we don't show them here. All proposed schemes are compared with DNUCA in terms of energy. Among the three proposed schemes, DPVA-NUCA-1 consumes the highest dynamic write energy because each write hit involve one swap with neighboring cache block. The dynamic write energy of DPVA-NUCA-2 is much lower since it avoids most of the data swaps as DPVA-NUCA-1 does. Compared to DPVA-NUCA-1, DPVA-NUCA-2 saves on average 65% of dynamic write energy and 12.4% of overall STT-RAM cache energy. Since DPVA-NUCA-1 is better than SPVA-NUCA in terms of performance, its leakage is lower than that of SPVA-NUCA. Although dynamic write energy of DPVA-NUCA-1 is much higher than that of SPVA-NUCA, the overall energies of DPVA-NUCA-1 and SPVA-NUCA are very closing on average.

Compared to 8MB ($8 \times 1024 \times 1024 \times 8$ bits) LLC, the hardware overhead including latency indicator for each block (5 bits/block), sorting recorder (maximum 128 entries $\times$ 32 $\times$ 32 bits), read/write prediction queue (2×32 entries $\times$ 40 bits) will be less than 1%.

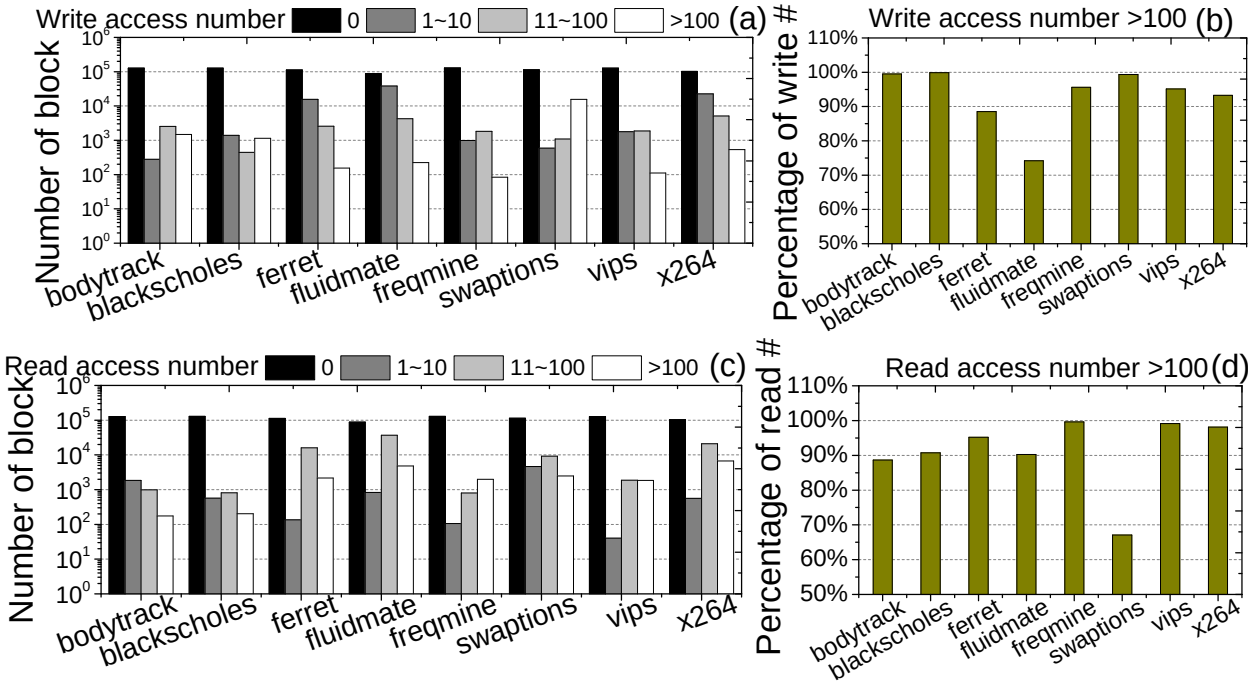


Figure 30: Number of blocks hold different (a) write or (c) read numbers; Percentage of (b) write or (d) read held by access intensive blocks

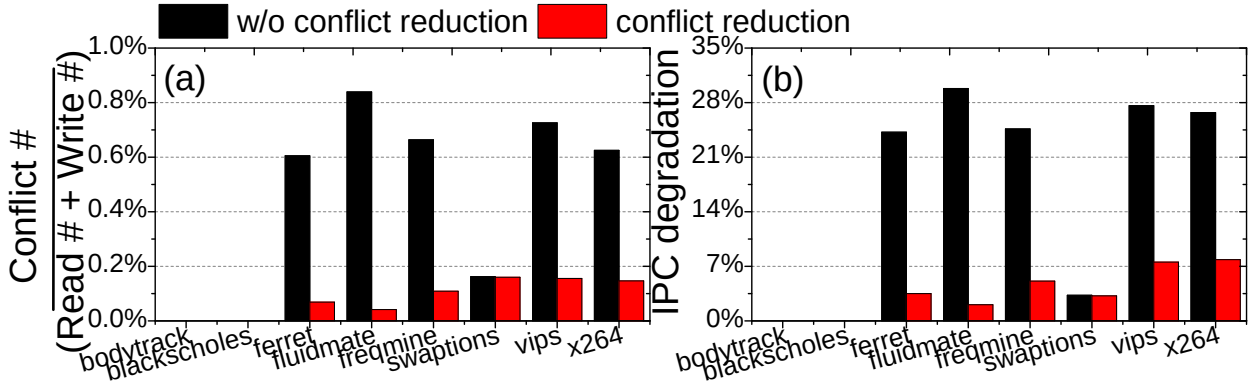


Figure 31: (a) Data conflict rate and (b) IPC degradation before and after conflict reduction technique.

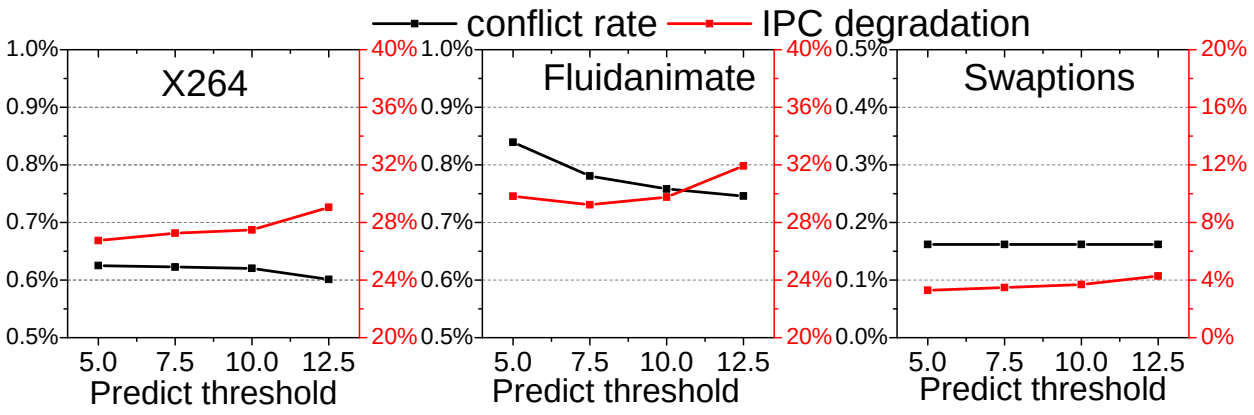


Figure 32: (a) Data conflict rate and (b) IPC degradation with different prediction threshold.

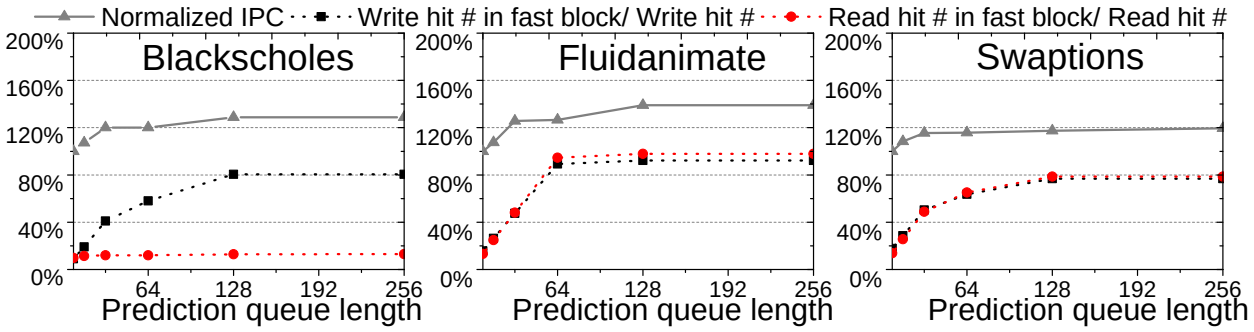


Figure 33: Normalized IPC and write/read ratio in fast cache blocks when varying prediction queue length.

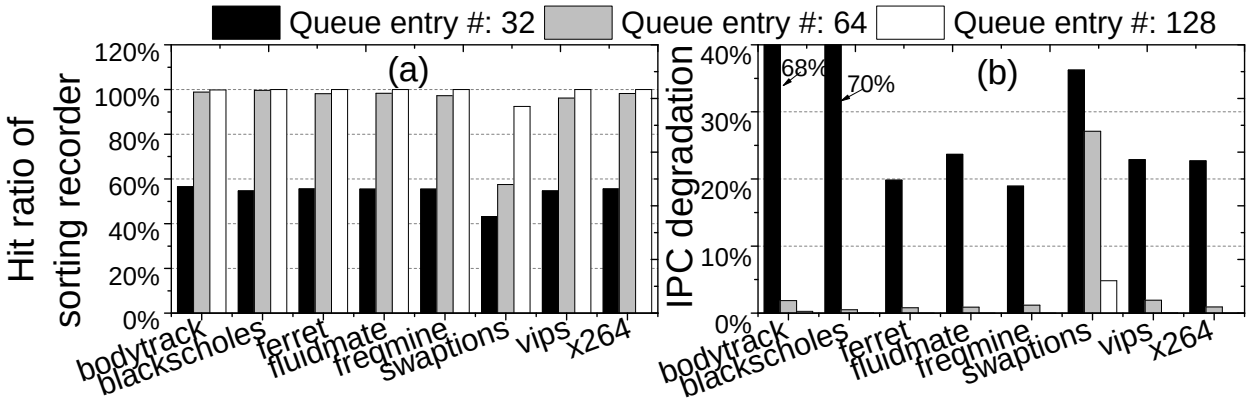


Figure 34: Sorting recorder hit ratio and IPC degradation when varying entry number.

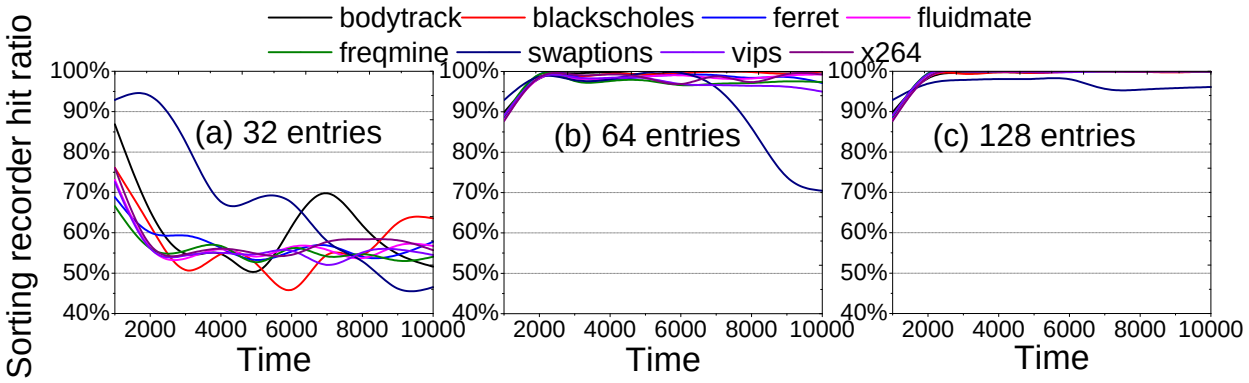


Figure 35: Sorting recorder hit ratio with time of different entry number.

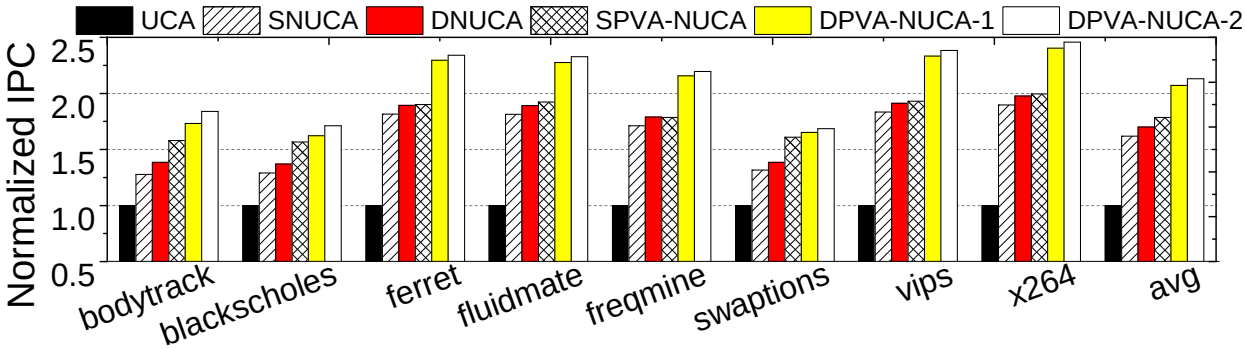


Figure 36: Normalized IPC of different schemes.

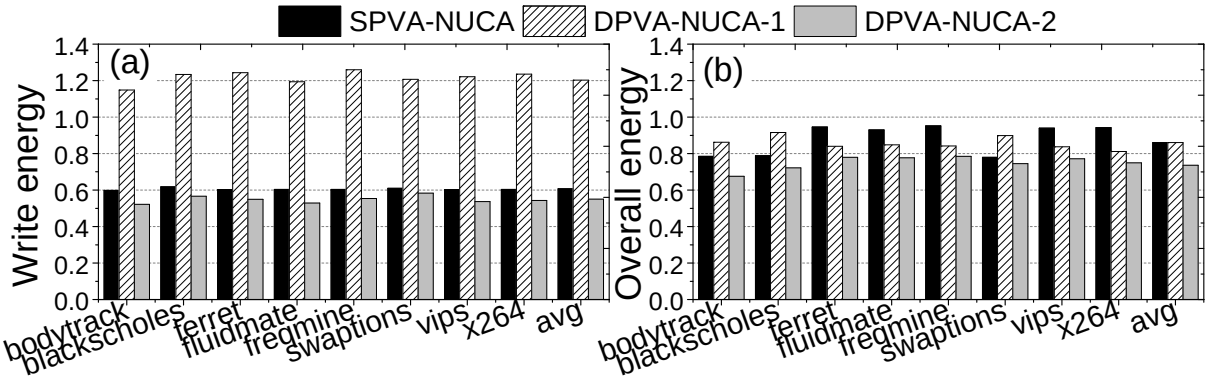


Figure 37: Energy comparison. (a) Dynamic write energy; (b) Overall cache energy.

4.4 SUMMARY OF CHAPTER

Process variation aware non-uniform cache access schemes have been proposed for the retention-relaxed STT-RAM last level cache. By considering non-uniformity of both interconnection latency and cell programming time, all the proposed schemes can achieve better performance and lower energy consumption than previous DNUCA and SNUCA. Among the proposed designs, DPVA-NUCA-2 with conflict reduction techniques can achieve 25.29% IPC improvement and 26.40% overall energy saving when compared to DNUCA.

5.0 DUAL-MODE ARCHITECTURE FOR FAST STT-RAM

The write performance of STT-RAM has obtained significant improvement, benefiting from the better understanding on magnetic device physics and the enhanced process engineering [83][84]. Sub-nanosecond switching of in-plane *magnetic tunneling junction* (MTJ), the data storage element of STT-RAM, has been reported [49]. Sacrificing the nonvolatility is another effective way to reduce the MTJ switching time [85][20]. Moreover, the novel perpendicular MTJ technology [86] has demonstrated faster switching and better scalability than the conventional in-plane MTJ. The STT-RAM design based on these *fast-switching* (FS) devices will be the focus of this work.

On the other hand, the importance of read operation, which is not as trivial as what people previously thought of, has been brought out. The stored data of STT-RAM is read out by detecting the MTJ resistance state. The large process variations degrade the resistance differences between the data cell and the reference cell [24]. Consequently, the sensing delay of sense amplifier is enlarged. The situation in *STT-RAM design with fast-switching devices* (called as FS-STT-RAM) is more severe: the improved write performance requires the read current amplitude to decrease accordingly in order to prevent the unintentional data switching, or *read disturbance* (RD). As we shall show in Section 5.1, the sub-nanosecond read speed under small read current that has been widely used is not true anymore.

Based on the comprehensive cross-layer (device-circuit-architecture) analysis, we first build the relationship amongst the read access latency, read disturbance probability, and the system performance. A novel FS-STT-RAM based memory architecture then is proposed, which switches between the *high accuracy* and the *low power* modes with the support of operating system. Thus, the speed, energy, and data reliability of the overall computing system can be prioritized and balanced according to users' requirement.

5.1 READ DISTURBANCE AND READ PERFORMANCE

5.1.1 Read Disturbance

The magnetization switching of the free layer in a MTJ is a stochastic procedure. Applying a current through an MTJ in a read operation can potentially result in unintentional data flipping, which is called as *read disturbance* (RD). The read disturbance probability P_{rd} of a MTJ at a read current I_R is determined by the critical switching current I_{c0}, the read current pulse width τ_{rd}, and the thermal stability Δ. In theory, P_{rd} can be expressed as [87]:

$$P_{rd} = 1 - exp\{-\frac{\tau_{rd}}{\tau_0} exp[-\Delta(1 - \frac{I_R}{I_{c0}})]\}, \tag{5.1}$$

where, τ_0 is the thermally activated reversal time, which is usually assumed to be $1ns$ [20]. The MTJ device parameters such as I_{c0} and Δ are determined by the fabrication process and remain unchanged after a device is made. Therefore, the read disturbance probability P_{rd} indeed is a function of I_R/I_{c0}, under a given τ_{rd}.

In read operations, τ_{rd} should be long enough so that the sense amplifier can complete the data detection. In other words, τ_{rd} is determined by the sense amplifier performance. Prolonging τ_{rd} significantly increases the possibility of read disturbance as shown in Figure. 38(a).

Under the same operating condition, *i.e.*, the same I_R and τ_{rd}, FS-STT-RAM suffers from a much higher read disturbance probability than Conv-STT-RAM. For example, Smullen and Sun *et al.* proposed to reduce the MTJ thermal stability for better switching performance [85][20]. Correspondingly, P_{rd} increases as Figure. 38(b) shows. Moreover, the smaller I_{c0} of FS-STT-RAM causes a higher I_R/I_{c0} ratio and aggravates the possibility of the unintentional MTJ flipping.

In order to reduce the design complexity of read peripheral circuitry, the read current is always applied along one single direction. As a consequence, the read disturbance is *unidirectional*. For instance, writing "0" into a STT-RAM cell requires a switching current from BL to SL (refer to Figure. 2). The read current flowing in this direction potentially induces the unwanted $1 \rightarrow 0$ flips, but the data "0" remains intact. On the contrary, supplying read current from SL to BL may disturb data "0" only.

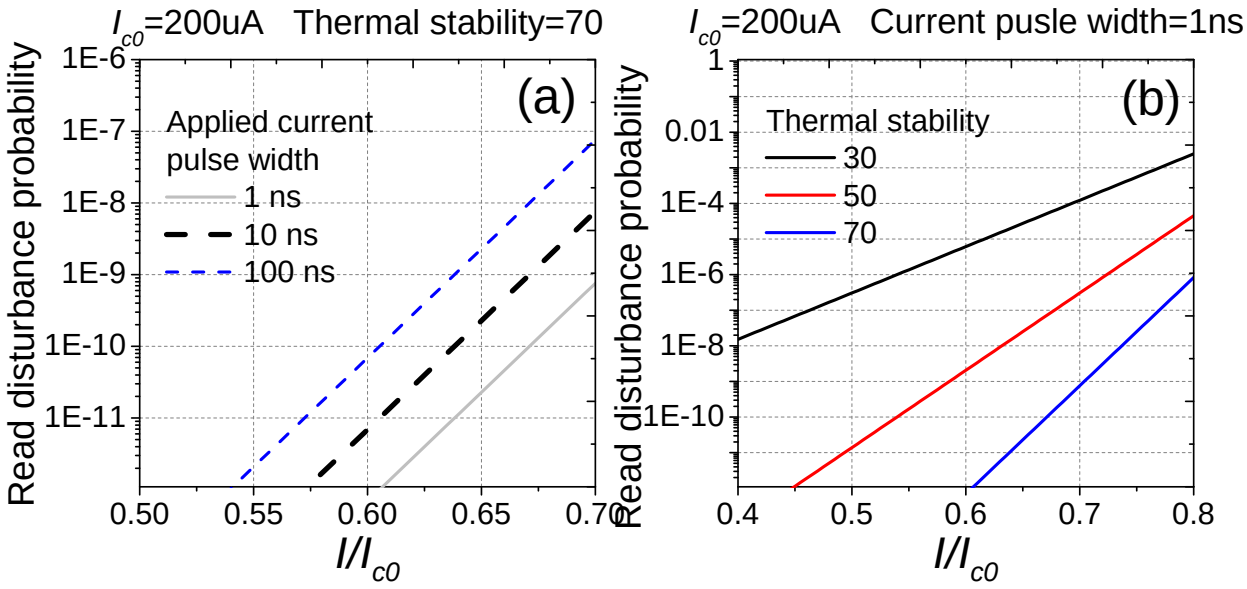


Figure 38: The MTJ read disturbance probability under (a) the different read current pulse width τ_{rd}, and (b) the different thermal stability Δ.

5.1.2 Read Current, Read Speed, and Read Disturbance

In STT-RAM design, a sense amplifier is used to detect the data stored in a target reading cell by comparing the voltage difference between its bit-line and a reference voltage. The voltage difference at the sense amplifier inputs is called as sensing margin. Given a sense amplifier design, the sensing delay is determined by the sensing margin: the smaller the sensing margin is, the longer the sensing delay is required, as shown in Figure. 39(a).

The sensing margin is proportional to the read current I_R. To keep the similar read disturbance probability and hence the induced read errors in Conv-STT-RAM and FS-STT-RAM, I_R/I_{c0} need remain unchanged, as indicated in Eq. (5.1). Compared to Conv-STT-RAM, the smaller I_{c0} of FS-STT-RAM results in a smaller I_R, which causes the smaller sensing margin and longer read time τ_{rd}. The simulation result in Figure. 39(a) shows that the sensing delay increases dramatically as the sensing margin reduces. Thus, the sub-nanosecond read speed that has been widely used is not true anymore.

On the other hand, we can increase the read current in FS-STT-RAM for better read speed. However, the corresponding higher read disturbance probability results in more unintentional data flipping, or *read errors*. The quantitative relation among the read current, sensing delay, and read disturbance error rate is shown in Figure. 39(b).

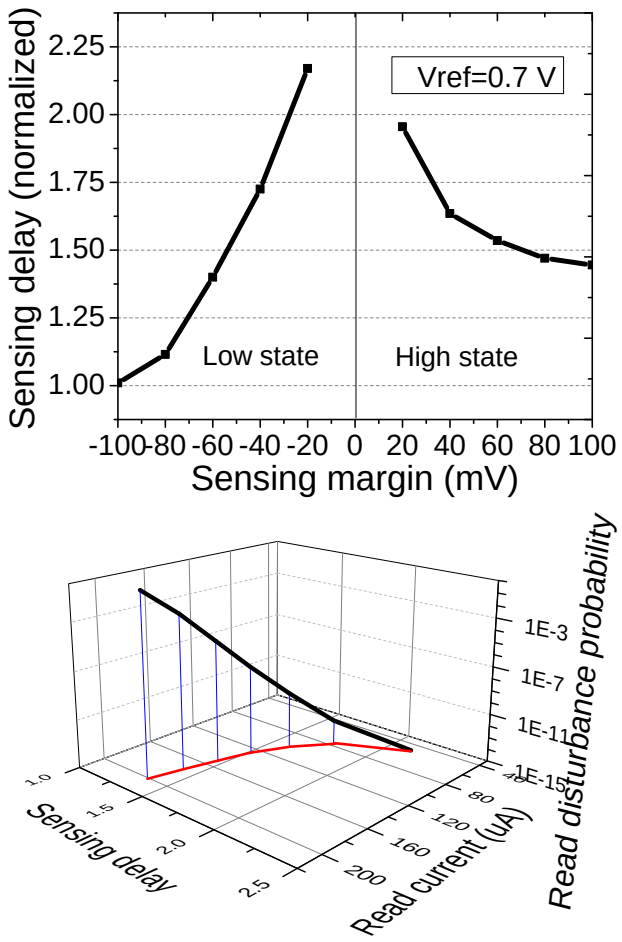


Figure 39: (a) The sensing delay vs. the sensing margin; (b) The read current determines the sensing delay and the read disturbance probability.

5.2 DUAL-MODE ARCHITECTURE FOR FS-STT-RAM

FS-STT-RAM can dramatically improve write performance and reduce the write energy consumption. But it faces severe challenge on data reliability due to the read disturbance. Reducing the read current can alleviate the situation by degrading the overall system performance, especially for the programs with high frequent reads and sensitive to the read speed. In this work, we propose a new architecture with dual operating modes for FS-STT-RAM design.

5.2.1 High accuracy mode and low power mode

A general purpose computing system need deal with various tasks with the different data accuracy and throughput demands. For example, the banking and financial transactions require 100% accurate data and relatively low throughput. In contrast, the video stream usually has a high data exchange rate but occasional erroneous pixels usually are not noticeable. Therefore, we propose a FS-STT-RAM architecture that can operate in either the *high accuracy* (HA) mode or the *low power* (LP) mode.

Figure. 40 illustrates the diagram of the proposed dual-mode architecture for FS-STT-RAM. At the beginning of a program, the system determines the STT-RAM operating mode based on the application-oriented data access requirement. Software-level assistance is needed to detect such information and to deliver to hardware as input control.

The low-power mode can be adopted for the applications with relatively low data accuracy request for high throughput and low power consumption. The read operations perform regularly as the conventional STT-RAM design. The regular *error correction code* (ECC), *e.g.*, Hamming code, is utilized that can dramatically lower the raw bit error rate. However the system does not take extra effort to fix the STT-RAM bit flips induced by read current.

For data correctness oriented applications, the FS-STT-RAM design switches to the high accuracy mode, in which every read is followed by a refreshing (rewrite), that is, writing the readout data back to the cache block, to eliminate the impact of the read disturbance. Considering that the read disturbance is unidirectional, we rewrite only those STT-RAM

cells with potential to be disturbed (for instance, data 1 when read current flows from BL to SL) and avoid unnecessary energy overhead.

5.2.2 Hardware and software interface

The proposed hardware scheme provides the support and flexility to switch operation between two modes. The mode selection can be controlled at higher system levels. The software interface shall provide users options to select their flavor of data accuracy. Programmers are able to define the required data accuracy based on the requirement of programs. Extra instructions might be inserted to enable mode switching. Compiler shall also translate and deliver the mode switching information to the hardware. On the other hand, the hardware structure must support the compiler in return. For example, one extra bit data indication operation mode is required in the memory access instructions.

5.2.3 Enhancement in high accuracy mode

We further propose to enhance the high accuracy mode and mitigate the performance and energy overheads induced by the rewrite after each read through two design techniques, namely, the *shadow write buffer* and the *write bit inverting*.

Shadow rewrite buffer: Even in FS-STT-RAM, writing a cell is more difficult and takes longer time than reading out data (refer to simulation data in Section 7.3). Moreover, the read accesses are more frequent in caches than the write accesses as the cache access statistics in Figure. 43(a) shows. Therefore, the system performance in the high accuracy mode significantly degrades compared to that of the low-power mode.

Figure. 41 illustrates an cache access example in the high accuracy mode. Assume *addr* 1 and *addr* 2 belong to the same bank, while *addr* 3 goes to a different bank. 1) the rewrite at *addr* 1 in *cycle*2 does not stall the incoming new write *addr* 3 since they go to the different cache banks; 2) in *cycle*3, the new write and the rewrite goes to the same cache line *addr* 1. Therefore, the rewrite can be throttled and only the new write need to be executed since the new write will update the value anyway; 3) the conflict happens only when the rewrite operation locates into the same bank but not same cache line, *i.e.*, *addr* 1 and *addr* 2 from

*cycle*7. At this time, the rewrite stalls the new coming writes and introduce performance overhead.

Instead of sharing the same write queue with regular write requests, we buffer the rewrites after reads in a separated shadow rewrite buffer as shown in Figure. 40(b). Rewrites have a lower priority than the new write requests and are executed when the write driver is available. During a read operation, both cache and its corresponding shadow rewrite buffer are simultaneously accessed, and the data from the latter one, if any, will be taken as the golden copy. The shadow rewrite buffer should be sufficiently large to mitigate performance degradation caused by the conflict between the incoming write and the rewrite. On the other side, it should not be too large after considering the overhead of read latency and energy consumption.

Write bit inverting: A 1T1J STT-RAM cell provides the different write currents when writing 0 and 1 due to the different biasing condition, which makes the corresponding write energies different [27]. Moreover, the read disturbance affects either data 0 or 1, determined by the read current direction. Therefore, properly selecting the read current direction could help reduce write energy consumption. For example, if writing 1 costs less energy than writing 0, the read current from BL to SL could be more preferable since it causes only $1 \rightarrow 0$ flips.

The write bit inverting scheme is proposed for the same motivation. Previously, the bit inverting technique has been used in STT-RAM [27] and PCM [88] for write energy reduction. Since read accesses are more frequent than the write accesses, the rewrite energy consumption becomes more important FS-STT-RAM operating in high accuracy mode. We utilize the write bit inverting scheme to maintain more 0's and less 1's and hence minimize the rewrite energy.

As illustrated in Figure. 42, when the number of 1's of a cache block is greater than half of the total bit number, we invert all the writing value. An additional bit is used to indicate the inverting status. Dividing a cache block into several smaller segments can help reduce the overall number of 1's. However, it also introduces more indicating bits. Our simulation results in Figure. 43(b) shows that the of write-0 bits are much greater than the number of write-1 bits. Statically the chance of bit inverting is not very high.

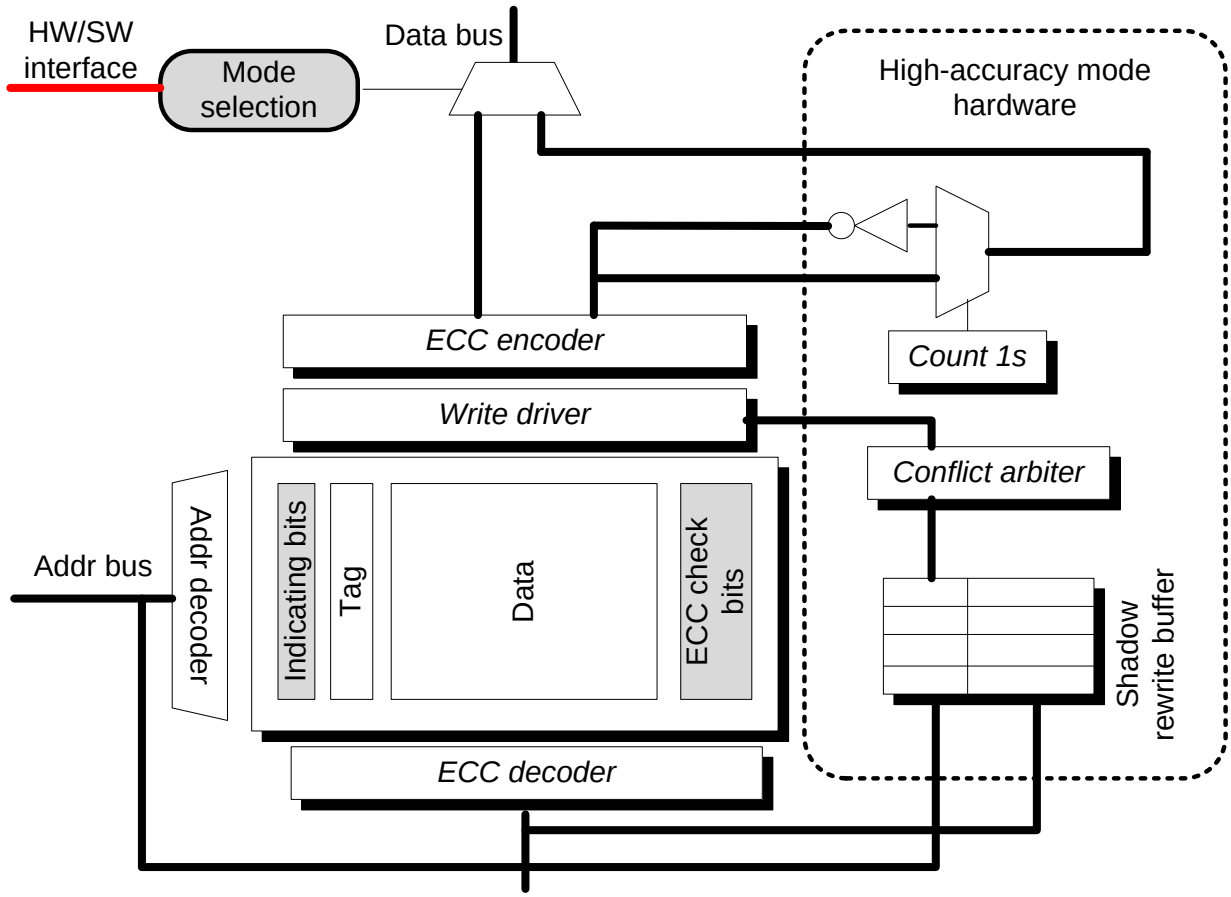


Figure 40: The proposed dual-mode architecture for FS-STT-RAM.

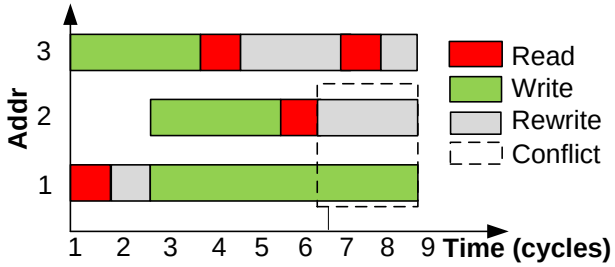


Figure 41: An cache access example in the high accuracy mode.

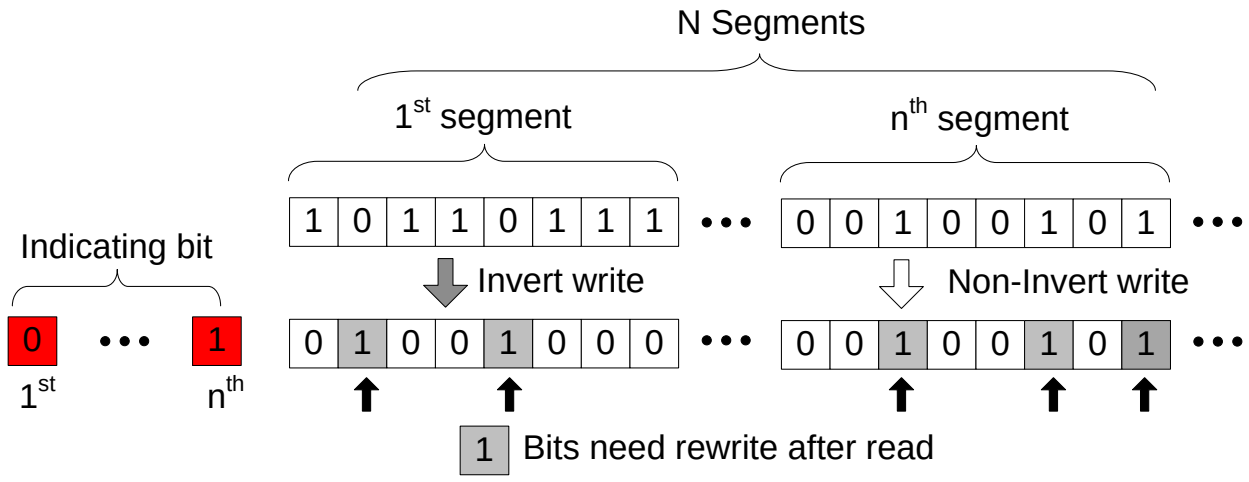


Figure 42: Write bit invert scheme.

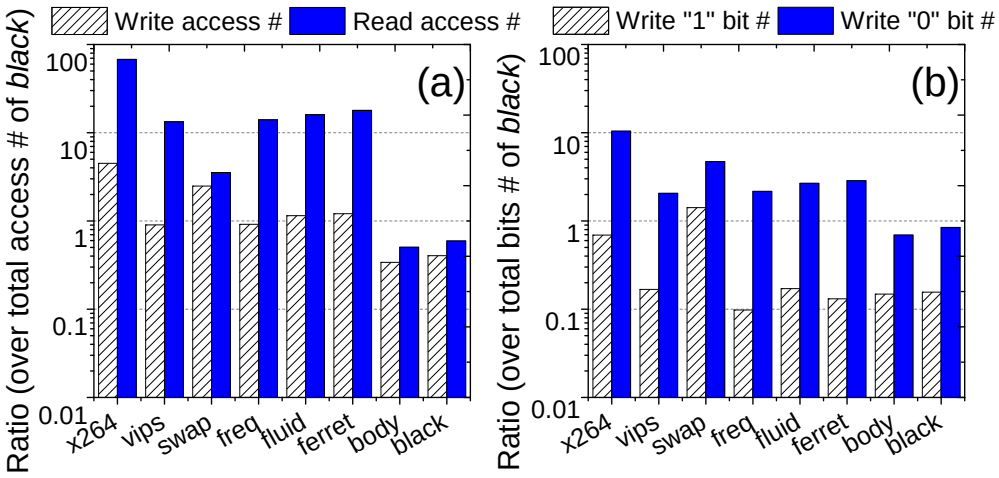


Figure 43: (a) Read access frequency vs. write frequency. (b) Write "1" bits vs. write "0" bits.

Table 8: Processor Configuration

Processors	8 cores, 2GHz, 4 threads/CPU core, 1-way issue (in order)
SRAM L1 Cache	Local, 16KB I/D, 2-way, 64B line, 2-cycle, write-back
STT-RAM L2 Cache	Shared, 4MB, 32 banks, 32-way, 64B line, write-back, 1 read/write port, 4 write buffers.
Main Memory	4GB, 400-cycle latency.

5.3 SIMULATION RESULTS

5.3.1 Evaluation of the dual mode architecture

We performed the evaluations on an 8-core in-order UltraSPARC T1 processor by replacing the SRAM L2 cache with Conv-STT-RAM or FS-STT-RAM. The Simics toolset [81] was used to obtain the cache access trace. More detail of the process configuration is depicted in Table 12. And Table 9 summarizes the cache latency and energy of both Conv-STT-RAM and FS-STT-RAM obtained from SPICE simulation and NVsim [79] calculation based on MTJ in [85]. The multi-threaded benchmarks from Parsec Benchmark Suite [82] were adopted in architecture simulations. For each benchmark, we fast-forward to region of interest, warm up the cache for 200 million instructions, and then execute 500 million instructions.

5.3.2 Read latency and read errors

Due to the large amount of read accesses, the read latency is critical to system's overall performance. In Figure. 44(a), we compare the *instructions-per-cycle* (IPC) performance of FS-STT-RAM in the low power mode by reducing the read current amplitude and hence increasing the read latency cycles. As expected, the system performance degrades significantly as read latency increases. From this perspective, the fast read operation is extremely important. However, as Figure. 44(b) shows that the data error induced by read disturbance,

Table 9: Cache Latency and Energy Configurations

Cache type	4MB FS-STT-RAM	4 MB Conv-STT-RAM
Write latency	Peripheral latency: 1.059 ns Switching latency: 5.356 ns	Peripheral latency: 1.121 ns Switching latency: 10.326 ns
Read latency	Peripheral latency: 2.119 ns Sensing latency: 1.5 ns	Peripheral latency: 2.242 ns Sensing latency: 1.5 ns
Write energy	0.932 nJ	1.916 nJ
Read energy	0.083 nJ	0.085 nJ
Leakage power	104 mW	110 mW

another critical system parameter, is also sensitive to the read current. Even integrating wtih the single-bit ECC, the error rate of the fast read operation (*i.e.*, $1.5ns$ sensing delay) is more than 10^{-12}.

5.3.3 The dual-mode architecture for FS-STT-RAM

Performance In the following evaluation, the fast read operation with $1.5ns$ sensing delay is adopted. The LP mode doesn't apply extra data recover scheme. Its data error rate is shown in Figure. 44(b). The hA mode intends to eliminate the read-induced errors but sacrifices performance due to the conflicts between write and rewrite. The performance comparison between LP and HA modes is shown in Figure. 45(a). Without integrating the shadow rewrite buffer, an average 18% of IPC degradation has been observed in HA mode compared to LP mode. By referring Figure. 45(b), we found that the workloads with more conflicts, such as `swap` suffers more on IPC degradation, while there is less impact on the workloads like `body` for its less conflicts.

The introduction of the shadow rewrite buffer can significantly reduce the performance degradation caused by the write and rewrite conflicts. Its effectiveness is related to the entry numbers. Figure. 46 shows three typical workloads with the different sensitivities to the

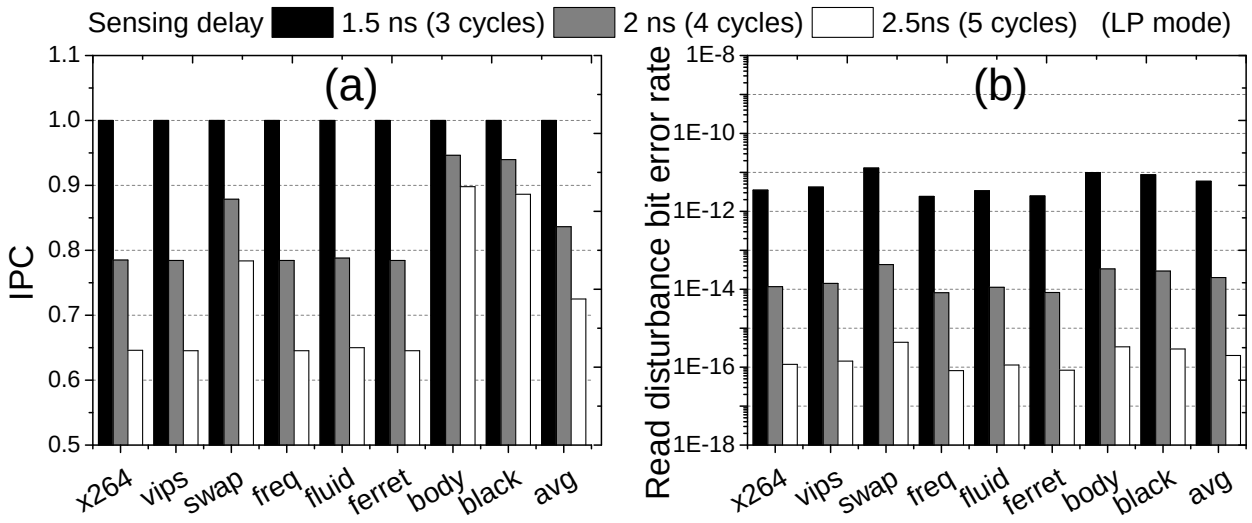


Figure 44: (a) IPC and (b) read error rate under different read latencies.

rewrite-induced degradation. In general, a larger shadow rewrite buffer is more effective to reduce the conflicts. When the entry number is above 4, the trend of IPC improvement becomes flat. The only exception is `x264`, which has much more intensive overall accesses as shown in Figure. 43(a). Further increasing the the buffer size can introduce the latency and energy consumption for buffer searching during read operations. After integrating the 4-entry shadow rewrite buffer, the average IPC degradation in HA mode compared to LP mode reduces to 7%.

Energy and EDP Figure. 47(a) compares the L2 cache overall energy consumptions under the different configurations and different operation modes, with the breakdowns of *shadow rewrite buffer search* (SD search), *rewrite* (Rw), *read* (Rd), *write* (Wr), and *leakage* (leak). In HA mode, the rewrite accounts for about 4% to 12% of overall energy, depending on applications. The leakage energy without the *shadow rewrite buffer* (SD) is much higher than that with the SD because of the prolonged execution time due to performance degradation. The SD increase the rewrite energy slightly and incurs a very small portion of SD searching overhead. Compared to Conv-STT-RAM, the FS-STT-RAM in HA mode with SD can

still save $\sim$ 12% total energy on average. The corresponding *energy-delay-product* (EDP) comparison is shown in Figure. 47(b). When the proposed architecture running in LP or HA mode, 34% or 16% of EDP improvement over Conv-STT-RAM can be achieved, respectively.

Reducing number of rewrite bits Even though only partial of the read bits, *i.e.*, data 1 bits, need to be rewritten after each read operation, the overall rewrite energy overhead is not negligible due to the high volumn of read instructions. The situation is especially severe for those read intensive workloads as we have shown in Figure. 47(a). More explicitly, Fig 48(a) shows the rewrite bits number is close to the total write bits number for some read intensive workloads, such as `x264`. Therefore, the rewrite overhead cannot be ignored.

We proposed the write bit inverting scheme to reduce the rewrite bit number. As shown in Figure. 48(b), a finer granularity can yield a lower rewrite bits number by incurring more bit overhead. When divide a cache block into 16 segments, the number of rewrite bits can reduce 38% compared to the design with only 1 segments. In other words, we can save $\sim$ 38% of rewrite energy with 16 extra indicating bits per cache block.

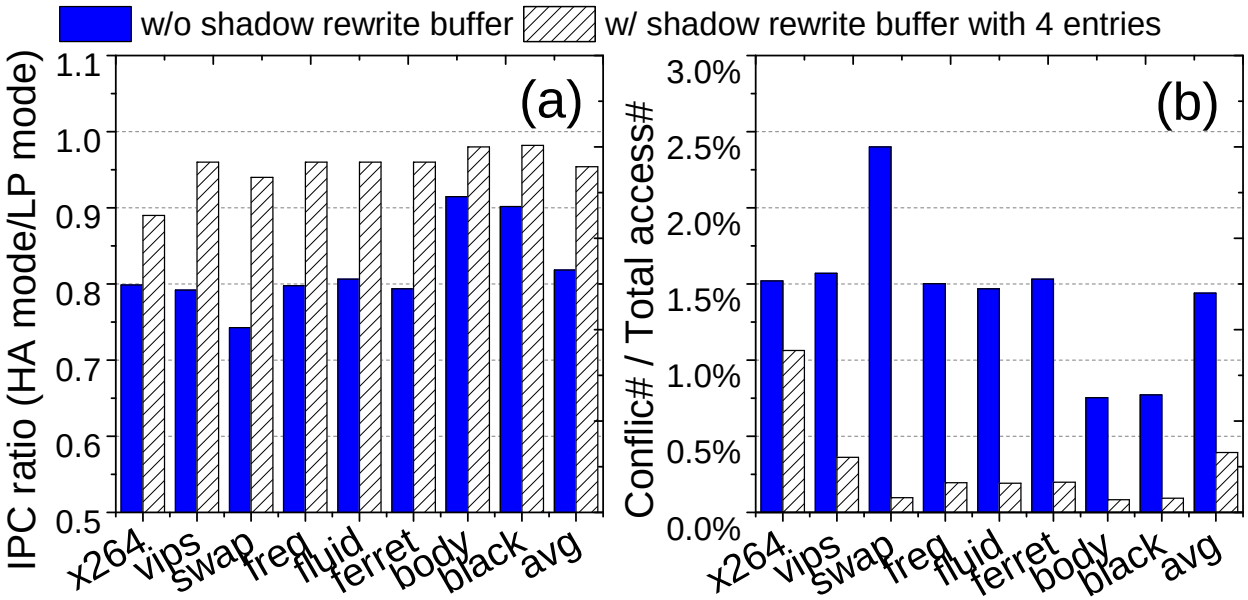


Figure 45: Comparison between LP and HA modes: (a) IPC, and (b) write and rewrite conflicts.

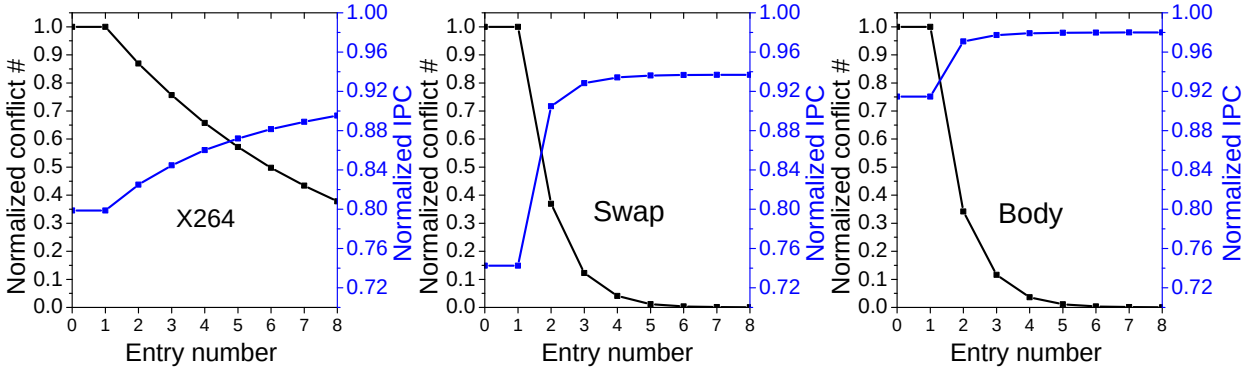


Figure 46: The effectiveness of shadow rewrite buffer at different entry numbers.

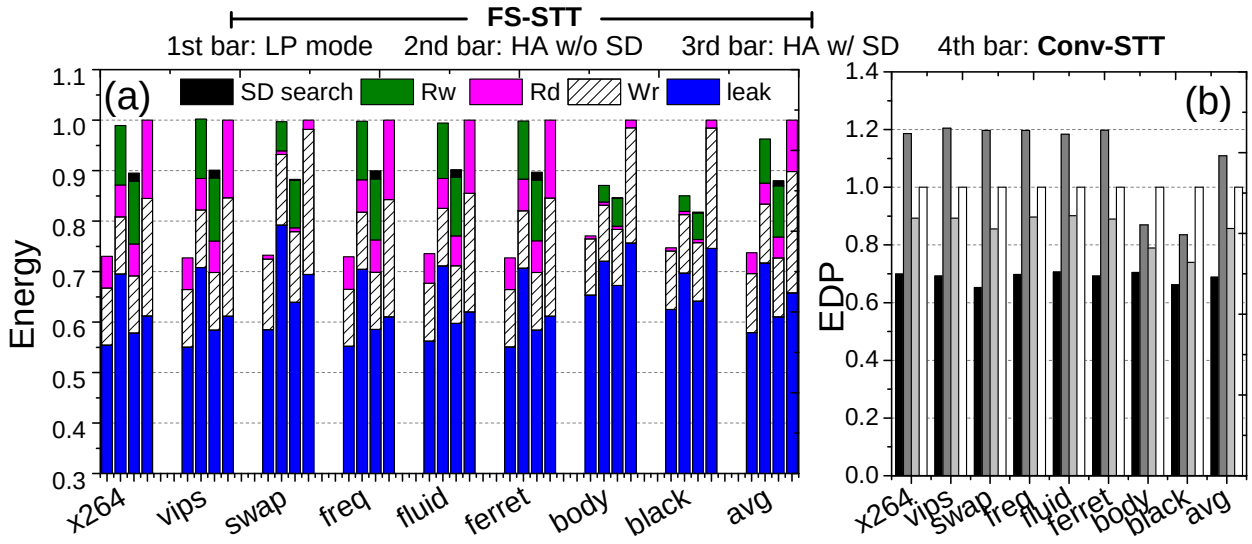


Figure 47: Energy and EDP comparison (normalized to Conv-STT-RAM).

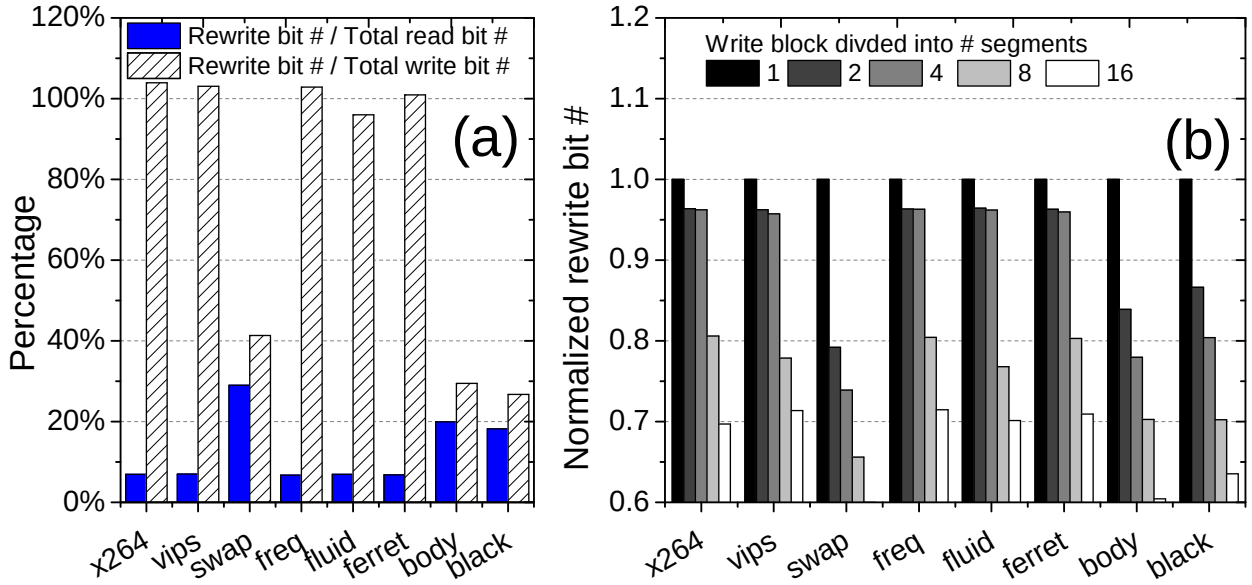


Figure 48: Rewrite bit number statistics and reduction.

5.4 SUMMARY OF CHAPTER

This chapter comprehensively study the write and read behavior of FS-STT-RAM at cross layers. Based on the device and circuit exploration, a memory architecture supporting both high accuracy and low power modes is proposed to cope with the read disturbance error for FS-STT-RAM. In the low power mode, data error rate is sacrificed to enable a high performance and low power operation, which on average achieves 6.7% of IPC and 34% of EDP improvements, respectively, over the Conv-STT-RAM. The FS-STT-RAM in the high accuracy mode uses a rewrite-after-read scheme to guarantee the data correctness. Combined with shadow rewrite buffer and write bit inverting scheme, the FS-STT-RAM still demonstrate 19% of EDP improvement over Conv-STT-RAM on average. Combined with shadow rewrite buffer, the FS-STT-RAM still demonstrate 16% EDP improvement over Conv-STT-RAM. The write bit invert scheme can further reduce rewrite energy by 38% with 16 bits/block overhead. With technology scaling, the process variation become more and more severe, we believe that our proposed schemes will become even more attractive because of its performance, low energy consumption, and simplicity.

6.0 RACETRACK MEMORY GENERAL EXPLORATION

In this chapter, we comprehensively consider design requirements across different abstraction layers. Based on the scaling trend of racetrack memory cell, a novel layout strategy is proposed to totally eliminates the area constraint of the access transistor size. A basic memory array structure is designed based on the layout approach, enabling both read and write operation at each access port. We comprehensively leverage various configurations of the racetrack memory architecture to find the most efficient one that providing highest floorplan area efficiency, lowest access delay and energy, general external interface, hardware support for further optimization, as well as simple and flexible physical-to-logic mapping. By considering the uneven distribution of data access pattern, an application-driven data management policy is designed. In the proposed policy, the access-intensive data blocks have more chance to be placed at the access port, further minimizing the racetrack shifting. We adapt racetrack memories with various geometrical dimensions to the proposed architecture to have an evaluation on the possible benefits and issues by doing scaling down.

6.1 REVISIT RACETRACK MEMORY CELL STRUCTURE

The racetrack memory comprises an array of magnetic stripes, namely, *racetracks* (RTs), arranged vertically [13] or horizontally on a silicon chip [18]. Figure 49 illustrates a horizontal RT structure that will be discussed in this work. It consists of many magnetic domains separated by ultra-narrow domain walls. Each domain has its own magnetization direction. Similar as STT-RAM, the binary values can be represented by the magnetization direction of every domain. And several domains share one access port for read and write operations. A select device together with an *magnetic tunneling junction* MTJ sensor are built at an access port. The motion of magnetic domain walls in a RT can be controlled by applying short current pulse I_{shfit} on the head or tail of the RT. To access a domain, we need a two-step operation: first *shift* it to an access port and then *read* or *write* the domain by applying an appropriate current (I_R or I_W).

Scalability of racetrack memory: Similar as STT-RAM, the device engineering of RT memory can be classified into two types – in-plane and perpendicular – according to the anisotropy direction of its magnetic layer. The *perpendicular magnetic anisotropy* (PMA) RT memory can provide a higher energy barrier even when the volume of domain cell is very small, enabling the continuous scalability for racetrack memory [89].

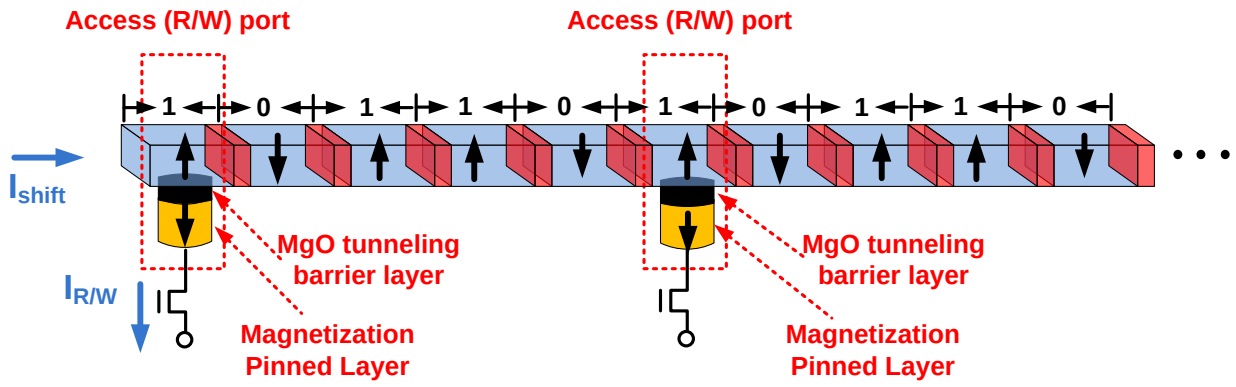


Figure 49: A racetrack in horizonal structure.

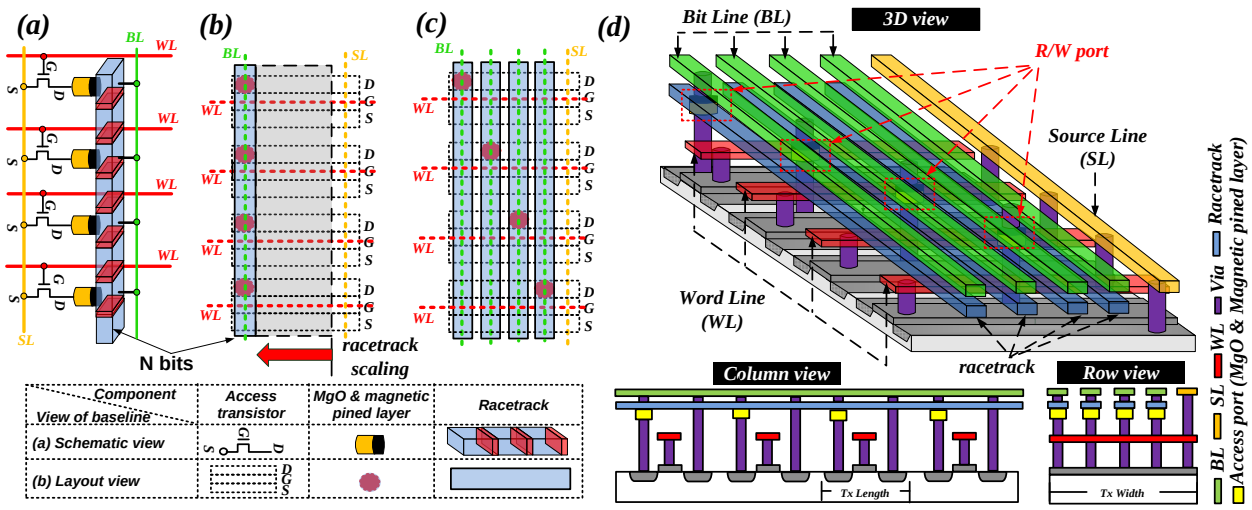


Figure 50: (a,b) Schematic and layout of the baseline RT design. (c,d) The proposed RT layout, 3D structure and cross-sections.

6.2 CROSS-LAYER DESIGN OPTIMIZATION

6.2.1 Cell and array designs

The accelerated storage density improvement introduced by the advanced RT memory enlarges the area gap between small memory element and relatively large NMOS select transistor. Figure 50(a) and (b) illustrate the schematic of a column of RT memory and the corresponding layout. The table in the figure summarizes the used components and symbols. The layout design shows that a RT (the blue strip) consumes only a small portion of the space above access transistors. The area highlighted in the gray shadow however is wasted.

A straightforward way to improve RT layout area efficiency is decreasing access transistor size. However, the driving current provided by small transistor might not be sufficient to switch magnetic domains. We can use these access points for only read operations (R-ports). One or a few W-ports associated with large access transistors are still necessary for a RT.

The fewer W-ports result in longer RT shifting in write operations and a larger overhead of domain cells. This is how macro cell was designed in *TapeCache* [40]. Without re-engineering the RT cell design, it is difficult to increase array area efficiency and read/write (R/W) accessibility at the same time. Here, we propose new RT cell and array designs to achieve better optimization.

Memory cell design: Figure 50(c) depicts the proposed layout of a column of RT memory. Multiple RTs are arranged side by side to cover the whole space above select transistors and their access points to the select transistors are placed in a diagonal manner. The 3D structure and cross sections in Figure 50(d) illustrate the placement and route of metal wires and RTs. In this design, the number of RTs per column is determined by the widths of RT and selection transistor. For example, Figure 50(c) assumes 4 RTs per columns, resulting in 4× memory density compared to the baseline layout. Note that the four transistors share one source-line (SL), but each of them is connected to only one RT and hence one bit-line (BL).

The proposed RT layout design can maximize the utilization of the space above CMOS layer. The size of the access transistor size, as far as it is large enough for write operations, is not the limiting design factor. In fact, the selection of access transistor size becomes more flexible and can be used to facilitate architecture optimization.

Memory array design: Figure 51 shows the circuit schematic of a basic RT memory array, which supports the following three basic operations:

Shift: Shifting a RT up (su) or down (sd) is realized by a bi-directional shifting current (I_{shfit}), which is controlled by signal 'su+', 'su−', 'sd+' and 'sd−'.

Write: The write current (I_W) in a write-'1' (or write-'0') operation is provided by enabling 'wr1+' and 'wr1−' (or 'wr0+' and 'wr0−').

Read: 'rd' and 'wr1−' are turned on so that a small read current I_R can be supplied to the target cell. The voltage generated on its BL will be delivered to a sense amplifier (not included in the figure) for data detection.

For array's perspective, extra magnetic domains, or *RT-overhead*, shall be added at both ends of a RT. The RT-overhead provides the extra space to store the bits shifted out of the original data portion during accesses.

Such memory cell and array designs significantly improve the area efficiency of RT memory, leading to a unprecedentedly high density. *The design is the first one enabling read and write operations at every access port without producing side effect on the area efficiency.*

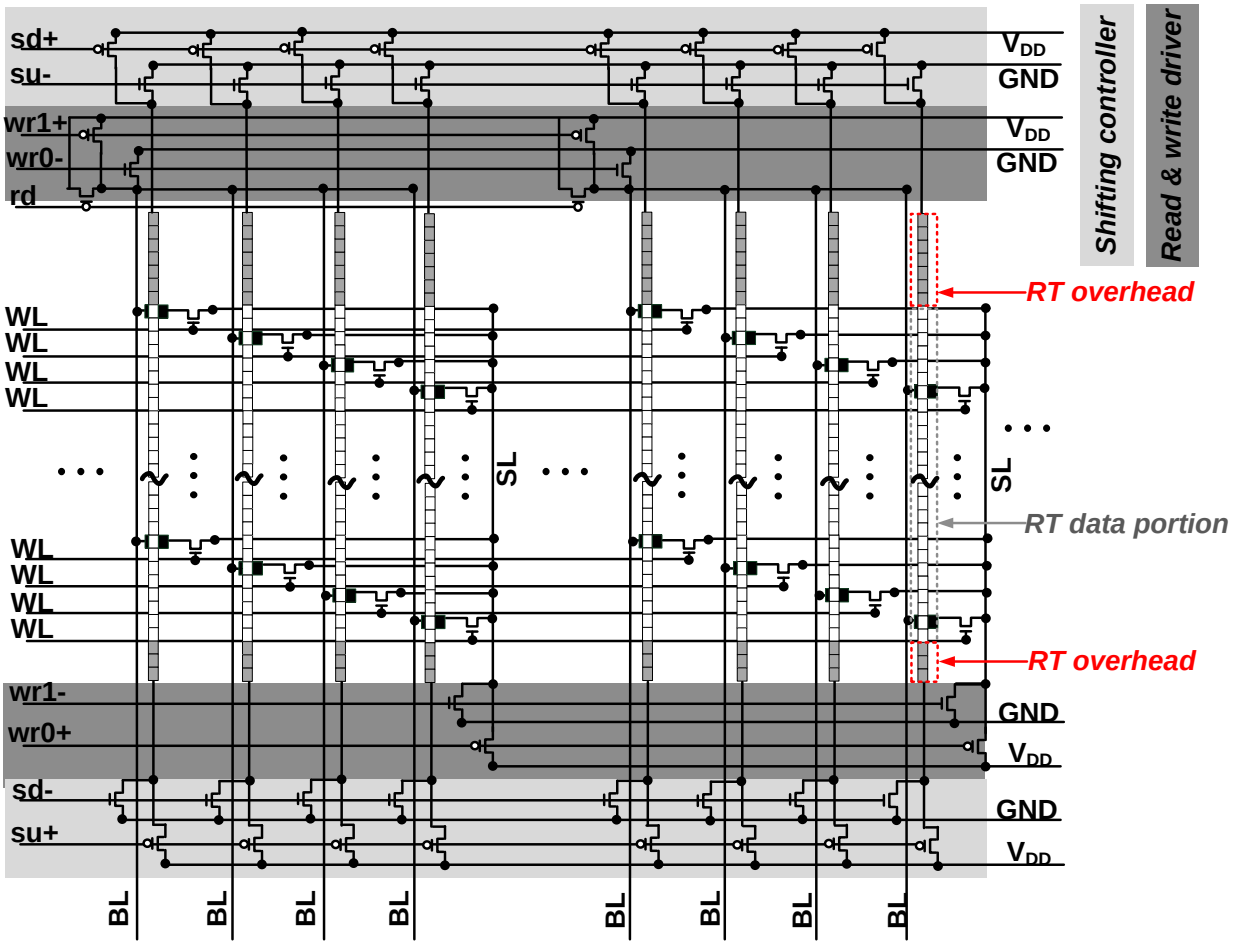


Figure 51: The circuit schematic of a RT memory array.

6.2.2 Architecture exploration

We explored the RT memory architecture based on the proposed cell and array design. The efficiency of a RT architecture is related to the basic array configuration, architectural organization, and physical-to-logical mapping. The complexity of hardware design shall also be included. In this section, we comprehensively investigate the impacts of these design considerations in the floorplan utilization, performance optimization, and energy consumption.

Figure 52(a) illustrates the proposed *hierarchical and dense architecture* for RT (**HDART**). The proposed HDART maintains the same I/O interface as current memory hierarchy to ease the technology adoption. Within the architecture, however, the bank organization could be flexible. For example, an entire cache architecture can be physically partitioned into N_B banks, each of which has its own I/O ports to support concurrent transactions.

Sub-array configuration: As the smallest component in architecture construction, a sub-array in Figure 52(b) can significantly affect on the overall performance of the entire architecture. The sub-array based on the memory array structure in Figure 51 shall be carefully configured according to design requirements. Three parameters are used when evaluating various sub-array configurations: (a) *the RT shifting energy*, (b) *the sub-array area efficiency* defined as the ratio of the data array area and the peripheral circuit area, and (c) *the RT overhead ratio* which is the ratio between the RT-overhead and the total length of RT.

RT length is an important design parameter related to RT shift energy. A longer RT produces higher runtime shifting energy, while a shorter RT degrades the sub-array area efficiency and has the higher RT-overhead ratio. We can also divide a long RT into several segments. Each segment needs its own shifting controller and the entire RT has one shared read/write driver. In general, more segments indicates lower shifting energy, but lower sub-array area efficiency and higher RT overhead. Figure 52(c) compares the different sub-array configurations in normalized scale.

Architecture exploration: Based on the basic array design, the memory architecture configuration, including banks, sub-banks, arrays etc., can be adjusted to satisfy the different design specifications including the criteria of performance, energy, and area constraint. We evaluated and compared the 4MB RT LLC designs based on different basic array configura-

tions. Here, three typical basic array configurations were selected, representing the designs with (1) more segments in one RT, (2) one medium-length RT (e.g., 64 bits long), and (3) one long RT, respectively.

The detail comparison in terms of area efficiency, performance, and energy consumption at the LLC architectural level were conducted from six design matrices and shown by the hexagon graphs in Figure 52(d). The dotted border of a hexagon map indicates the optimal expectation. All these design matrices are relevant with each other and determined by the both of the architecture and sub-array configuration. In the work, we are primarily interested in a RT-based cache design with high performance and low power consumption. We selected configuration (2) in Figure 52(d) for following evaluations, due to its lower access delay and energy.

Physical-to-logic mapping: The physical-to-logic mapping is orthogonal to the memory architecture design but shall be optimized based on the specific design requirement. Figure 53 gives an example of mapping on the top of the proposed HDART architecture. In the example, a sub-array contains 64 groups of RTs, each of which has 4 RTs. And each RT has 8 access points. The magnetic domains connected to a single R/W port in the physical design corresponds to the same bit number of cache blocks within the same set from different ways. For instance, as illustrated in the figure, Bit 0 (b0) of 32 cache blocks belonging to all the 32 ways (w) within Set 0 (s0) are all mapped to the first group of RTs in sub-array 0 of array 0. The RT shifting during an access can controlled by a physical-login mapping unit, e.g., LUT in Figure 53. The shifting distance shall be determined by the block's way number and the current track position stored in a track status register T_{reg}.

Because all the same bits from the different ways are within the same array and controlled by a single access port, such a design is in favor of data block reordering among different ways, the data management method that shall be discussed in Section 6.2.3. Nevertheless, unlike way reordering that can be easily implemented with tag mechanism, set reordering requires set re-mapping table which results in extra area, delay and energy overhead. Thus, we select way reordering for the data management method.

Hardware design complexity: Besides the RT memory itself, the design complexity of other related hardware need to be considered. For example, though tag array contributes

only 5% of total area in SRAM LLC design, it could become the major bottle in RT memory resulting from the unbalanced scaling trends of data storage and tag array. Here, utilizing STT-RAM for tag design can alleviate the impact, which is adopted in our design. Several components are introduced in the proposed RT memory, including a track status register (T_{reg}) to store RT position, a look-up-table (LUT) to assist physical-to-logic mapping, and a block counter (BCT) indicating the data access intensity for data management in Section 6.2.3.

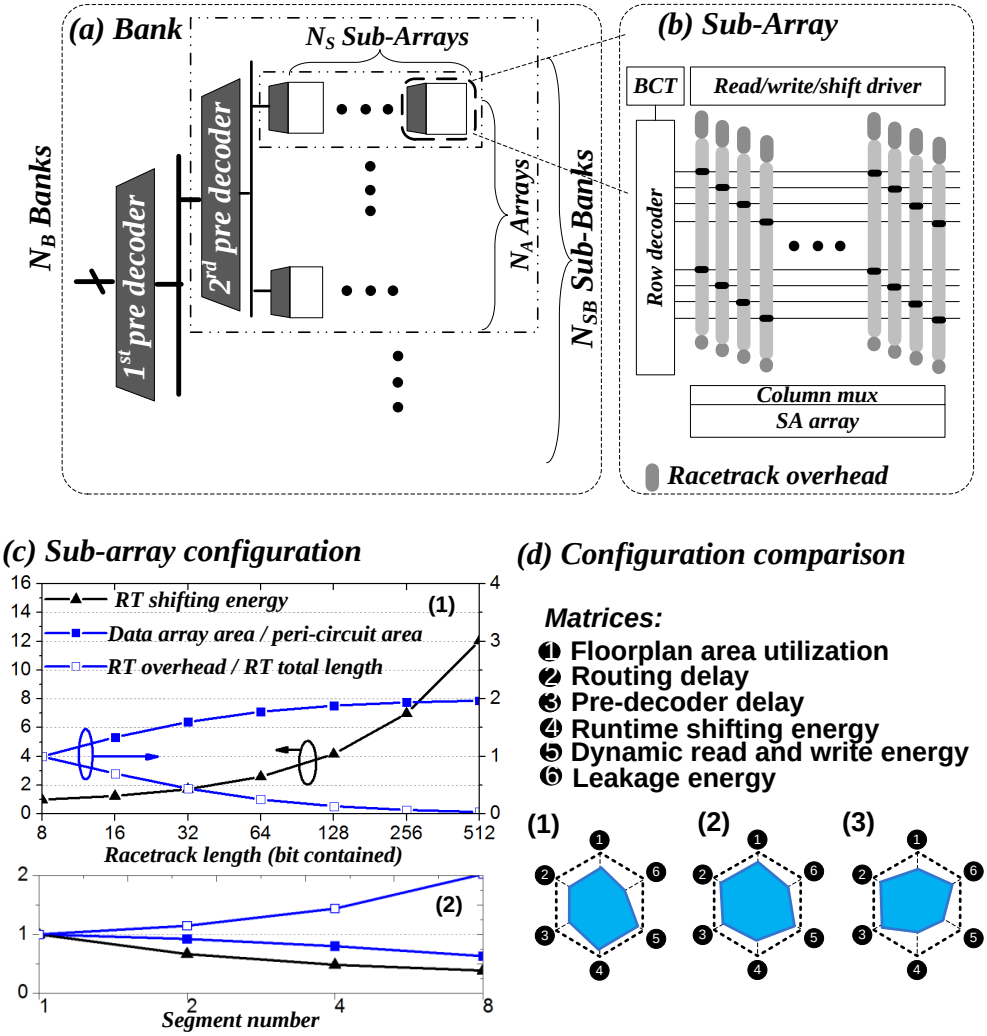


Figure 52: The RT memory architecture design exploration.

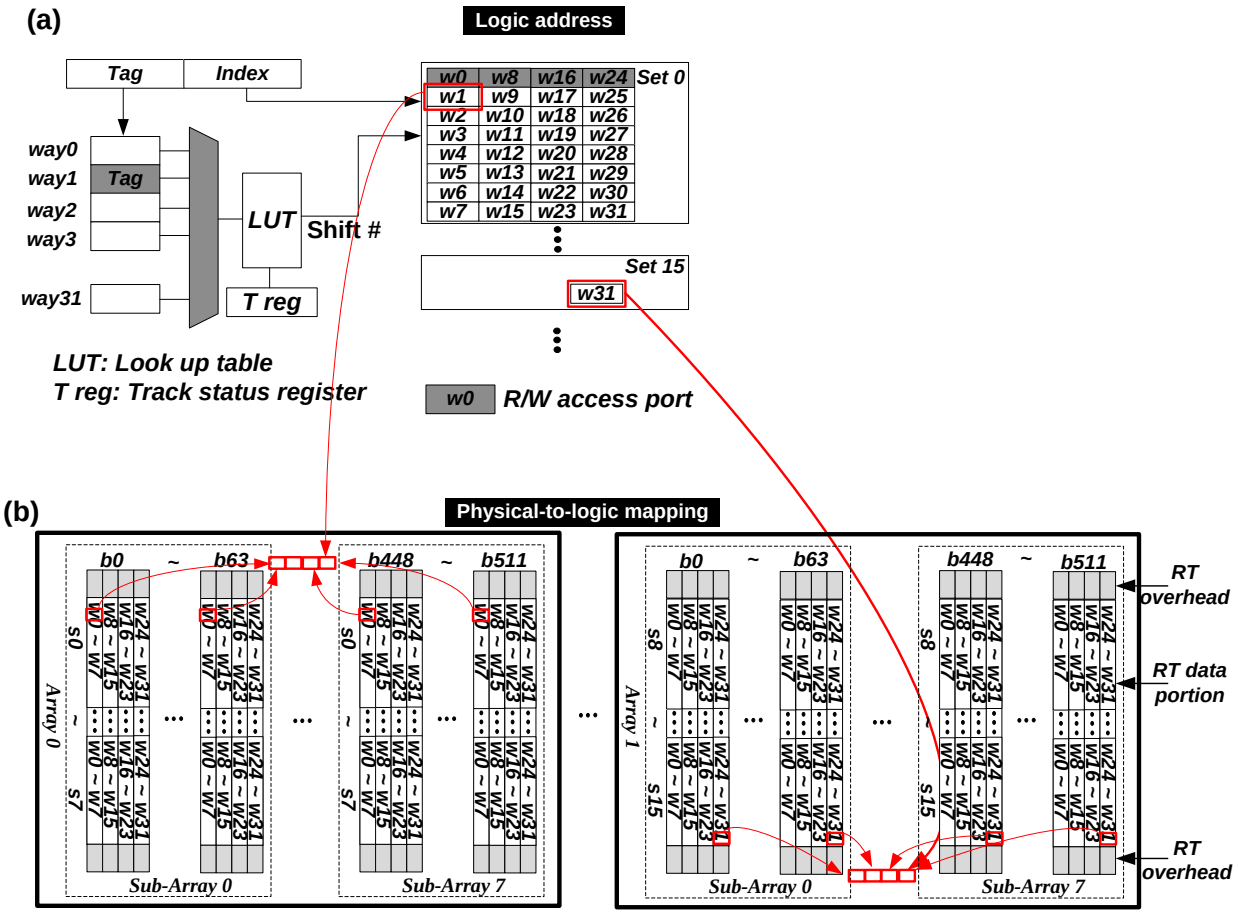


Figure 53: A physical to logic address mapping scheme for RT memory.

6.2.3 Data management policy

In a traditional random access memory, every storage element has its own access path. In contrast, many magnetic domains in a RT memory share one R/W port. During an access, a domain need to be shifted to R/W port, inducing extra overhead in access latency and energy consumption. We propose two *track shifting policies*:

- *TS1:* After an access is completed, a RT stays where it is.
- *TS2:* A RT returns to its original position after each access.

TS1 benefits when cache accesses show strong spacial locality, but generates frequent RT shifting when randomly distributed cache accesses dominates. Resetting RT in *TS2* potentially increases the frequency of RT shifting. But the data management in *TS2* is easier due to the fixed relation of the memory cells to their R/W ports.

The cache accesses in many applications are unevenly distributed and only a small portion of cache blocks are frequently accessed [85]. Thus, we propose a RT data management policy, named as *hardware based way block reorder* (HBWBR), to alleviate the shifting overhead. By tracing the data access pattern, HBWBR can identify the cache blocks with intensive accesses and then place/swap them to the physical locations onto the R/W ports.

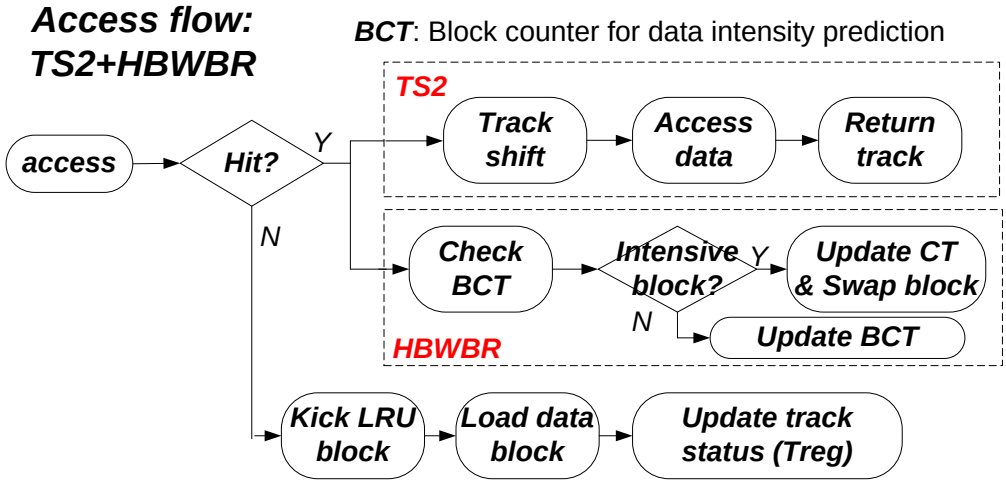


Figure 54: The RT memory access flow with TS2 and HBWBR.

The access flow and timing: Figure 54 depicts the HDART access flow when applying TS2 and HBWBR together. The corresponding access timing diagram is shown in Figure 55.

Note that track shifting (TS2) and data management (HBWBR) are executed simultaneously and are independent to each other. Thus, the access to block counter for data intensity prediction (BCT) does not introduce any extra latency overhead. An access hit in HDART triggers the examination of BCT. If the access block is predicted to be access intensive, we swap it with the one at R/W port. Otherwise, the BCT is updated by increasing 1. During a cache miss, the least-recently used (LRU) policy is adopted.

The timing flow and the critical path in HDART accesses is similar to STT-RAM cache access, except extra RT shifting delay shall be included in every data access. The track reset delay can be hidden with the routing to output (RTO) delay during read. A data swap induces a relative big delay overhead, but it occurs much less frequently with regard to total access number. The detail explanation of timing components are listed at the bottom of Figure 55 for reference. And the timing component parameters can be found in Table 10. The effectiveness of HBWBR relies on the efficiency of both the intensive block prediction and the data swap.

Intensive block prediction: For design simplicity, a counter-based scheme is used for cache access intensity prediction. In the design, each data block is associated with a *block counter* (BCT). When a cache hit occurs, the corresponding counter of the data block increments by one. All the counters are self-decremented periodically till it becomes to '0'. A data block is considered as an access-intensive block once its counter exceeds the predefined threshold.

Block swap: In Section 6.2.2, we propose to map the same bits from all the ways within one set into one array. Therefore, the data on some ways sitting right on the R/W ports can be accessed without any track shifting. We name these ways as '**fast way**'. Moreover, since data exchanges occurs within the same array in such a design, the way-based block swapping is convenient and energy efficient. Therefore, we propose to swap an access-intensive block with those on fast ways. When a block outside fast way is regarded as access-intensive block, it swaps with the data block in one of the four fast ways with the smallest access number.

As shown in Figure 53, each set consists of four fast ways on four RTs. Accordingly, the data swap could occurs within the same track or between two different tracks. The different latency and energy overhead caused by these two types of data swap operations will be included in system evaluation in Section 6.3.

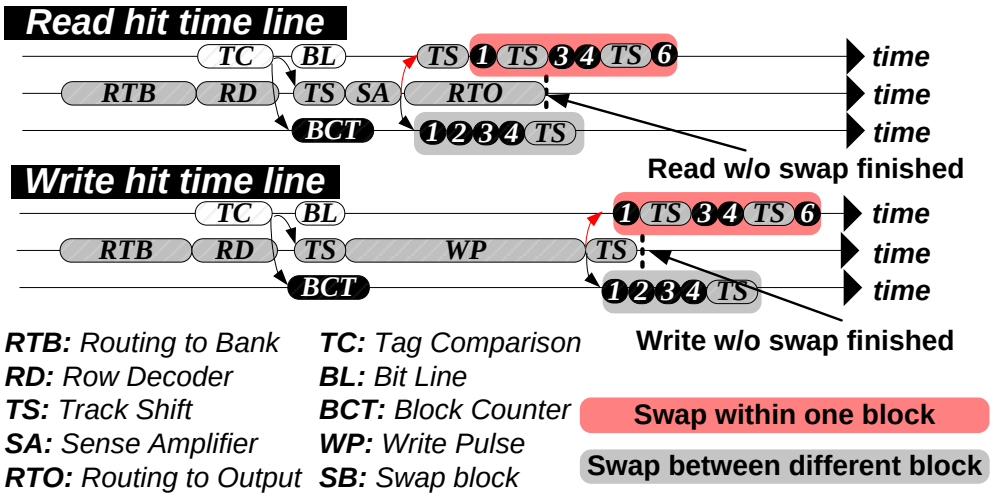


Figure 55: Cache access timing.

Easy integration with HDART: *HBWBR* is an efficient data management scheme for RT memory that has variable access latences. In the previous RT memory design, a RT macro cell has only one R/W port but multiple R-ports [40]. The data management in such a design requires to distinguish the read- and write-intensive data blocks and allocate them to different ports. Especially, the limited W-port number constrains the optimization space. So it is less adaptable for efficient data management. In contrast, our proposed HDART supports both read and write operations at every access ports, easing the design complexity and enhancing the efficiency of data management. Therefore, the data management can be naturally integrated on the proposed HDART.

6.3 SIMULATION

6.3.1 Simulation setup

Table 10: Design parameters for different cache types

Cache parameters	Area				
	Cell size	Total area	Data	Tag	Periph
SRAM	125 F^2	9.09 mm^2	93.43%	4.67%	1.90%
STT	32 F^2	2.51 mm^2	86.28%	11.44%	2.28%
OP-STT	28 F^2	2.24 mm^2	84.55%	12.80%	2.65%
Baseline RT	16 F^2	1.41 mm^2	76.91%	20.37%	2.72%
HDART (4 F^2)	4 F^2	0.599 mm^2	45.37%	48.08%	6.55%
HDART (1 F^2)	1 F^2	0.390 mm^2	17.42%	73.84%	8.74%

Cache parameters	Timing (Timing components are illustrated in Figure55(a)							
	RTB	*RD*	*BL*	*SA*	*TC*	*RTO*	*WP*	*TS*
SRAM	**2.03**ns	0.19ns	0.12ns	0.2ns	0.5ns	**2.10**ns	–	–
STT	0.56ns	0.16ns	0.10ns	0.2ns	0.3ns	0.58ns	10ns	–
OP-STT	0.52ns	0.15ns	0.10ns	0.2ns	0.3ns	0.53ns	5ns	–
Baseline RT	0.31ns	0.14ns	0.07ns	0.2ns	0.3ns	0.32ns	5ns	0.5ns/shift
HDART (4 F^2)	**0.13**ns	0.13ns	0.03ns	0.2ns	0.3ns	**0.13**ns	10ns	0.5ns/shift
HDART (1 F^2)	**0.08**ns	0.1ns	0.01ns	0.2ns	0.3ns	**0.08**ns	5ns	0.5ns/shift

Note: One shift operation means shift the whole racetrack memory cell up or down with the unit distance of one domain cell bit. The shifting current can be tuned by sizing the shifting controller transistor. The racetrack shifting (domain wall motion) velocity is determined by the shifting current density. With carefully tuned current, it takes one cycle (0.5ns) to shift the racetrack cell for one unit of distance.

Cache design parameters: We evaluated and compare 4MB LLC design by using different memory technologies, including SRAM, STT-RAM, and RT memory. The cache configuration is set as $N_B = 4$, $N_{SB} = 8$ and $N_S = 8$ (refer Figure 3(a)). The cache latency and energy parameters were obtained based on SPICE simulation and the modified NVsim [79]. The domain wall shifting energy was calculated from micro-magnetic simulations. These parameters are summarized in Table 10 and Table 14.

Evaluation platform: We performed the evaluations on an 8-core UltraSPARC T1 processor by adopting various memory technologies as 4MB LLC. Table 12 summarizes the process configurations. The cache model of Simics toolset [81] was modified according to the different memory requirements. The multi-threaded benchmarks from Parsec Benchmark Suite [82] were adopted in simulations. For each benchmark, we fast-forward to region of interest, warm up the cache for 200 million instructions, and then execute 500 million instructions.

The following baseline memory technologies were selected for comprehensive comparison. They are: **SRAM**, **STT-RAM**, **OP-STT** [20], and **Baseline RT** (layout in Figure 61(b)).

Table 11: Energy components of diff. memory technologies.

	Write eng	Read eng	Leakage pwr	Shift Eng
SRAM	$0.35nJ$	$0.42nJ$	$4100mW$	–
STT	$1.92nJ$	$0.36nJ$	$130mW$	–
OP-STT	$1.52nJ$	$0.34nJ$	$120mW$	–
Baseline RT	$1.07nJ$	$0.22nJ$	$83mW$	–
HDART($4F^2$)	$0.57nJ$	$0.074nJ$	$46mW$	$0.62nJ$/shift
HDART($1F^2$)	$0.46nJ$	$0.037nJ$	$33mW$	$0.31nJ$/shift

Note: The write, read and shift energy is for one cache block.

Table 12: Processor configuration

Processors	8 cores, 2GHz, 4 threads/CPU core, 1-way issue
SRAM L1 Cache	Local, 16KB I/D, 2-way, 64B line, 2-cycle, write-back
LLC Cache	Shared, 4MB, 4 banks, 32-way, 64B line, write-back, 1 read/write port, 4 write buffers.
Main Memory	4GB, 400-cycle latency.

We evaluated the impact of the RT technologies under the proposed HDART by comparing two RT geometrical dimensions: a moderate domain size of $4F^2$ to reflect current device engineering and an aggressive racetrack design with domain size of $1F^2$, corresponding to the racetrack with a width of $1F$. After applying different tracking shifting and data management policies proposed in the work, totally six RT memory configurations were examined. They are: ***$4F^2$+TS1***, ***$4F^2$+TS2***, ***$4F^2$+TS2+HBWBR***, ***$1F^2$+TS1***, ***$1F^2$+TS2***, and ***$1F^2$+TS2+HBWBR***.

6.3.2 Simulation results

Comparison to baseline memories: To demonstrate the potential of RT memory, we first compare the HDART with baseline memory technologies. Figure 56(a) shows the performance results represented by the normalized *instruction per cycle* (IPC). HDART the simple track shifting policy *TS1* achieves 10% $\sim$ 15% IPC enhancement over SRAM and OP-STT. The shorter routing latency for both read and write operations in HDART dominates the IPC performance improvement, though the extra delay caused by track shifting slightly offsets the benefit. Compared the most advaned OP-STT, HDART+*TS1* can achieve an average 19% energy saving. The detail energy breakdowns of three most energy efficient memory technologies are shown in Figure 56(b). *Baseline RT* has more R/W ports (due to much more access transistors) and hence consumes $\sim 2\times$ less track shifting energy than

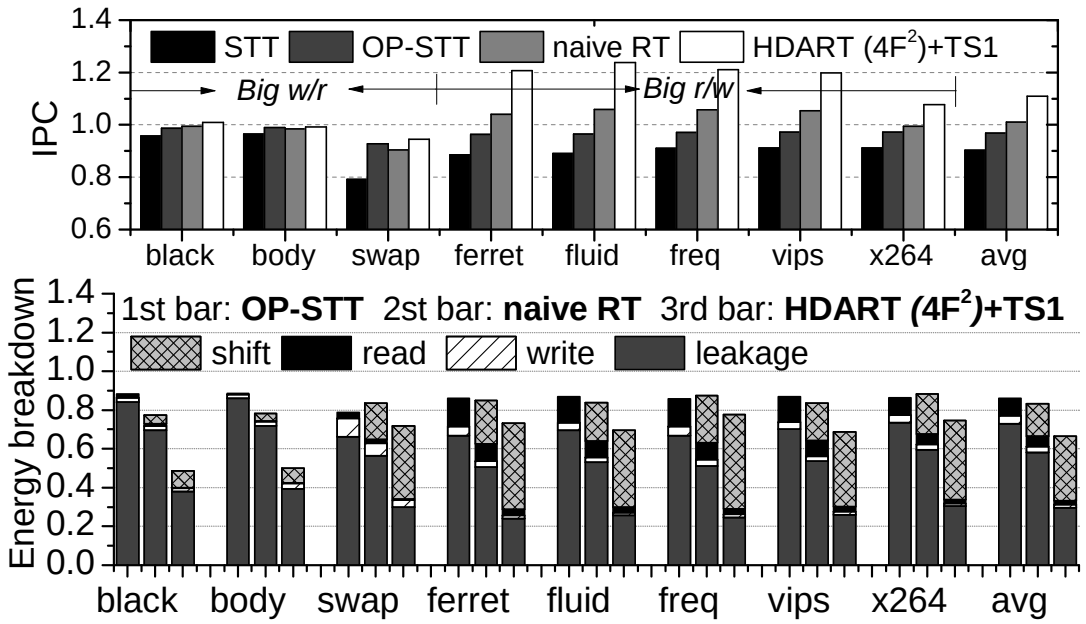


Figure 56: Comparison with baseline memory technologies (a) IPC performance (b)Energy breakdown.

HDART. However, HDART's overall energy consumption is 18% less than Baseline RT. The saving comes from less leakage and dynamic energies due to its higher density (refer Table 1).

The swap threshold selection: directly determines the effectiveness of data intensity prediction and the frequency of runtime data block swaps. Figure 57(a) summarizes the average statistical data of all benchmarks including the hit number on fast ways, the shift number, and the swap number when changing the swap threshold from 3 to 32. It shows that when the threshold exceeds 11, the hit number in fast way decreases dramatically, resulting in a significant increasing of racetrack shift number. Moreover, the swap number reduces fast before the threshold approaches 11 and becomes flat when the threshold is big. According to the results in Figure 57(b), setting the swap threshold as 10 can achieve the best energy-delay product.

HBWBR effectiveness: The above comparison to baseline memories, HDART is only equipped with *TS1*. We evaluate the effectiveness of the proposed *HBWBR* in this section. The results in Figure 56(b) show that a big portion of HDART energy comes from track

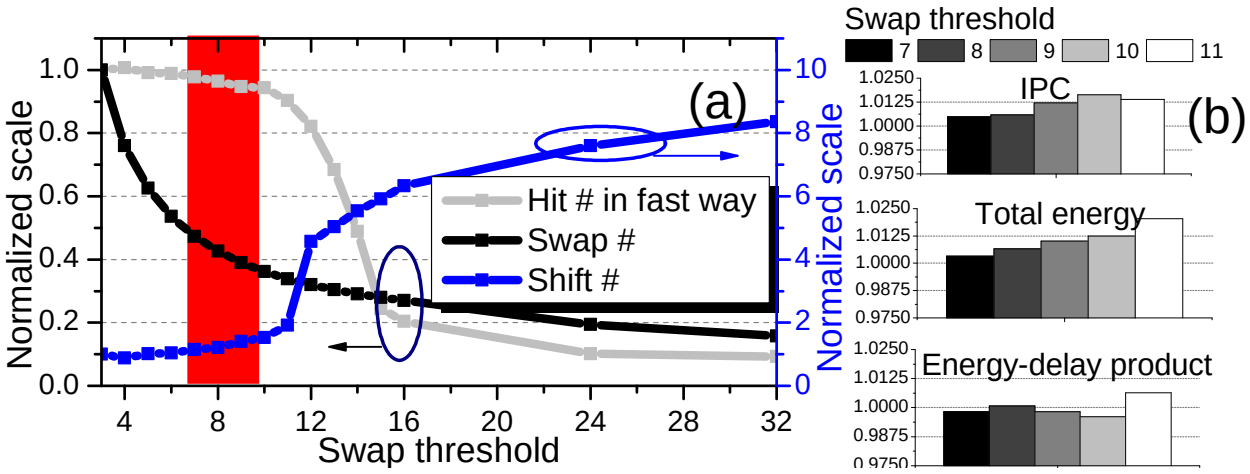


Figure 57: Swap threshold selection.

shifting. Therefore, reducing the track shifting is necessary to improve energy consumption. Figure 58 shows the trend of shift numbers at the beginning of simulations for different benchmarks. The shift number decreases with time with the assistant of *HBWBR*. Note that the different benchmarks apply the different observation windows in the figure for their different reduction rate of shift number. On overage, *HBWBR* helps remove 60% of track shift, indicating 60% shifting energy reduction.

Figure 59 shows the IPC performance comparison of HDART under different policies. And the detail energy comparison can be found in Figure 60. *TS2* alone suffers from more track shift operations, resulting in 2.5% performance degradation than *TS1*. *HBWBR* effectively reduces shifting overhead and improves IPC 7.5% on average. In summary, the HDART design $4F^2$+*TS2*+*HBWBR* achieves achieve 4.2× area reduction, 20% performance enhancement, and 49% energy saving, compared to STT-RAM cache design. Compared to SRAM, the performance is improved by 13% and the energy consumption is reduced by 40×.

Impact of RT memory cell sizes: We evaluate the effect of the RT memory cell size under the same technology node (*e.g.* 45*nm*). Smaller racetrack domain cell leads to an even more compact LLC, leading to faster access and less energy (including both dynamic and leakage) consumption. However, the possible side effect could be more shifting numbers, because one R/W port will be shared by more magnetic domains. But, smaller domain

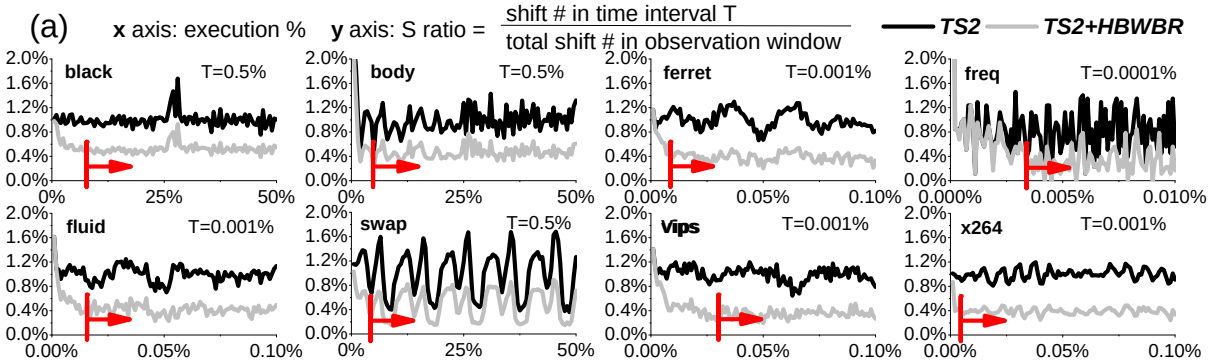


Figure 58: Shifting reduction by *HBWBR*.

dimension could lower per shift energy so that the side effect can be partially compensated. We compared the HDART designs built with a moderate domain size of $4F^2$ and an aggressive racetrack design with domain size of $1F^2$. The performance and energy comparison can be found in Figure 59 and Figure 60, respectively. The results show that designs of $1F^2$ can achieve 3% and 11% performance and energy saving compared to designs of $4F^2$.

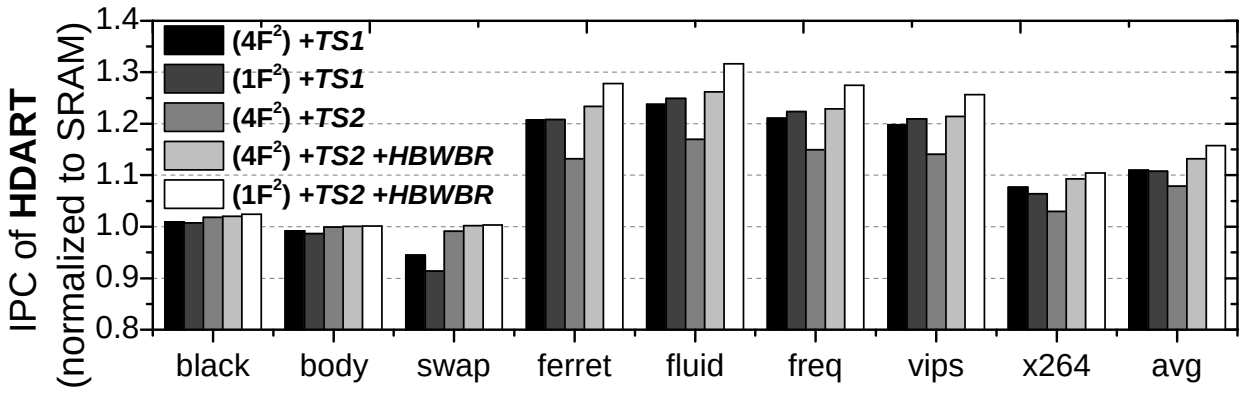


Figure 59: Performance enhancement by different policy for **HDART** (Normalized to ($4F^2$+*TS1*)).

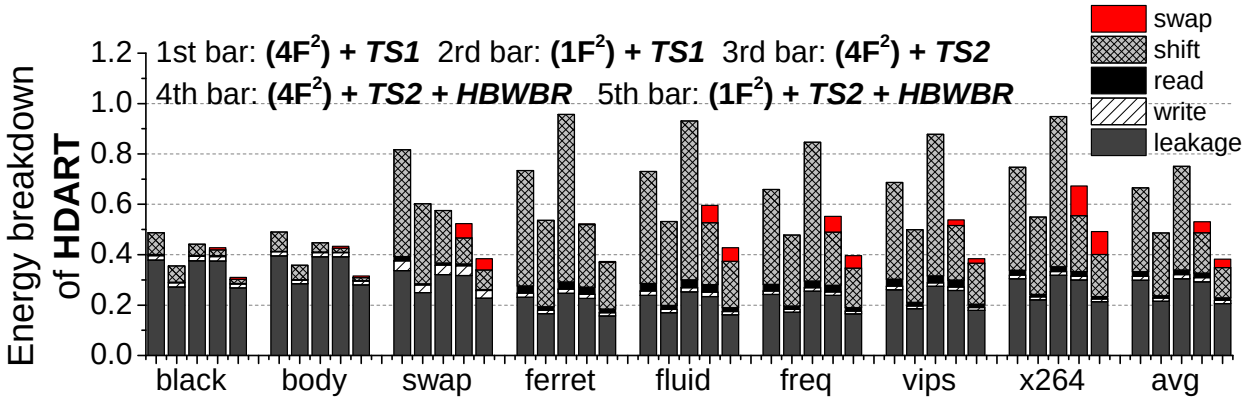


Figure 60: Energy breakdown by different policy for **HDART** (Normalized to STT).

6.4 SUMMARY OF CHAPTER

In this chapter, we performed a comprehensive exploration and design enhancement for *Racetrack* memory across multiple layers. We initialize the design exploration with a novel layout approach which enables an all R/W ports memory array structure. A flexible hardware architecture (HDART) is also proposed based on the memory cell and array design. We proposed a data management scheme that can be naturally integrated onto the proposed HDART, further improving the efficiency for the RT based LLC. We compared RT based HDART with state of art memory technologies. The RT based HDART with data management can achieve 6.4× area reduction, 25% performance enhancement, and 62% energy saving, compared to STT-RAM cache design. The improvement obtained from the proposed HDART is much higher than *TapeCache*.

7.0 INSIGHT OF RACETRACK MEMORY

In this chapter, we thoroughly exploited and evaluated the different physical layout strategies and array organizations of racetrack-based LLCs by comparing with the conventional SRAM and the latest magnetic STT-RAM technologies. We proposed a mixed array organization composing of hybrid-port and uniform-port arrays, optimized for serial regular cache accesses (i.e., instruction requests) and random data accesses, respectively. Depending on applications' access patterns, we propose to dynamically change the utilization of racetracks to realize resizable-set or resizable-way cache design to improve performance and reduce energy consumption.

7.1 RACETRACK VS. OTHER MEMORY TECHNOLOGIES

7.1.1 Racetrack memory array crafting

Memory cell structure and array organization can dramatically affect the area, performance, and energy consumption of cache designs. The situation in racetrack memory could be even more severe. Access ports can be designed to conduct only read accesses (*R-port*) or to support both read and write operations (*R/W-port*) [40]. From circuit design perspective, the fundamental difference between R-port and R/W-port is the size of access transistors. An read access requires only a small current so that a minimum-size transistor can be used in R-port design. The access transistor in a R/W-port, however, shall be much bigger to provide sufficient current (and hence enough torques) to switch the magnetization direction of magnetic domains in write operations. Moreover, the access transistor shape and the connection to racetracks are also related to the memory cell structure, leading to various array organization.

Figure 61 summarizes four typical racetrack array organizations. For each design, a piece of layout and its corresponding array are illustrated. Based on the combinations of access ports, we classified these designs into *hybrid-port array* and *uniform-port array*.

Hybrid-port array contains both R-ports and R/W-ports on each racetrack nanowire, as shown in Figure 61(a). To align with the layout of small R-port transistors, a wide R/W-port transistor has to be broken into two or more parallel-connected segments (*fingers*). In the array view, we use dark squares to represent the cells that can be read and written without extra shifts. The squares with cross correspond to the data bits that can be read out right away. Accessing a magnetic domain in blank squares, however, need first move it to a R-port or a R/W-port. Apparently, the shift cost of a data bit is determined by its physical location as well as the type of access. Writes executed on only R/W-ports incur more shift overheads than reads that can be conducted on any access ports. Increasing the number of R/W-port, however, results in less port numbers and potentially degrades read access latency.

Uniform-port array containing only R/W-ports can be realized with different R/W-port shapes and racetrack connections. Figure 61(b) $\sim$ (d) demonstrate three typical array

organizations. Figure 61(b) is the first-generation racetrack array design [39]. Due to the mismatch between the large access transistors of R/W-ports and narrow racetrack nanowires, a large portion of space on access transistors is wasted. Figure 61(c) utilizes the R/W-port design in (a), which break a wide access transistor into a few fingers. So the R/W-port is shrunk alone the row direction but extended in the column direction. It avoids the space wasting issue but the distance between adjacent access ports prolongs. Figure 61(d) adopts wide 1-finger layout for access transistors. A few racetracks are paved above a column of transistors and share one source line. The design fully utilizes the blank space and achieves super high density. However, the distance between adjacent access ports further increases.

7.1.2 Comparison of LLCs in Different Technologies

We utilize the racetrack array organizations RT1 $\sim$ RT4 (Figure 61) in 4MB LLC designs and compare them with those in the conventional SRAM and the latest STT-RAM technologies. Table 13 and Table 14 summarize the related cache latency and energy parameters obtained from the modified NVsim [79]. The domain wall shifting energy is calculated based on micro-magnetic simulations. More details in simulation setup and CPU configurations are provided in Section 7.3.1.

Figure 62 shows *instruction per cycle* (IPC) performance comparisons of different LLCs. The SRAM based cache is selected as the baseline. Without any optimization, simple iso-capacity replacement of SRAM or STT-RAM based LLC by racetrack memory results in fast system performance, mainly benefiting from its smaller array size and shorter interconnects. The average IPC gains of RT1 and RT4 designs are more than 3.7% over the SRAM based LLC.

Figure 63 shows the energy breakdowns of racetrack cache designs by using different arrays. Here, the baseline is the optimal STT-RAM based cache [**?**] which can reduce 70$\sim$90% of energy consumption compared to SRAM based LLC, which is omitted in the figure for better illustration. For all the four racetrack LLCs, we observe on average 25$\sim$30% of energy saving over the STT-RAM cache. Note that the shift energy is proportional to the runtime shift numbers. The corresponding statistics normalized to the shift number of RT1 design are shown in Figure 64. Without further optimization, the racetrack cache demonstrates a strong

correlation between the runtime shift number and the access ports distance. The designs in RT1 and RT2 have more R-ports or R/W-ports on racetracks and hence consume less shift energy. In contrast, the design in RT4 occurs much more runtime racetrack shifts because of the sparse access port distribution. However, for its highest density, RT4 array pays less energy in read, write, and leakage, and hence, obtains the lowest energy consumption in most of benchmarks. In summary, compared with the conventional SRAM and the latest STT-RAM technologies, the racetrack memory advances in terms of *higher density*, *less access transistors*, and *simpler peripheral circuitry*. Although it occurs the similar and even higher latency/energy on a single cell operation itself (read, write, and shift), the costs on *interconnect latency* and *leakage power* are dramatically reduced. Overall, racetrack memory wins in all the design matrices in terms of area, performance, and energy consumption.

Meanwhile, we note that the runtime racetrack shifts and the associated latency and energy overhead become a new design issue. The results in Figure 62 and Figure 64 show a large amount of racetrack shifts, resulting a significant portion of dynamic energy consumption. Especially, the LLC by using RT4 arrays obtains the highest density and the best energy-delay product among all the flavors, but pays the highest cost in shift operations due to the long distance of access ports. From the other side, RT4 array could have the greatest potential in architecture optimization.

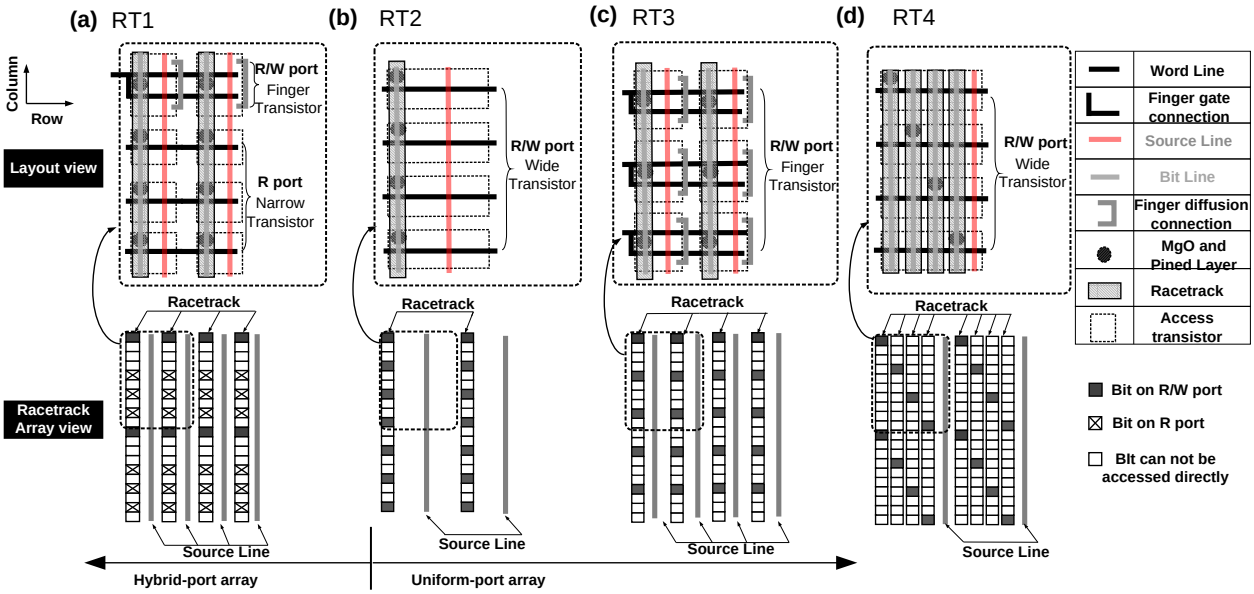


Figure 61: Various racetrack array organizations: (a) hybrid-port array; (b $\sim$ d) uniform-port arrays.

Table 13: Access Latencies of Diff. Memory Technologies

	SRAM	STT-RAM	RT1	RT2	RT3	RT4
Rd Lat.[1]	9+1	7+1	4+1	5+1	4+1	2+1
Wr Lat.[1]	9+1	7+10	4+10	5+10	4+10	2+10
Port Dist.[2]	–	–	Rd: 3; Wr: 8	3	4	8

[1] *Latency in clock cycles = peripheral latency + cell latency*
[2] *The port distance is represented by the number of magnetic domains.*
[3] *One shift means a racetrack moves a unit distance of one magnetic domain. We assume each shift takes one clock cycle* (0.5 ns) [39].

Table 14: Energy Consumptions of Diff. Memory Technologies

	SRAM	STT-RAM	RT1	RT2	RT3	RT4
Read Energy (nJ)	0.42	0.34	0.16	0.22	0.16	0.074
Write Energy (nJ)	0.35	1.52	0.97	1.07	0.97	0.57
Shift Energy (nJ/shift)	–	–	0.62	0.62	0.62	0.62
Leakage Power (mW)	4100	120	65	83	70	46

Energy per cache block includes consumptions on peripheral circuit & cell operations.

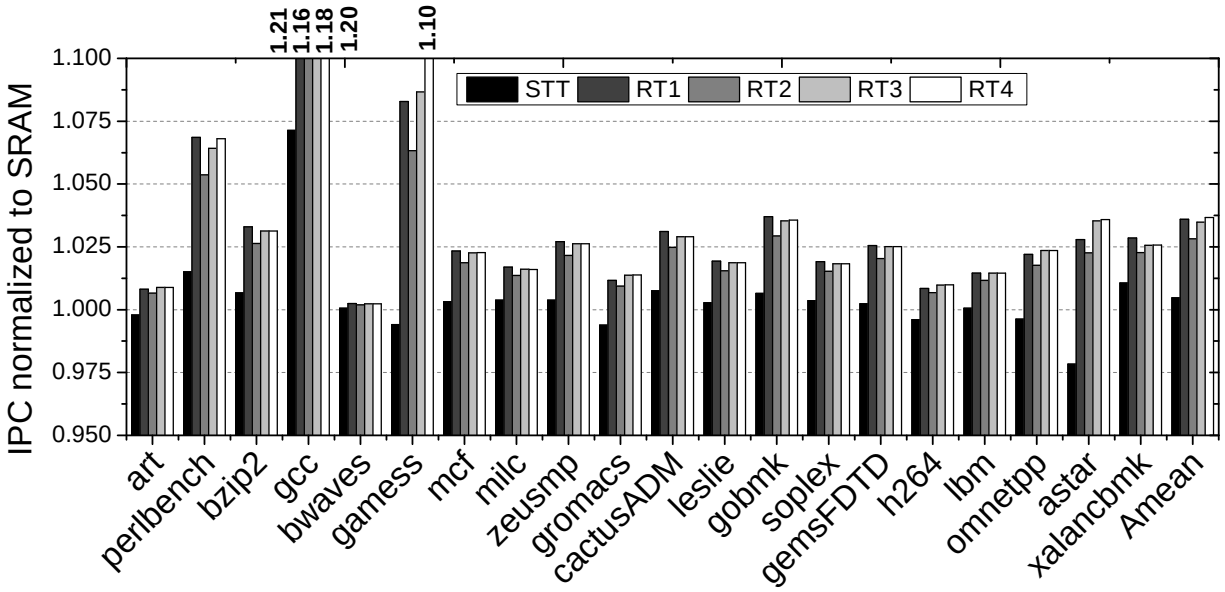


Figure 62: Performance comparison among various memory technologies.

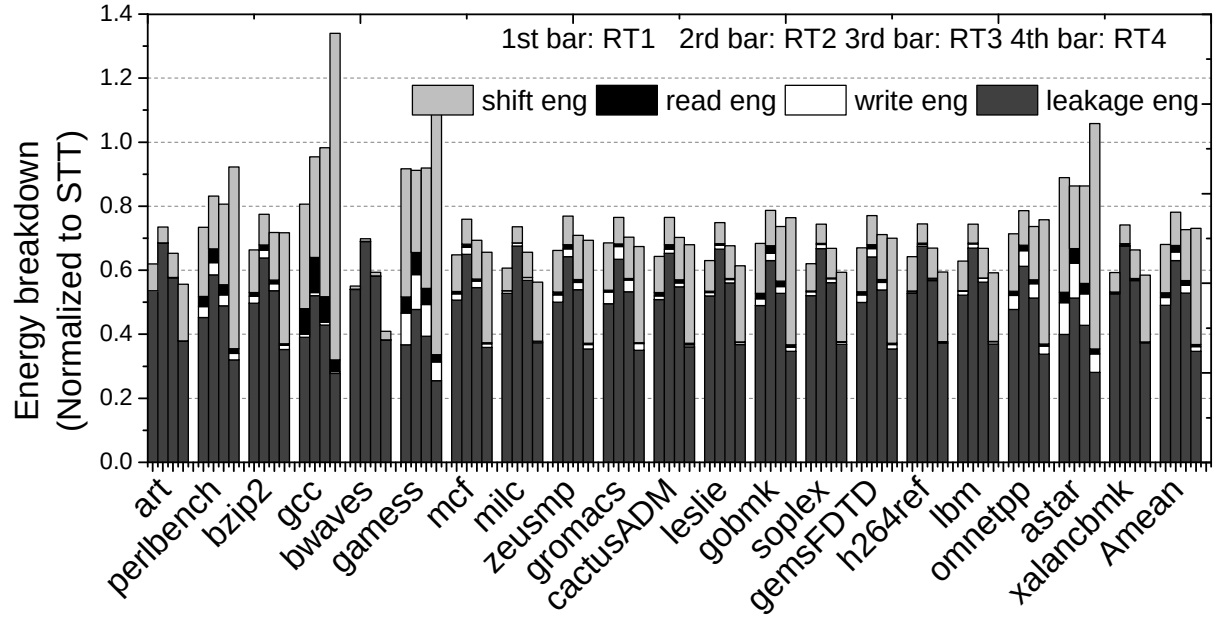


Figure 63: Energy comparison among various memory technologies.

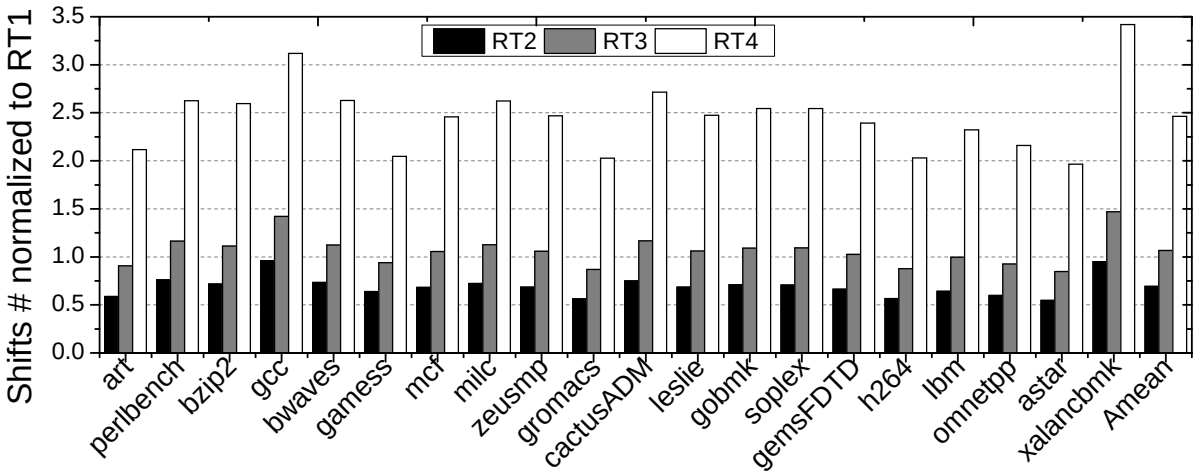


Figure 64: Shift number comparison among various racetrack memory structures.

7.2 OPTIMIZED RACETRACK LLC ARCHITECTURE

The aforementioned racetrack shifts and the associated performance and energy overheads are resulted by racetrack's fundamental structure so that device and circuit design cannot solve the issue. Thus, in this work, we explore the design space of the racetrack based LLCs at architecture level and propose a two-step optimization methodology.

First, considering that the shift overhead is tightly related to array designs, we propose ***a mixed array organization*** with hybrid-port and uniform-port arrays favorable to read-intensive and write-intensive cache blocks, respectively. Since the array organization shall be predefined at the design time and cannot be changed afterwards, we consider it as a static optimization technique. On top of it, a resizable cache access strategy is applied to adjust the racetrack usage and shift overhead upon runtime workload requirement. The second-step dynamic optimization method is called as ***resizable racetrack cache***. The details of our proposed techniques and the corresponding cache data management are explained and discussed in this section.

7.2.1 Mixed array organization

Observation 1: Impact of array organization on shift overhead. Our analysis in Section 7.1 shows that the types and the distribution of the access ports significantly affect the runtime shifts. The usage of R/W-ports supporting both read and write operations is easy and flexible while R-ports are limited to read operations only. However, due to its small area, more R-ports can be integrated into an array, leading to shorter distance between access ports and less racetrack shifts. Based on this observation, we may partition a racetrack memory design based on request patterns. More specific, the hybrid-port array with many R-ports are more favorable to read-intensive cache accesses, while the data with more frequent writes or random access patterns shall be allocated to the uniform-port arrays.

Observation 2: Instruction accesses vs. data accesses. As examples, Figure 65 shows the read and write numbers of the first 500 access requests of four benchmarks selected from SPEC CPU 2006 suite [90]. The statistical results of both *instruction requests* (I) and *data requests* (D) in LLC are shown in the figure. Here, *load* and *update* operations can be

regarded as cache writes. Once an instruction is loaded into cache, the cache block could be accessed (read) many times. Accordingly, Figure 65(a) demonstrates a small number of loads but much more read accesses. In contrast, the read and write frequencies of data blocks shown in Figure 65(b) are more balanced. Some workloads, *e.g.*, `gromacs`, even show an opposite trend, that is, more writes than reads.

Observation 1 encourages to combine different array organizations optimized based on access behaviors in racetrack LLCs. In fact, the cache block access frequency has been widely used in STT-RAM design for performance improvement and write energy reduction [**?**]. These techniques utilize the cache block access history to classify read-intensive and write-intensive blocks. The similar design philosophy has also been applied to racetrack memory [39]. However, the cache block classification based on access history is more or less lagged behind, so predication cannot be quite accurate. Moreover, the initial placement of cache blocks are also very important, which cannot be handled by previous techniques. On the other hand, *Observation 2* shows significant difference in access patterns of instruction and data requests. Therefore, we propose ***a mixed array organization*** consisting of both hybrid-port and uniform-port arrays for racetrack LLCs.

Figure 66(a) illustrates the usage of the proposed mixed array organization in a LLC splitting instruction and data caches. The instruction region (*I-region*) is constructed with hybrid-port arrays since instruction requests have more read accesses. The uniform-port arrays are used for data blocks with diverse and random access patterns, named as *D-region*. A simple controller bridges the upper level caches and LLC and direct cache requests to the corresponding regions. For example, if a miss in L1 data cache occurs and eventually requests data from main memory, the data block in LLC will be loaded into the D-region. Similarly, an instruction block initially goes to the I-region. Expanding the design to the LLC in a 3-level cache hierarchy requires an extra bit for each cache block, indicating if it is a data or instruction. The split cache design separates I-region and D-region so that they can be optimized and configured with different associativity and set numbers. However, the data migration in between is not allowed.

We can also apply the mixed array organization to unified shared LLC widely adopted in the state-of-art processors [32, 33, 34]. A few associative ways of the LLC are constructed

with hybrid-port arrays while most of them use the uniform-port arrays. For convenience, we still name them as *I-region* and *D-region*, respectively. The design prefers to allocate an instruction to I-region and a data block to D-region by using two separated *least-recently used* (LRU) queues. The unified cache structure can facilitate block migration across the whole LLC. For instance, an extreme read-intensive but non-write-intensive data block can be migrated into I-region in the case that it is not heavily accessed.

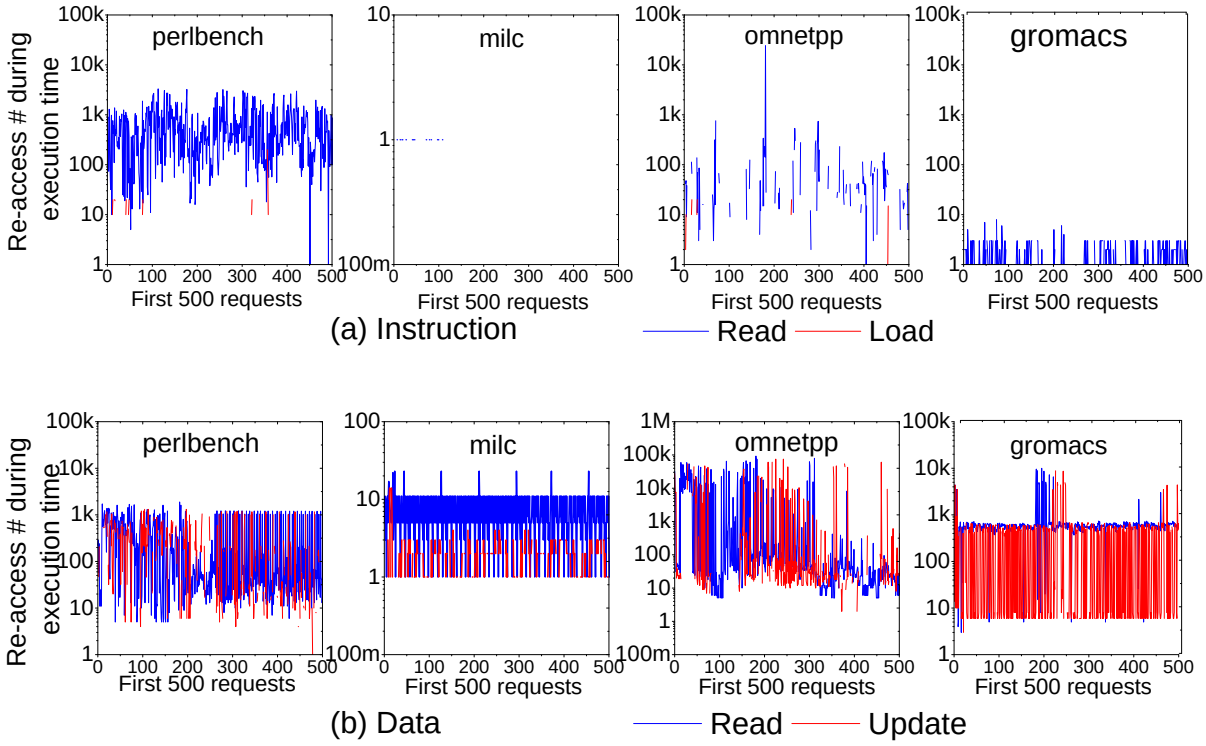


Figure 65: Cache re-accesses of first 500 requests: (a) instruction requests; (b) data requests.

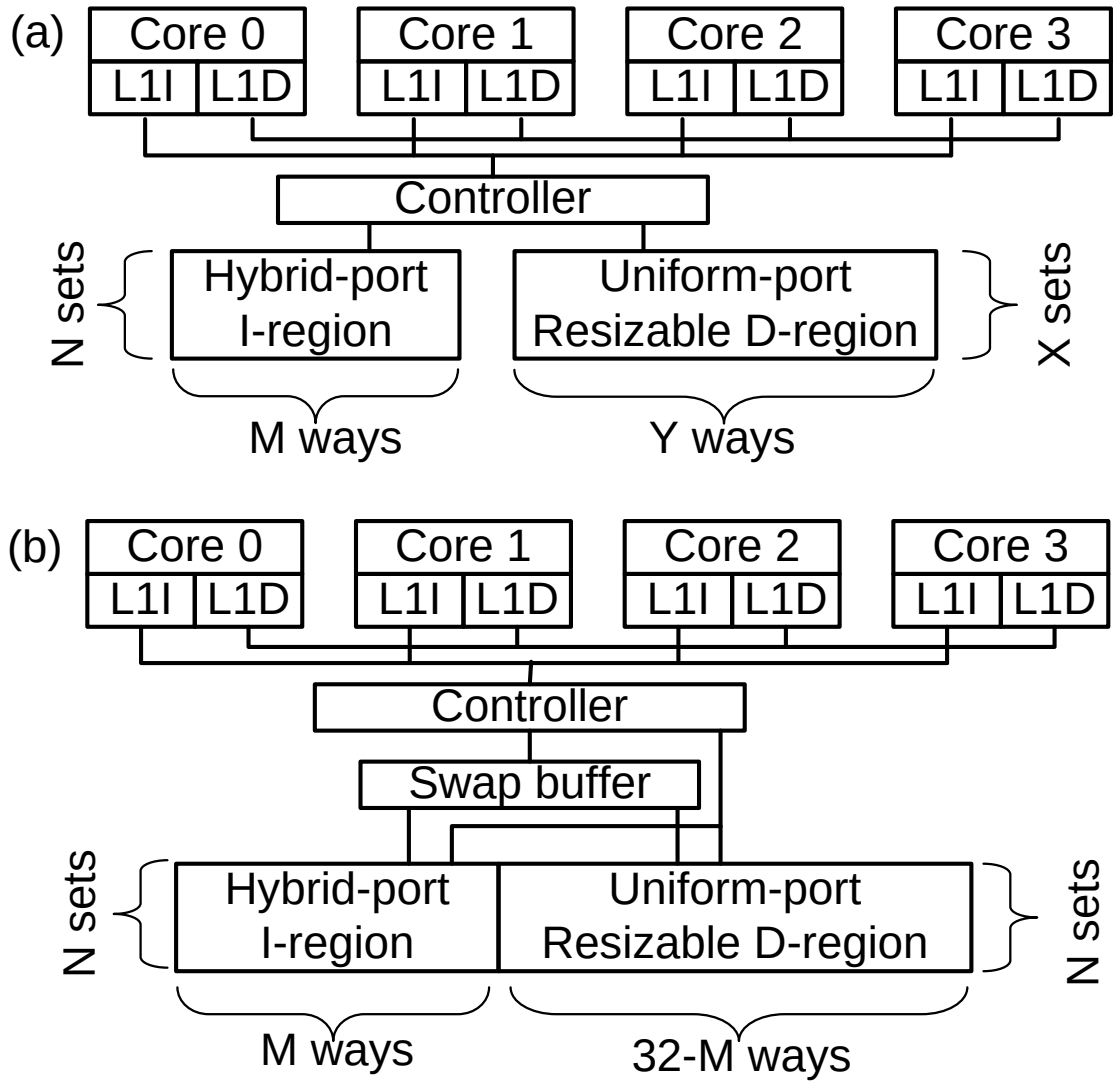


Figure 66: The mixed array organizations of racetrack LLC: (a) I/D split LLC; (b) unified shared LLC.

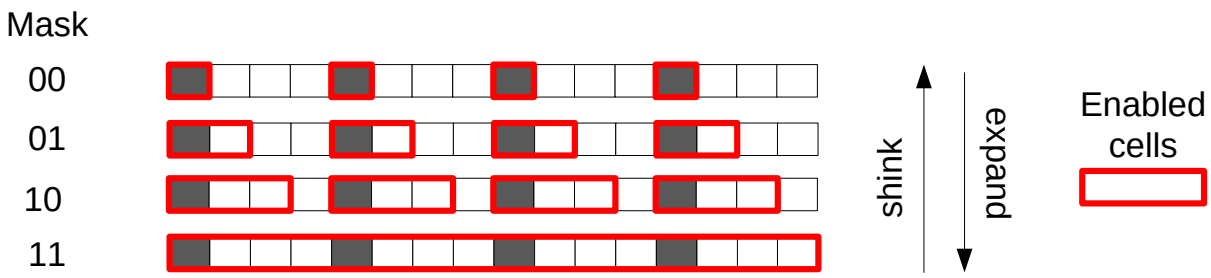


Figure 67: Resize a racetrack with a bit address mask.

7.2.2 Resizable racetrack cache

Observation 3: Resizing racetrack nanowires. Resizing a racetrack nanowire can be simply realized by using a bit address mask to manipulate and restrict the shift distance. Figure 67 depicts an example in which an access port corresponds to four magnetic domains. By tuning the bit address mask, the shift distance is limited and hence only partial bits can be accessed. For instance, setting the mask to '00' enables only the magnetic domains right on the access ports. No extra shift is needed when accessing the cache but the allowable capacity is only 1/4 of the original design. Unlike the previous SRAM architectures that change array size by turning on/off power supplies [35], the racetrack resizing is independent on power supply control and can be implemented in a much finer granularity.

Observation 4: Unbalanced usages of cache blocks. The usage of cache block in LLC is usually strongly biased. Figure 68 calculates the number of sets that hold zero access during the whole program runtime of four selective benchmarks. Different applications reveal the different set usages, which also change with time. Similarly, Figure 69 is the statistics of the number of ways without any cache accesses. For instance, the last bar of `art` represents that in a 32-way 4MB LLC, 3559 out of total 4096 sets have more than 28 ways not being visited at all. In other words, reducing the cache to 4 ways (8× smaller) might not degrade the runtime of `art` much. Also, the way usage varies across all the sets and all the benchmarks.

Considering the unbalanced usage of cache blocks in LLC, we propose ***a resizable cache design*** by leveraging the intrinsic resizability of racetracks. Conflict miss rate and

shift number are used to control the cache shrinking/expanding. Since most of the access ports in the hybrid-port arrays are R-ports, the optimization space of I-region is very small. Thus, we utilize the resizable design only to the uniform-port arrays in D-region.

The capacity of racetrack caches can be adjusted in sets or ways, which is determined by the physical-logic mapping styles shown in Figure 7.

Resizable-set racetrack design

The racetrack array based on the physical-logic mapping in Figure 7(a) supports set resizing. Figure 70 illustrates the array organization in which racetracks with a moderate length of 16 bits are adopted for illustration purpose.

Limited by hardware cost of power gating, the SRAM based caches in [71] tune set usage globally. The proposed resizable-set racetrack cache can easily achieve a much finer control by assigning each array with its individual set mask. The set mask configuration is determined by the access port distance and the resizing options. For instance, the example in Figure 70 enables 4 of the 16 sets when the set mask is '00'. If the set mask is '11', the whole array can be accessed.

To adaptively reflect the change of set accesses in spatial distribution during execution, we propose to dynamically arrange the set usage. Our design periodically evaluates the cache efficiency based on miss rate and shift number and determines set expanding or shrinking (refer Section 7.2.2). As shown in Figure 70, the set resizing involves the adjustment of the tag and index of memory address. We divide the index into the global and local selections: the global bits determines sub-array while the local bits together with the set mask points the target set. Consequently, a memory address could be directed into difference physical locations due to set resizing. An example is illustrated in Figure 71.

At time M, a cache write to the given memory address initializes the cache block of *Set 3* and its status becomes *dirty*. Assume that the array expands to provide higher capacity at the end of Interval 1 so that the address decoding changes. Combined with the set mask, the last three bits of the index (instead of the last two bits in Interval 1) will be used for local set selection. A read request to the same address at time N within Interval 2 goes to *Set 7* and incurs a cache miss. In such a situation, the data shall be loaded from main memory. To properly reflect the cache write at time M and keep the data integrity, the dirty

data blocks must be evicted and written to main memory during set expansion. Similarly, the set shrinking results in reduction of cache capacity and change in address decoding. So cache block eviction is necessary in resizable-set design. Besides the overheard of moving dirty data to main memory, the reduction of valid cache data due to set resizing potentially increases the cache miss rate and degrades system performance.

Resizable-way racetrack design

Figure 72 shows the racetrack array organization in which the way number can be dynamically changed. The design utilizes the physical-logic mapping in Figure 7(b). We use the way mask to control the way accessability. The address decoding remains the same as regular cache design and the tag of a memory block determines the cache block selection. We adopt a separated tag array in STT-RAM technology by following previous racetrack architecture [39, 40]. So the tag array can be accessed and updated promptly, no matter the corresponding cache block is enabled or disabled.

The data management in a resizable-way design is easier than a resizable-set array. First, expanding way number will not cause data integrity issue and hence does not require special treatment. Second, it is not necessary to abandon the data belonging to the disabled ways when shrinking way usage. Racetrack is a nonvolatile memory technology. The data in the disabled ways are still valid. The design can check the tag array for each cache request, no matter if the cache block is disabled or not.

Figure 73 illustrates the three situations of accessing a disabled cache block. (1) A read request from the upper level cache indicates the cache block to be re-used (update after read). If the request is directed to a disabled block, we need read it out, move it to an enabled way, and invalid the status bit of the disable block. (2) An eviction from the upper level cache introduces a write to the LLC. The cache block in a disabled way is label as *dirty* and the new data is directly written to an accessible slot. (3) Once a cache miss occurs, the data from main memory is restrictively placed at an enabled way, replacing an invalid or LRU block.

Figure 74 depicts the detail of data movements when a hit occurs in a disabled way. Most of such data migrations happen right after decreasing the way usage, relocating the access-intensive cache blocks from the recently disabled ways to the enabled ones.

Resizing policy

We propose two cache resizing schemes that can adaptively arrange the racetrack usage upon the change of cache accesses in spatial and time domains. Since our design tends to balance the overheads induced by racetrack shifts and cache misses by reduced capacity, both criteria are used for resizing control. The compulsory misses are not related to cache capacity and associativity and hence are excluded from the resizing condition. Only the misses associated with block replacement are counted.

Figure 75(a) shows the first resizing policy with fixed evaluation interval. Each state represents a size configuration, which determines the set/way mask. In this example, we assume four possible configurations: '00' corresponds the the smallest capacity and '11' indicates all the cache blocks are accessible. The resizing condition is checked periodically at the end of a fixed monitor interval. Our design enlarges cache capacity as miss rate increases. The control of cache shrinking shall be more careful, which is triggered when observing big shift number and obvious miss rate reduction. It is not necessary to reduce cache size when the shift number is low, which potentially results in high miss rate.

A more flexible and simpler resizing policy is shown in Figure 75(b). In this scheme, N sequential conflict misses trigger the set/way expansion. When M hits occur in sequence and the total shift number in this period is smaller than the threshold SB, some sets/ways are disabled. The duration of evaluation period depends on the cache access pattern, making the resizing more efficient and responsible to real time applications. This policy provides a more reasonable resizing timing, but potentially occur more frequent changes in cache capacity.

The two resizing policies have different hardware and control overheads. Assuming that the fixed-interval scheme is applied to a resizable-way cache. All the miss rate counters and shift counters corresponding to each set shall be examined at the end of an evaluation interval. The operation cost is pretty high, especially when the evaluation interval is small. The scheme with dynamic evaluation intervals is better. Counters are needed only for those sets under accessed. And the frequency to check and update these counters depends on the access frequency and resizing threshold configuration.

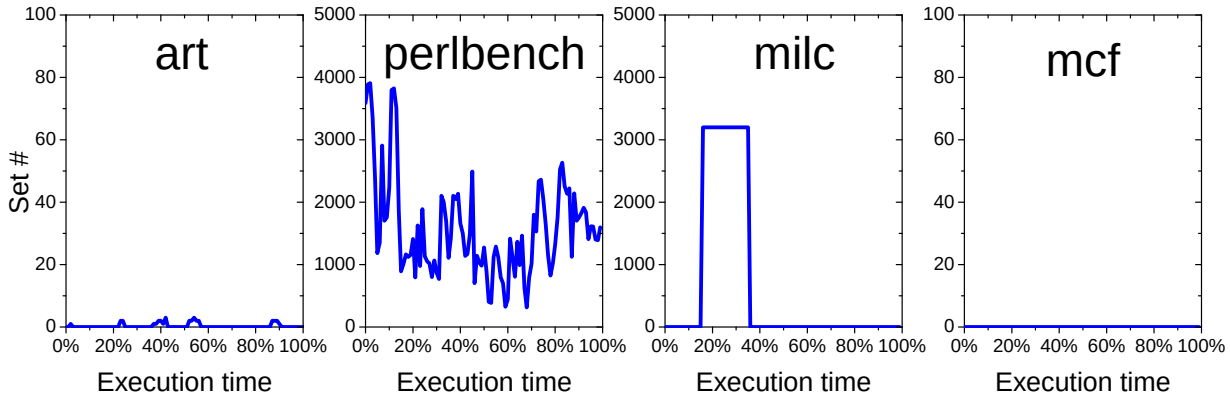


Figure 68: Number of sets having zero access during runtime.

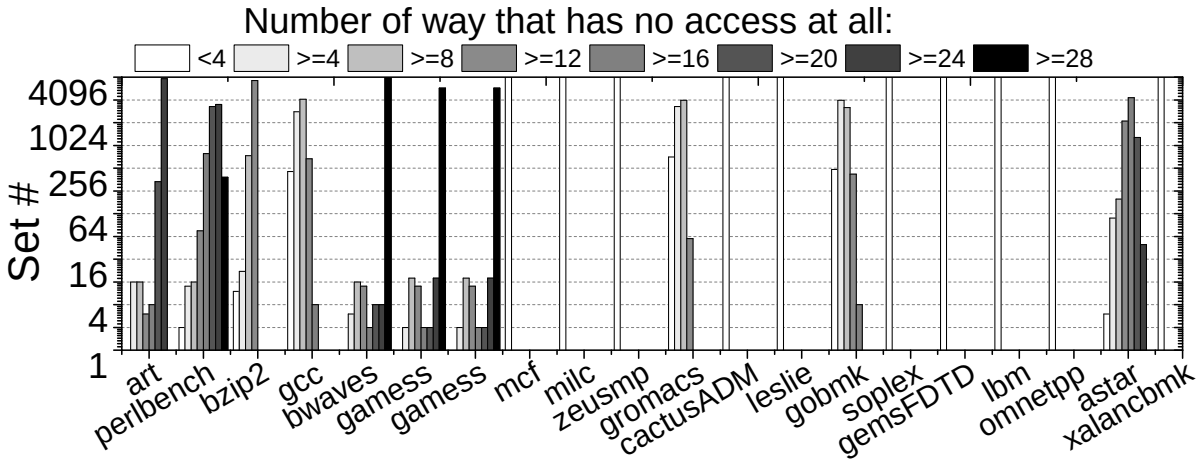


Figure 69: Number of ways having no access during runtime.

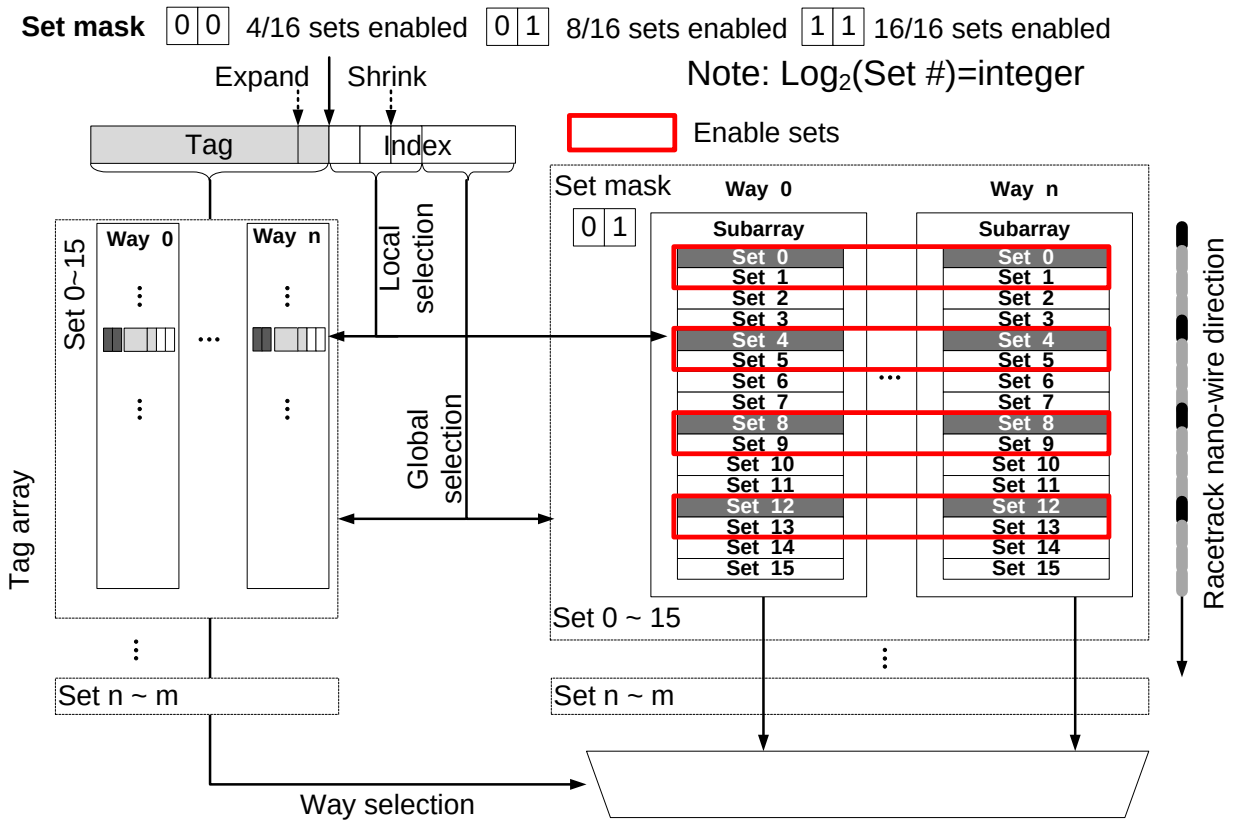


Figure 70: Resizable-set array organization.

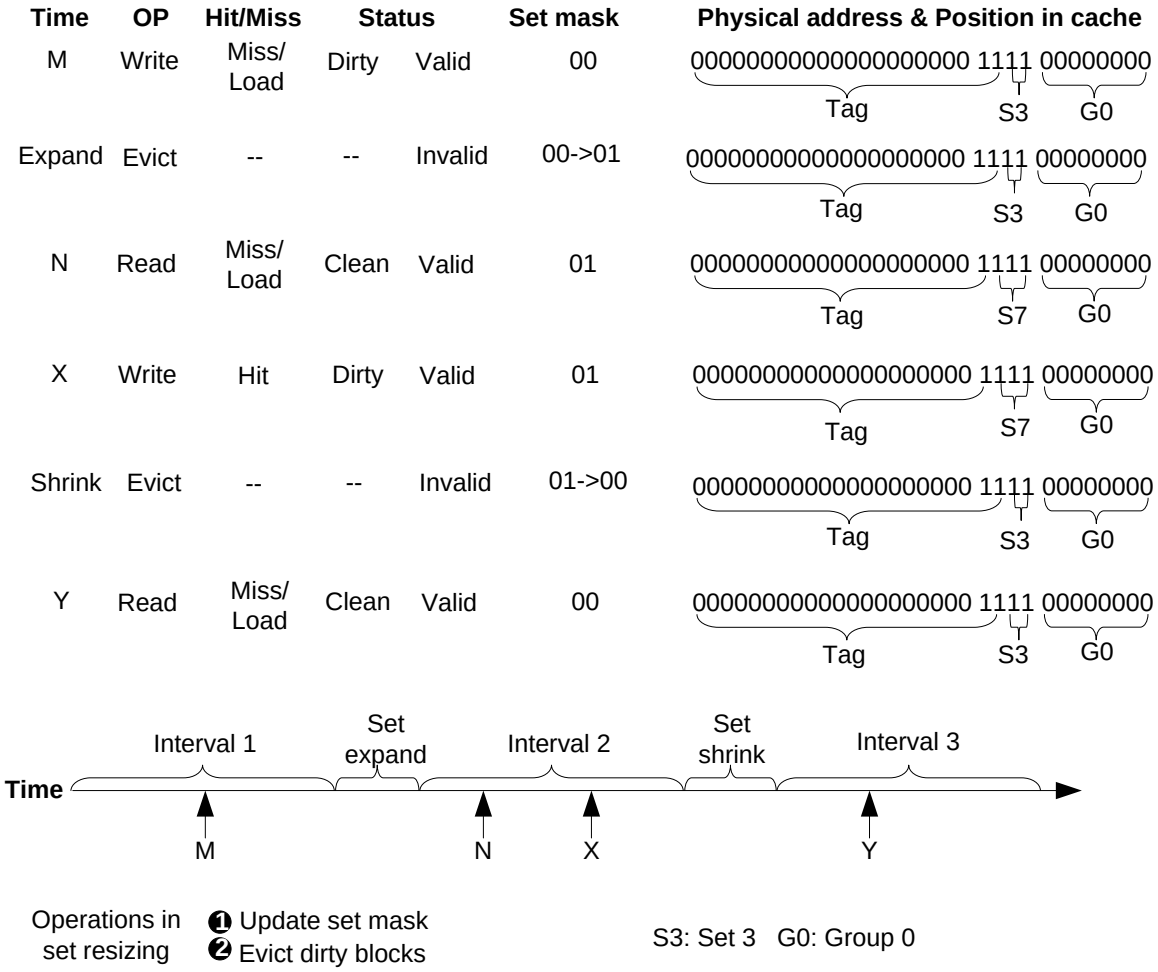


Figure 71: An example that one memory address is accessed multiple times across set resizing.

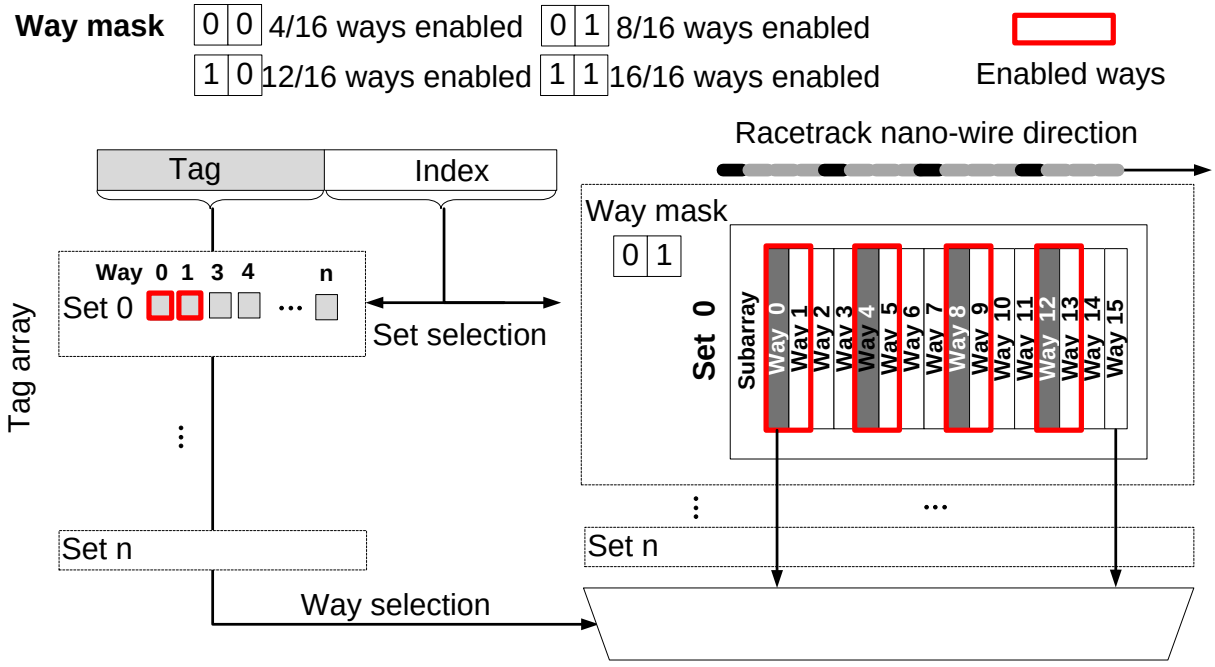


Figure 72: Resizable-way array organization.

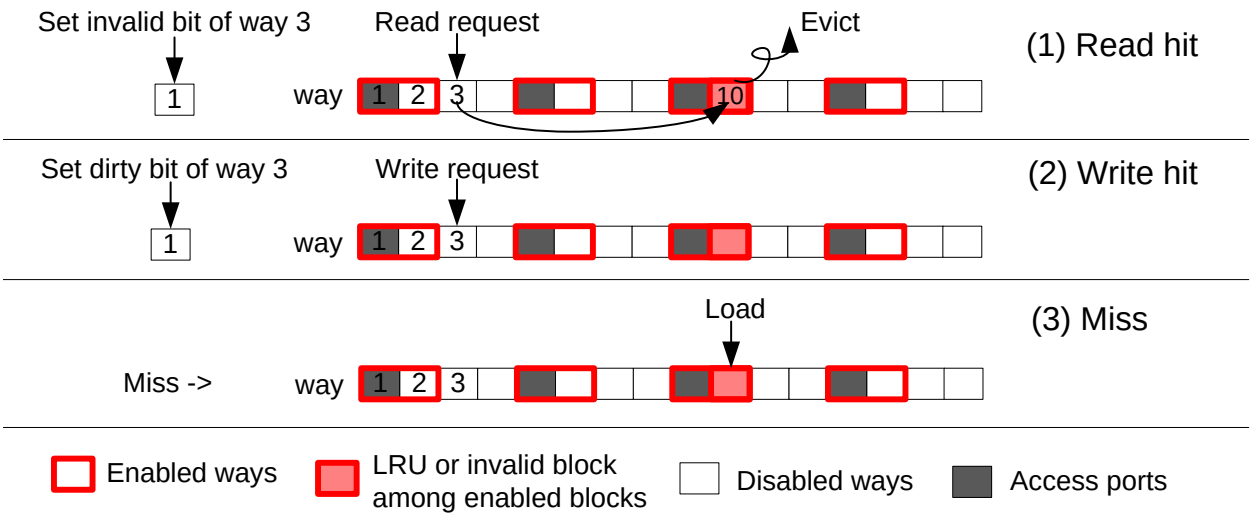
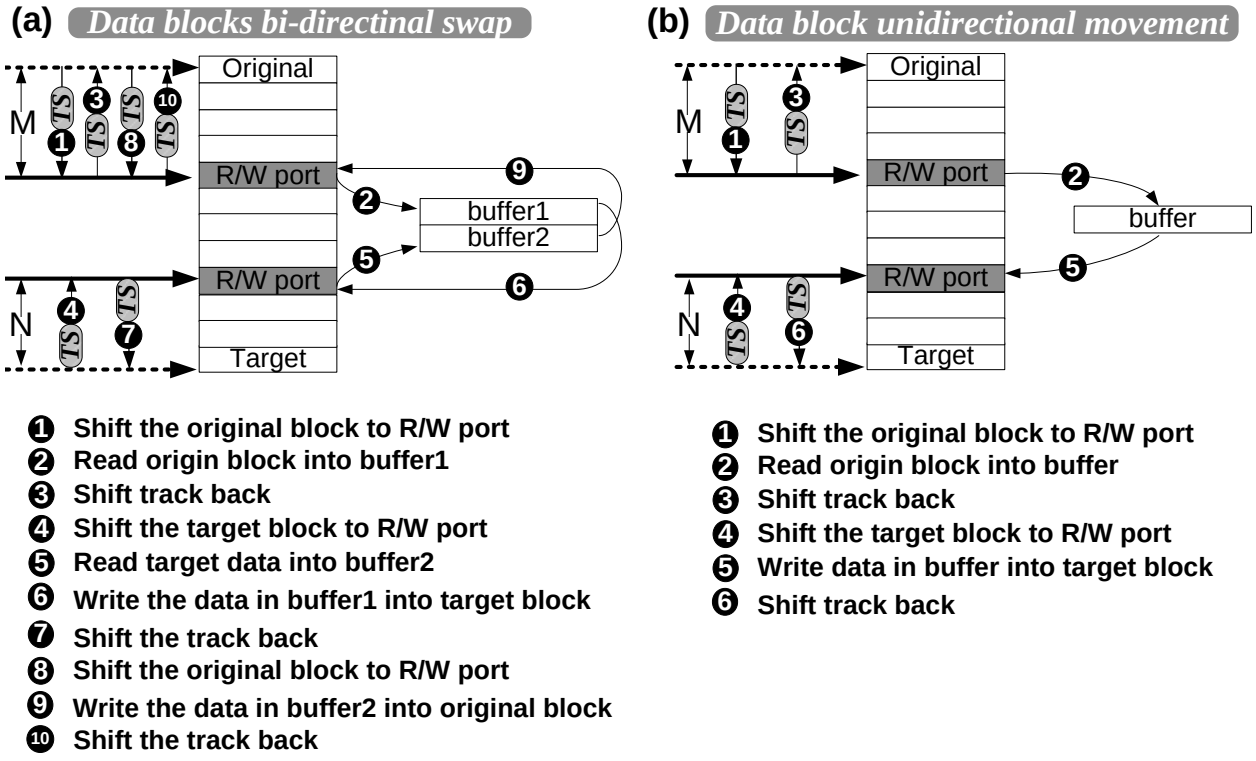


Figure 73: Three situations to access a disabled way.



❶ Shift the original block to R/W port
❷ Read origin block into buffer1
❸ Shift track back
❹ Shift the target block to R/W port
❺ Read target data into buffer2
❻ Write the data in buffer1 into target block
❼ Shift the track back
❽ Shift the original block to R/W port
❾ Write the data in buffer2 into original block
❿ Shift the track back

❶ Shift the original block to R/W port
❷ Read origin block into buffer
❸ Shift track back
❹ Shift the target block to R/W port
❺ Write data in buffer into target block
❻ Shift track back

Figure 74: Data movement for a hit in a disabled way.

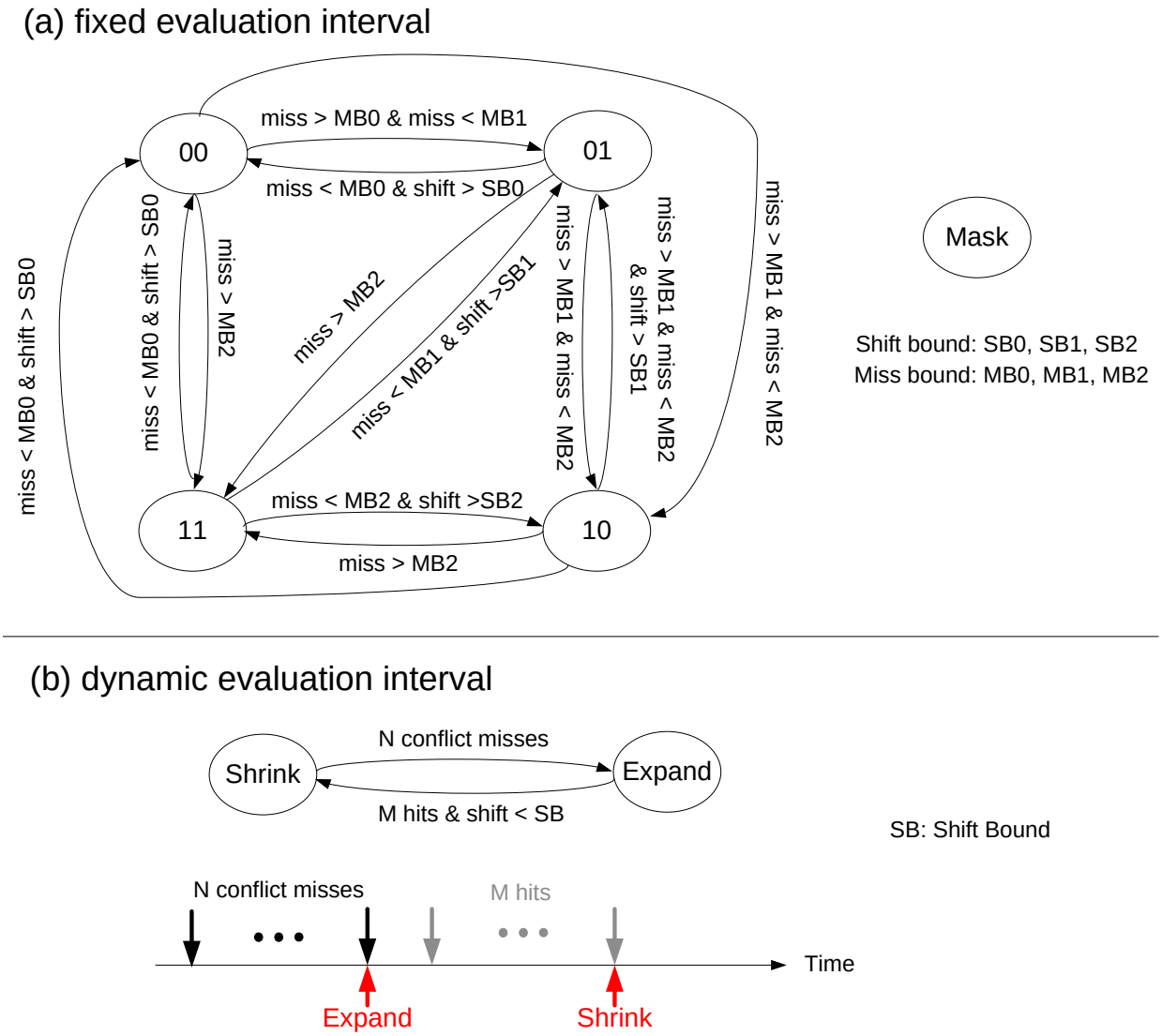


Figure 75: Two resizing policies with the fixed (a) and dynamic (b) evaluation intervals.

Table 15: The CPU Configuration

CPU	2GHz, 2 Cores, out-of-order, 2-way issue
L1 Cache (SRAM)	32K I/D, 32B cache line, write-back, 2-cycle R/W, private
L2 Cache	4MB, 32B cache line, write-back, shared
Main Memory	1GB, 400 cycle

Table 16: The Configuration of Racetrack LLC in Mixed Array Organization

Baseline Racetrack LLC	I-region	D-region
Capacity & Design	0.5 MB, RT1	3.5 MB, RT4
Latency[1]	read: $(2+1)$ cycles; write: $(2+10)$ cycles	
Distance of adjacent read ports[2]	3	8
Distance of adjacent write ports[2]	8	8
Dynamic energy	0.084 nJ/read; 0.62 nJ/write	
Leakage power	48 mW	
Shift energy	0.62 nJ/shift	
Data swap energy[3]	$(1.4 + 4 \times M \times 0.62 + 2 \times N \times 0.62)$ nJ	
Data move energy[3]	$(0.7 + 2 \times (M + N) \times 0.62)$ nJ	

[1] *Latency = peripheral latency + cell latency*
[2] *The port distance is represented by the number of magnetic domains.*
[3] *M (N) is the distances between the original block (target block) and the nearest access port. Refer Figure 16(b) for details.*
[4] *One shift means a racetrack moves a unit distance of one magnetic domain. We assume each shift takes one clock cycle* (0.5 ns) [39].

7.3 EVALUATION OF THE PROPOSED RACETRACK LLC

In this section, we evaluate the effectiveness of the proposed racetrack LLC design with the mixed array organization and the resizable sets/ways, compared to the LLC designs based on SRAM, STT-RAM, and previous racetrack architectures. The iso-capacity replacement of 4MB LLC is applied.

7.3.1 Experimental setup

Previously we summarize the latency and energy parameters of SRAM, STT-RAM, and racetrack memory in Table 13 and Table 14. The parameters used in this work are obtained from the modified NVsim. The domain wall shifting energy was calculated based on micro-magnetic simulations.

We modified the cycle-accurate simulator MacSim [91] for real-time microarchitecture estimation. The processor configuration and memory hierarchy can be found in Table 15.

The SPEC CPU2006 benchmark suite [90] is adopted. In each simulation, we fast-forward 500 million instructions and run 1 billion instructions.

Various LLC designs are thoroughly evaluated in terms of the system *instruction per cycle* (IPC) performance, the energy consumption of LLC, the cache miss rate, and the racetrack shifts. The simulation results are normalized to the racetrack design based on the uniform RT4 arrays (see Section 7.1).

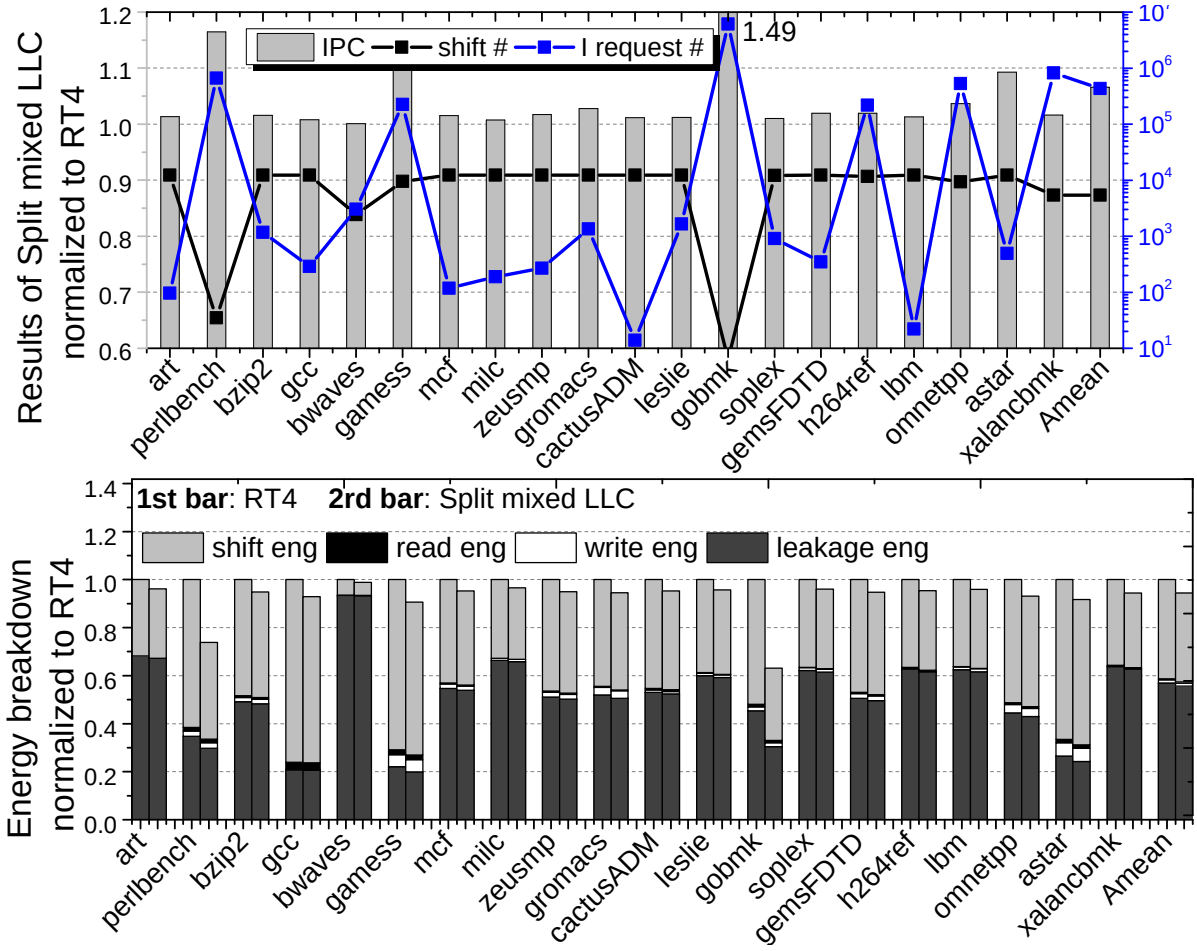


Figure 76: The IPC performance and energy of I/D split racetrack LLC.

7.3.2 Racetrack LLC with mixed array organization

The baseline 4MB racetrack LLC with mixed array is partitioned into a 512KB I-region and a 3.5MB D-region. The detail configuration parameters are summarized in Table 16.

Figure 76 shows the performance and energy consumption of the I/D split LLC. On average, the I/D split organization achieves 6.6% improvement in IPC performance over the uniform RT4 design. The major benefit comes from the reduced shift operations. On average 12.7% of racetrack shifts are removed in the mixed array design. As expected, the workloads with more instruction requests, such as `perlbench` and `gobmk`, reveal higher performance improvement. On the other hand, the locality of instruction request also plays an important role in performance improvement and shift reduction. If a instruction with frequent accesses is located to a cache block far from access ports in I-region, it still cannot gain much from the mixed array design. For example, distinguishing instruction requests do not results in significant IPC increase in some benchmarks, such as `h264ref` and `lbm`. This is because of the tradeoff between the locality and the access frequency. Compared to the uniform RT4 design, the fast execution and the reduction of domain shifts result in 6.7% of the energy consumption in the I/D split LLC as shown in Figure 76(b).

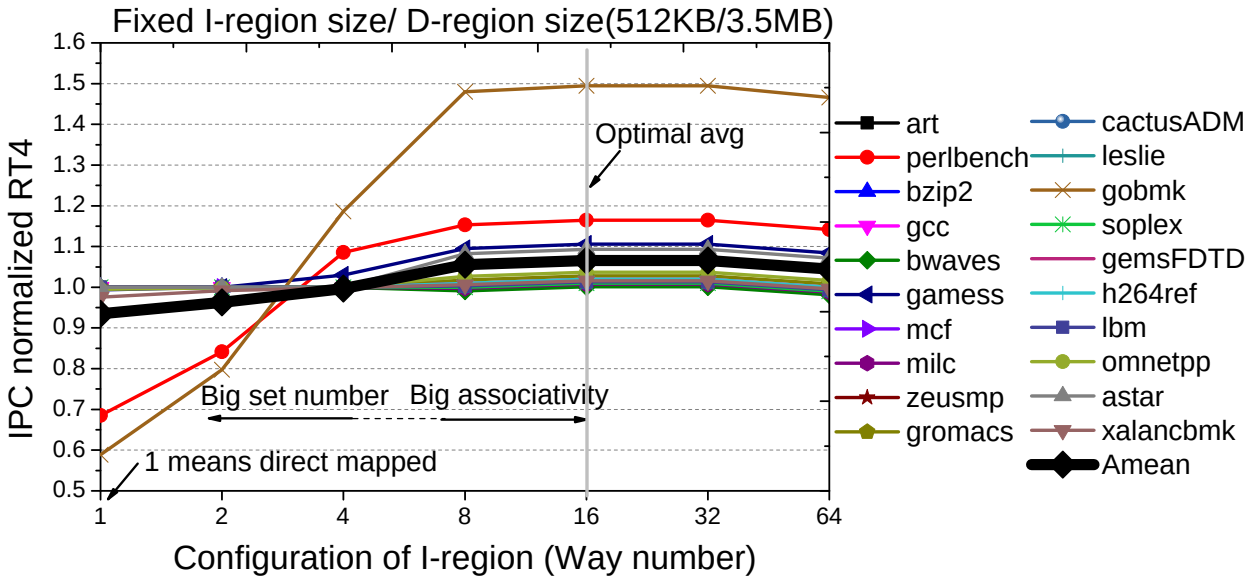


Figure 77: Sensitivity analysis of I/D split racetrack LLC.

Since the I-region is separated from the D-region, properly adjusting its associativity (sets vs. ways) could further enhance the system performance. We analyze the sensitivity of IPC performance on I-region configuration. The result is shown in Figure 77. In general, system

executes faster as the I-region associativity increases. The benchmark `gobmk` demonstrates a strong dependence on the I-region associativity. When the I-region associativity exceeds 16, the IPC improvement becomes saturated or even starts degrading in some benchmarks. The optimal I-region configuration is 16 ways, which is corresponding to the results in Figure 76.

We also evaluated the effectiveness of the mixed array organization in the unified shared LLC design. The configuration of I-region and D-region remains the same as the I/D split LLC in Table 16. Figure 78 summarizes the simulation results. Comparably, the enhancement of the mixed array organization in unified shared LLC is less significant. On average, only 2.0% IPC performance improvement is obtained across all the benchmarks.

The impact of the I-region and D-region distribution in the unified shared LLC is investigated by increasing the number of ways belonging to the I-region (that is, decreasing the capacity of D-region). Figure 79 shows the simulation result of the sensitivity study. Since the unified share LLC requires the consistent set numbers in all the associative ways, a small I-region results in significant performance degradation. Increasing the way number of I-region helps relax the situation. Examples include `perlbench` and `gobmk`. However, further squeezing D-region capacity can also lead to performance degradation in some workloads such as `lbm`. When the I-region expands to 32 ways, the LLC design turns to be a uniform design with RT1 arrays. Overall, our simulation indicates an optimal configuration consists of 4 ways of I-region and 28 ways of D-region, which is corresponding to the results in Figure 78.

7.3.3 Resizable cache

We utilize the resizable-set (*r_set*) or resizable-way (*r_way*) designs to the uniform-port arrays in D-region of I/D split racetrack LLC. In this section, the effectiveness of the proposed adaptive scheme is examined by comparing with the *history based way reorder* (HBWR) scheme which monitors the racetrack position and dynamically controls its movement [39]. All the related hardware and energy overheads have been included in the simulations.

Figure 80 compares the IPC, shift number, and energy consumption of different designs. As aforementioned, the implementation of r_way is simpler than r_set. Moreover, r_set generates many unnecessary evictions, resulting increase of LLC miss rate. Accordingly, the

r_way design outperforms the r_set one in term of IPC performance. Compared to the baseline RT4, on average 13.2% or 7.2% of system performance improvements are obtained after applying r_way or r_set to D-region of I/D split racetrack LLC, respectively. On top of RT4, the r_way design reduces 57.9% of racetrack shifts while r_set obtain an average 60.7% reduction. The slight difference comes from the uneven usages of sets and ways. The energy breakdowns in Figure 80(c) shows 30.4% and 27.8% of energy savings obtained by r_set and r_way, respectively.

Compared to HBWR that tends to migrate access-intensive data blocks to access ports, r_way is more efficient in shift reduction and removes 31.3% of racetrack shifts. Consequently, r_way obtains 3.0% performance improvement and 15.9% energy saving over HBWR. Though r_set eliminates more shifts and hence consumes less energy than HBWR. However, its IPC performance is slight worse due to the inevitable data evictions when expandig/shrinking set numbers.

In Figure 81, we trace the racetrack usage (the valid blocks) following the available resources (the enabled capacity) for four selected benchmarks in the r_way design. Here, we divide the overall execution time into 100 intervals and snapshot the enabled capacity and the valid blocks at the end of each interval. All the results are represented in percentage of the total cache block number. Since the resizing is passively triggered by the cache capacity requirement and racetrack shift numbers, the adjustment of cache block usage is slightly lagged behind. The benchmark `gcc` obtains higher performance improvement for its relatively lower cache usage and smooth change. In contrast, `art` doesn't benefit much from its low capacity requirement mainly because of the frequent fluctuation in cache size. We set the minimum size as 1/8 of the full cache capacity. Thus, if the real cache usage continues decreasing, the enabled capacity will maintain at the minimum size as shown in the tails of `perlbench` and `gcc`. The comparison of the resizing policies with static and dynamic evaluation intervals for selected benchmarks are shown in Figure 82. Here, the effectiveness of the resizing policies are measured by the percentage of the enabled cache capacity. Our results show that the dynamic policy responses aggressively and actives fast to the data requirement, while the static one with fixed evaluation interval behaves lagged behind.

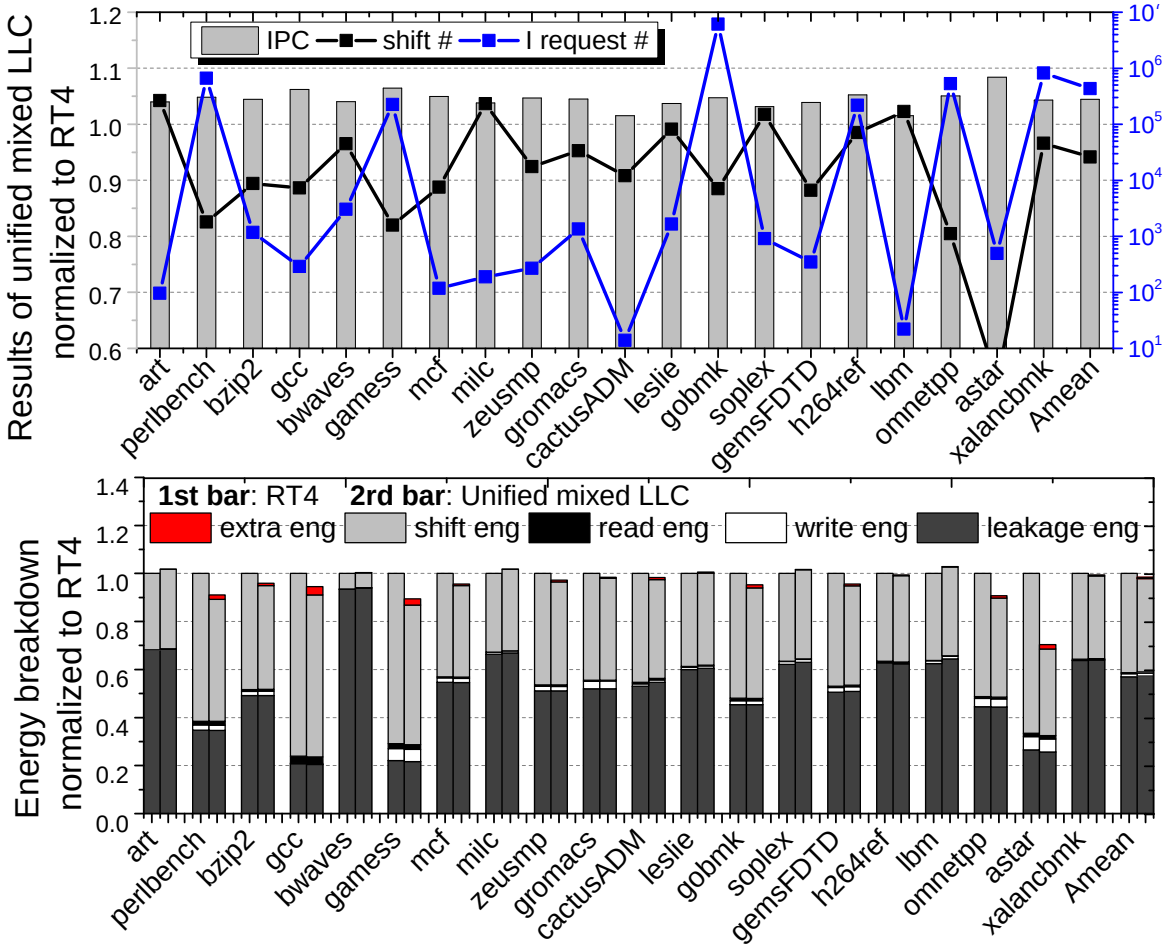


Figure 78: The IPC performance and energy of unified shared racetrack LLC.

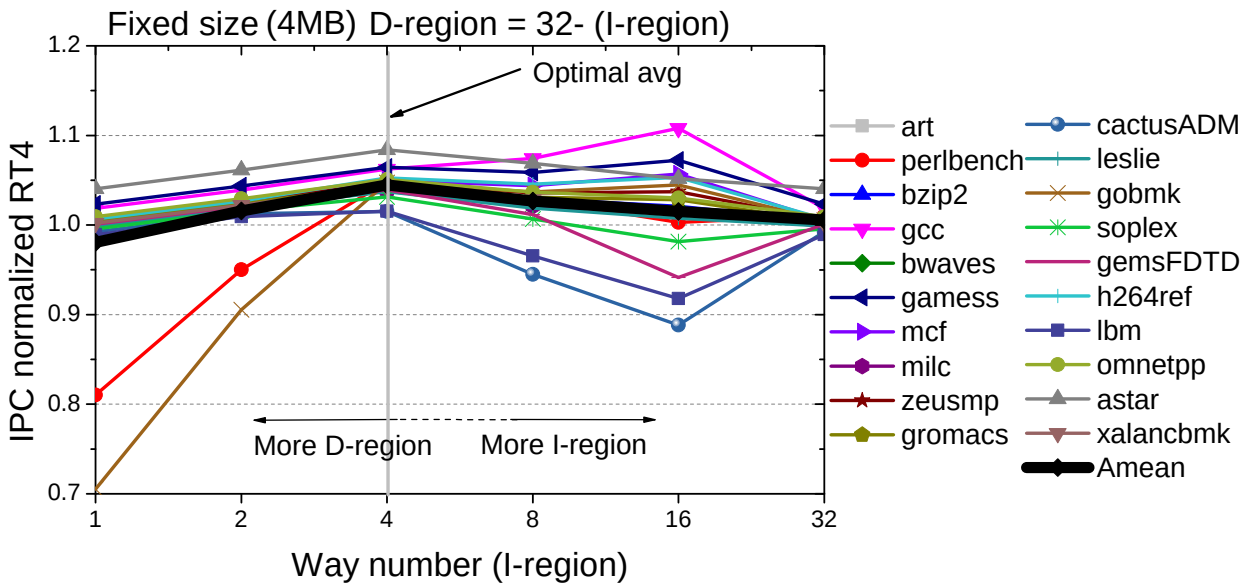


Figure 79: Sensitivity analysis of unified shared LLC.

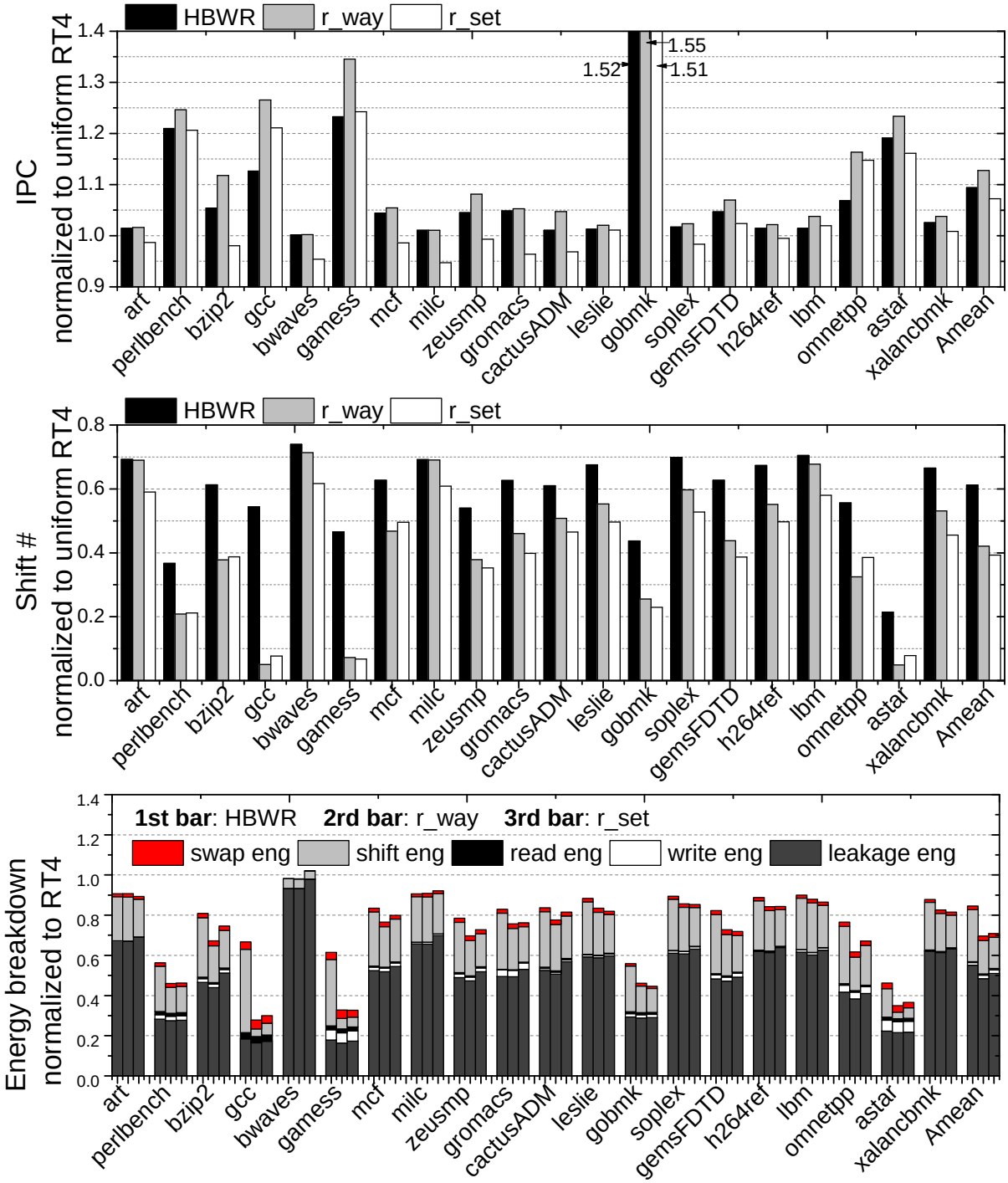


Figure 80: (a) IPC performance, (b) shift number, and (c) energy breakdowns after applying resizable cache.

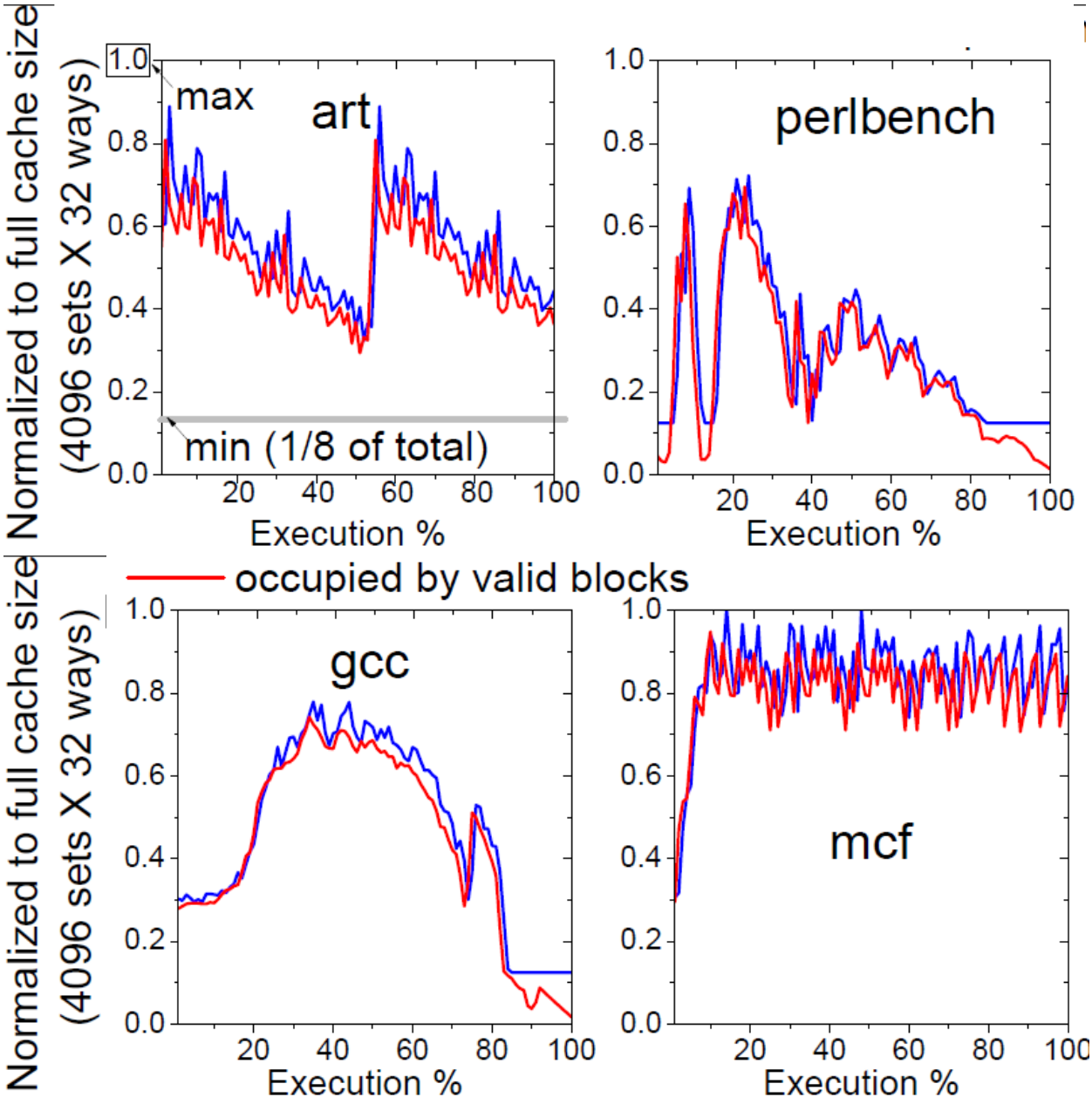


Figure 81: The runtime cache block usage vs. the enabled cache capacity in resizable-way LLC design.

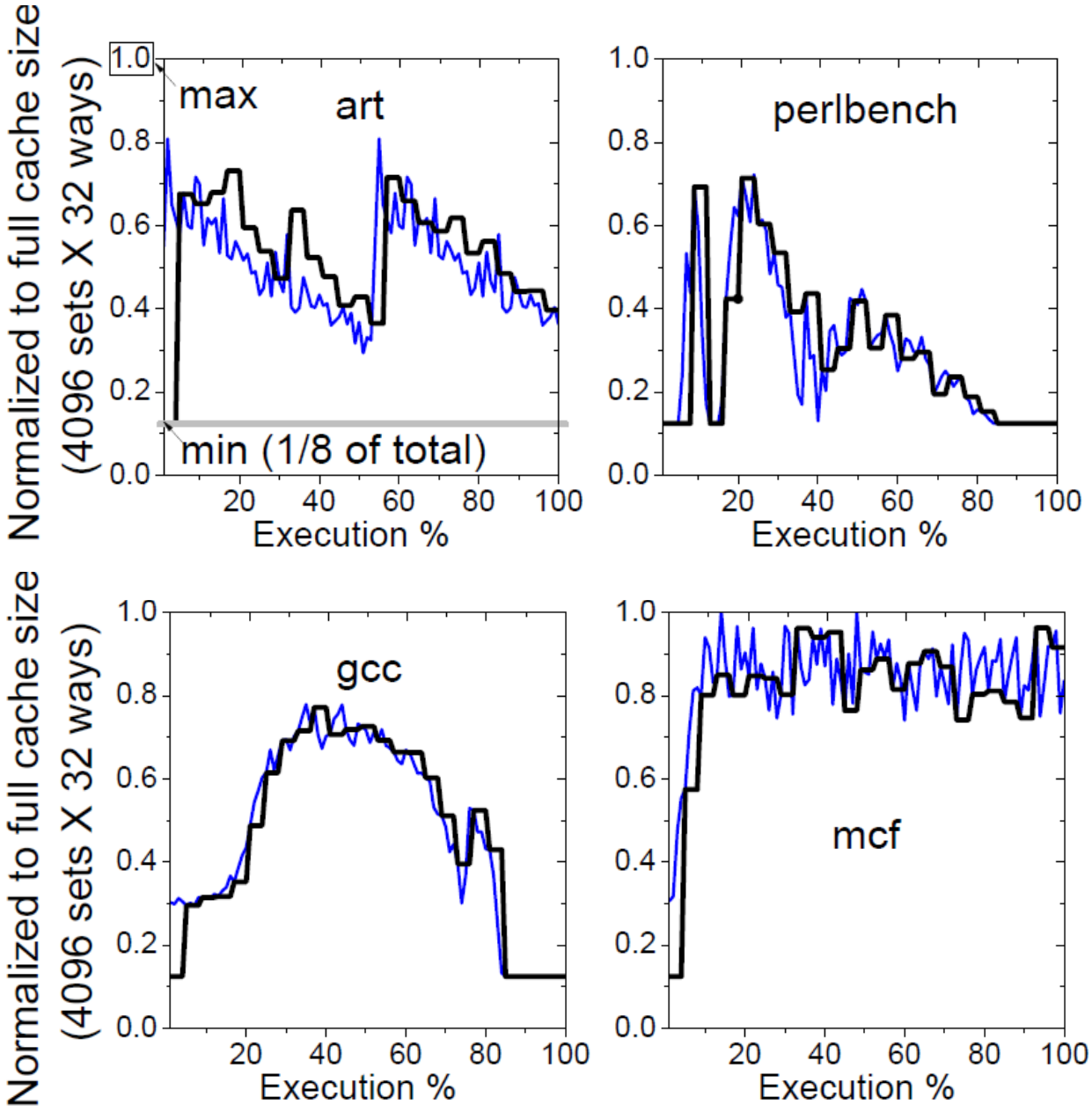


Figure 82: Resizing policy comparison.

7.4 SUMMARY OF CHAPTER

In this chapter, a comprehensive exploration for racetrack memory is performed at architecture level. We summarize and compare all the possible access port designs and the corresponding array organizations.

The simulation results demonstrate its great potential as LLC: on average 3.7% performance improvement over SRAM and 30.0% energy saving over STT-RAM can be obtained.

Our analysis also reveals a strong relationship among racetrack array structure, cache access patterns, and system performance. Based on it, a two-step design methodology for racetrack memory based LLCs is proposed. First, we optimize the cache organization by including both hybrid-port arrays and uniform-port arrays. Instruction requests with more reads can go to hybrid-port arrays with more R-ports while the data blocks with diverse and random access patterns can be directed to uniform-port array. The partition of two types of array can be determined at design time based on pre-evaluation of workloads. Second, based on the unbalanced utilization of cache blocks in LLCs and the simple control on racetrack usage, we propose the resizable cache scheme supporting dynamic set or way selections upon cache access requirement. Two policies with the fixed or dynamic evaluations intervals can be used to control cache capacity.

The simulation results show that the racetrack LLC utilizing both mixed array organization and resizable-way can achieve 13.2% performance improvement and 30.4% energy saving over the baseline in uniform array structure. Compared with the HBWR scheme which can aggressively control the racetrack movement, the proposed design can still improve 3.0% of IPC performance and reduces 15.9% of energy consumption.

8.0 CONCLUSION

8.1 CONCLUSION OF DISSERTATION

This dissertation has investigated many aspects of magnetic material based memory in designing embedded memory systems. Chapter 3 described the STT-RAM cell design tradeoff between switch performance and data retention time. A multi retention level STT-RAM cache hierarchy that trades off the STT-RAM cell's nonvolatility was introduced to enable the full STT-RAM cache hierarchy for energy saving and performance improvement. A memristor-controlled refresh scheme was proposed for the STT-RAM L1 cache to ensure data validity with the minimized hardware cost. Compared to the classic SRAM or a SRAM/STT-RAM hybrid cache hierarchy, our proposal uses only STT-RAM. This can save significant die cost and energy consumption. Moreover, compared to the previous STT-RAM relaxed retention design that only has a single retention level, our design utilizes multiple retention levels, resulting in an architecture that is optimized for the data access patterns of the different cache levels.

Process variation can not be neglected especially when technology is scaling down. Chapter 4 described three process variation aware non-uniform cache access schemes to minimize the cache cell programming overhead caused by cell programming time variety based on the analysis of STT-RAM process variation. Compared to traditional DNUCA which neglects the impact of process variation and read intensive data block, our proposed schemes can achieve a much better performance and be more energy efficient. A conflict reduction mechanism has also been introduced to overcome the drawback of our DPVA-NUCA-2 to fully maximize its advance. With technology scaling, the process variation become more and more severe, we believe that our proposed schemes will become even more attractive.

With the write performance enhanced, read performance of STT-RAM becomes another issue. Chapter 5 discussed the potential issues in STT-RAM based on these *fast-switching* (FS) devices and explored the design techniques. We explored the design implication for FS-STT-RAM with different requirements such as capacity, read/write latency, error rate, energy constraint for different types of STT-RAM based memory systems. A memory architecture has been proposed based on the exploration of the tradeoff among read performance, read current amplitude and read disturbance error of *fast-switching* STT-RAM. Under the support of operating system, the achieved memory architecture can switch between two mode - low power and high accuracy. Software provides the information of required data accuracy level as well as performance requirement. And operating system performs to monitor and deliver the information from software to hardware.

To further explore the density and energy enhancement of the cache hierarchy, another novel magnetic memory technology is introduced. Chapter 6 generally explored the racetrack memory design from different design layers. During the design exploration, we first focus on increasing area efficiency on lower level design layer by reorganizing the physical layout and modifying memory array structure. Then, a flexible hardware architecture is proposed to facilitate the design of access policy and data management. The low power consumption and high integration density of the proposed racetrack memory are highly beneficial for lower level caches, enabling high-performance and low energy computation. The constructed racetrack memory based LLC significantly improve the performance, power consumption and density when compared to SRAM and STT-RAM. Since the Racetrack memory cell array is much smaller than other memory technologies, the pitch mismatch between cell array and row decoder becomes very critical. The proposed architecture reduces the number of rows compared to traditional cache design by slightly increasing the address decoder overhead to mitigate the pitch mismatch problem. Array size become much smaller that the delay and energy cost on H-three is also reduced exponentially. The potential performance degradation is mitigated by the proposed access management scheme. On the other hand, the small peripheral delay also enhances the performance.

Due to the unique storage feature of racetrack memory, more design potential shall be unveiled. Chapter 7 demonstrated the comparison among various racetrack designs and other

memory technologies including SRAM and STT-RAM. The simulation results demonstrate 3.7% performance improvement over SRAM and 30.0% energy saving over optimized STT-RAM obtained by a racetrack memory in a uniform array organization. Two architecture-level optimization techniques for the racetrack memory based LLC are introduced. In the exploration of the first technique, we unveil the relation between the cache access pattern and racetrack memory structure. Then, we proposed a mixed-structure racetrack memory design that allocate data and instruction blocks separately. Moreover, based on the uneven distribution feature of LLC, we proposed the resizable cache design. The proposed resizable cache design can be realized without power gating, but orthogonal to the power gating design. Two different logic organizations are evaluated. The resizable-way design combined with the HBWR and mixed-structure can achieve 13.2 % performance improvement and 30.4% energy saving over the baseline with uniform structure and non-resizablility.

8.2 FUTURE WORK

The thermal stability of MTJ is also sensitive to the process variation, so that the STT-RAM cell retention time could across a certain range. The refresh of the STT-RAM array shall cover the worst-case retention time. If so, the refresh period could be much smaller than we expected, resulting in a high refresh overhead. Although the proposed dynamic refresh can largely reduce the refresh overhead. There are still a lot of design space if the variation of the retention is taken into consideration. Such variation shall follow certain distribution spatially. And the distribution is predictable based on some post-silicon testing methods. Therefore, the refresh can be designed to have different refresh periods locally. Such local refresh scheme is orthogonal to the proposed dynamic refresh scheme. Once extra timer is needed to record the refresh time of each region.

Regarding the dual-mode architecture of the write performance optimized STT-RAM, the system support is not fully discussed in this dissertation. When and how to switch modes between data accuracy and system performance can be a very complicated topic. It involves hardware/software co-design, operating system, compiler, human interface and so

on. The read disturbance issue of STT-RAM can be solved from other aspect. We can use voltage level detector to monitor the bit-line voltage of the cells under read. The voltage level detector can trigger the rewrite once it detects the voltage change during read operation.

Even this dissertation has done a very comprehensive study of racetrack memory from various design layers. There are still a lot of design issues existing in the racetrack memory design. Since the racetrack memory device physics modeling is ideally utilized. Many assumptions are made. For example, the domain wall movement is assumed to be at uniform speed. In addition, the coupling effect of the domain cells within a racetrack memory is not considered, such effect could result in unwanted flipping of adjacent cells. Therefore, the variation and reliability of the racetrack memory can be further investigated in the future work. Moreover, more solutions at circuit and architecture design level can be invented accordingly. Actually, more architecture level design explorations are still not fully tried. For example, the racetrack access policy on the resizable cache design can be very flexible. Instead of evicting the LRU way, we can also swap the read hit with the LRU way within the enabled portions. More design attempt can be explored to further optimize the racetrack memory.

BIBLIOGRAPHY

[1] S. Kirolos and Y. Massoud. **Adaptive SRAM design for dynamic voltage scaling VLSI systems**. *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pages 25–30, 2011.

[2] P. Nair, S. Eratne, and E. John. **A quasi-power-gated low-leakage stable SRAM cell**. *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 761–764, 2010.

[3] C.H. Kim, J.J. Kim, S. Mukhopadhyay, and K. Roy. **A forward body-biased low-leakage SRAM cache: device, circuit and architecture considerations**. *IEEE Transations on Very Large Scale Integration (VLSI) System*, **13**(3):349–357, 2005.

[4] J. Barth, D. Plass, E. Nelson, C. Hwang, G. Fredeman, M. Sperling, A. Mathews, W. Reohr, K. Nair, and N. Cao. **A 45nm SOI embedded DRAM macro for POWER7TM 32MB on-chip L3 cache**. *2010 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 342–343, 2010.

[5] G. De Sandre, L. Bettini, A. Pirola, L. Marmonier, M. Pasotti, M. Borghi, P. Mattavelli, P. Zuliani, L. Scotti, and G. Mastracchio. **A 90nm 4Mb embedded phase-change memory with 1.2 V 12ns read access time and 1MB/s write throughput**. *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 268–269, 2010.

[6] **http://www.mram-info.com/history**.

[7] X. Wang, W. Zhu, H. Xi, and D. Dimitrov. **Spin Torque Random Access Memory Down to 22 nm Technology**. *Applied Physics Letters*, **93**(10):102508–102508, 2008.

[8] X. Wang, Y. Chen, H. Li, D. Dimitrov, and H. Liu. **Spin Torque Random Access Memory down to 22nm Technology**. *IEEE Transactions on Magnetics*, **44**(11):2479–2482, 2008.

[9] **Everspin Throws First ST-MRAM Chips Down, Launches Commercial Spin-Torque Memory Era**. *http://www.engadget.com/2012/11/14/everspin-throws-first-st-mram-chips-down/*, 2012.

[10] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. **A novel architecture of the 3D stacked MRAM L2 cache for CMPs**. *IEEE Symposium on High-Performance Computer Architecture (HPCA)*, pages 239–249, 2009.

[11] CJ Lin, SH Kang, YJ Wang, K Lee, X Zhu, WC Chen, X Li, WN Hsu, YC Kao, and MT Liu. **45nm Low Power CMOS Logic Compatible Embedded STT MRAM Utilizing a Reverse-Connection 1T/1MTJ Cell**. *Intl Electron Devices Meeting*, pages 1–4, 2009.

[12] Wei Xu, Hongbin Sun, Xiaobin Wang, Yiran Chen, and Tong Zhang. **Design of Last-Level On-Chip Cache Using Spin-Torque Transfer RAM (STT RAM)**. *IEEE Transations on Very Large Scale Integration (VLSI) System*, pages 483–493, 2011.

[13] S. Parkin. **Racetrack Memory: A Storage Class Memory Based on Current Controlled Magnetic Domain Wall Motion**. *Device Research Conf.*, pages 3–6, 2009.

[14] T Koyama, D Chiba, K Ueda, K Kondou, H Tanigawa, S Fukami, T Suzuki, N Ohshima, N Ishiwata, and Y Nakatani. **Observation of the Intrinsic Pinning of a Magnetic Domain Wall in a Ferromagnetic Nanowire**. *Nat. Mat.*, **10**(3):194–197, 2011.

[15] X. Jiang, L. Thomas, R. Moriya, and S.S.P. Parkin. **Discrete Domain Wall Positioning due to Pinning in Current Driven Motion along Nanowires**. *Nano Lett.*, **11**(1):96–100, 2010.

[16] L. Thomas, R. Moriya, C. Rettner, and S.S.P. Parkin. **Dynamics of magnetic domain walls under their own inertia**. *Science*, **330**(6012):1810–1813, 2010.

[17] S Fukami, T Suzuki, K Nagahara, N Ohshima, Y Ozaki, S Saito, R Nebashi, N Sakimura, H Honjo, and K Mori. **Low-Current Perpendicular Domain Wall Motion Cell for Scalable High-Speed MRAM**. *Symp. on VLSI Technology*, pages 230–231, 2009.

[18] AJ Annunziata, MC Gaidis, L Thomas, CW Chien, CC Hung, P Chevalier, EJ O'Sullivan, JP Hummel, EA Joseph, and Y Zhu. **Racetrack Memory Cell Array with Integrated Magnetic Tunnel Junction Readout**. *Intl Electron Devices Meeting*, pages 24–3, 2011.

[19] R. Nebashi and et al. **Nebashi, R and Sakimura, N and Tsuji, Y and Fukami, S and Honjo, H and Saito, S and Miura, S and Ishiwata, N and Kinoshita, K and Hanyu, T**. *VLSIC-25*, pages 300–301, 2011.

[20] C.W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M.R. Stan. **Relaxing Non-Volatility for Fast and Energy-Efficient STT-RAM Caches**. *IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 2011.

[21] Yaojun Zhang, Xiaobin Wang, and Yiran Chen. **STT-RAM Cell Design Optimization for Persistent and Non-Persistent Error Rate Reduction: A Statistical Design View**. *Proc. of ICCAD*, pages 471–477, 2011.

[22] A. Agarwal, D. Blaauw, V. Zolotov, S. Sundareswaran, M. Zhao, K. Gala, and R. Panda. **Path-based statistical timing analysis considering inter-and intra-die correlations**. *Proc. TAU*, pages 16–21, 2002.

[23] C. Kim, D. Burger, and S.W. Keckler. **An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches**. *Proc. of ASPLOS*, page 222, 2002.

[24] Y. Chen, H. Li, X. Wang, W. Zhu, W. Xu, and T. Zhang. **A nondestructive self-reference scheme for spin-transfer torque random access memory (STT-RAM)**. *Proc. of the DATE*, pages 148–153, 2010.

[25] Mircea R Stan and Wayne P Burleson. **Bus-invert coding for low-power I/O**. *IEEE Transactions on Very Large Scale Integration Systems*, **3**(1):49–58, 1995.

[26] Sangyeun Cho and Hyunjin Lee. **Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance**. In *Microarchitecture*, pages 347–357, 2009.

[27] A. Nigam, C.W. Smullen, V. Mohan, E. Chen, S. Gurumurthi, and M.R. Stan. **Delivering on the promise of universal memory for spin-transfer torque RAM (STT-RAM)**. *Symp. on ISLPED*, pages 121–126, 2011.

[28] C. Augustine, A. Raychowdhury, D. Somasekhar, J. Tschanz, V. De, and K. Roy. **Design Space Exploration of Typical STT MTJ Stacks in Memory Arrays in the Presence of Variability and Disturbances**. *IEEE Trans. on Electron Devices,*, **58**(12):4333–4343, 2011.

[29] **The International Technology Roadmap for Semiconductors**. http://www.itrs.net, 2008.

[30] Y Zhang, WS Zhao, D Ravelosona, J-O Klein, JV Kim, and C Chappert. **Perpendicular-Magnetic-Anisotropy CoFeB Racetrack Memory**. *Journal of Applied Physics (JAP)*, **111**(9):093925–093925, 2012.

[31] C Burrowes, AP Mihai, D Ravelosona, J-V Kim, C Chappert, L Vila, A Marty, Y Samson, F Garcia-Sanchez, and LD Buda-Prejbeanu. **Non-adiabatic Spin-Torques in Narrow Magnetic Domain Walls**. *Nat. Phys.*, **6**(1):17–21, 2009.

[32] G. Linley. **Sandy Bridge spans generations**. *Microprocessor Report*, **9**(27):10–01, 2010.

[33] Malcolm Ware, Karthick Rajamani, Michael Floyd, Bishop Brock, Juan C Rubio, Freeman Rawson, and John B Carter. **Architecting for Power Management: The IBM® POWER7 Approach**. *HPCA-16*, pages 1–11, 2010.

[34] G. Mark. **Unifying Primary Cache, Scratch, and Register File Memories in a Throughput Processor**. *MICRO-45*, pages 96–106, 2012.

[35] Koji Nii, Yasumasa Tsukamoto, Tomoaki Yoshizawa, Susumu Imaoka, Yoshinobu Yamagami, Toshikazu Suzuki, Akinori Shibayama, Hiroshi Makino, and Shuhei Iwade. **A 90-nm Low-Power 32-KB Embedded SRAM with Gate Leakage Suppression Circuit for Mobile Applications**. *IEEE Journal of Solid-State Circuits (JSSC)*, **39**(4):684–693, 2004.

[36] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L.C. Wang, and Y. Huai. **Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory**. *Journal of Physics: Condensed Matter*, **19**:165209, 2007.

[37] A. Raychowdhury, D. Somasekhar, T. Karnik, and V. De. **Design Space and Scalability Exploration of 1T-1STT MTJ Memory Arrays in the Presence of Variability and Disturbances**. *IEEE International Electron Devices Meeting*, pages 1–4, Dec. 2009.

[38] J. Z. Sun. **Spin-current interaction with a monodomain magnetic body: A model study**. *Physics Review B*, **62**:570–578, 2000.

[39] Z Sun, W Wu, and H Li. **Cross-Layer Racetrack Memory Design for Ultra High Density and Low Power Consumption**. *DAC-50*, page to appear, 2013.

[40] Rangharajan Venkatesan, Vivek Kozhikkottu, Charles Augustine, Arijit Raychowdhury, Kaushik Roy, and Anand Raghunathan. **TapeCache: a High Density, Energy Efficient Cache based on Domain Wall Memory**. *Proc. of ISLPED*, pages 185–190, 2012.

[41] X. Dong, X. Wu, G. Sun, Y. Xie, H. Li, and Y. Chen. **Circuit and Microarchitecture Evaluation of 3D Stacking Magnetic RAM (MRAM) as a Universal Memory Replacement**. *ACM/IEEE Desgin Automation Conference (DAC)*, pages 554–559, 2008.

[42] R. Desikan, C.R. Lefurgy, S.W. Keckler, and D. Burger. **On-chip MRAM as a High-Bandwidth Low-Latency Replacement for DRAM Physical Memories**. http://www.cs.utexas.edu/ftp/ pub/techreports/tr02-47.pdf, 2008.

[43] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. **Energy reduction for STT-RAM using early write termination**. *IEEE/ACM International Conferernce on Computer-Aided Design (ICCAD)*, pages 264–268, 2009.

[44] X. Wu, J. Li, L. Zhang, E. Speight, and Y. Xie. **Power and performance of read-write aware hybrid caches with non-volatile memories**. *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 737–742, 2009.

[45] J. Li, C.J. Xue, and Y. Xu. **STT-RAM based energy-efficiency hybrid cache for CMPs**. *19th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, pages 31–36, 2011.

[46] Y.T. Chen, J. Cong, H. Huang, B. Liu, C. Liu, M. Potkonjak, and G. Reinman. **Dynamically reconfigurable hybrid cache: An energy-efficient last-level cache design**. *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 45–50, 2012.

[47] Q. Li, J. Li, L. Shi, C.J. Xue, and Y. He. **MAC: migration-aware compilation for STT-RAM based hybrid cache in embedded systems**. *Proceedings of the ACM/IEEE international symposium on Low power electronics and design*, pages 351–356, 2012.

[48] J. Li, L. Shi, C.J. Xue, C. Yang, and Y. Xu. **Exploiting set-level write non-uniformity for energy-efficient NVM-based hybrid cache**. *9th IEEE Symposium on Embedded Systems for Real-Time Multimedia (ESTIMedia)*, pages 19–28, 2011.

[49] H. Zhao, A. Lyle, Y. Zhang, PK Amiri, G. Rowlands, Z. Zeng, J. Katine, H. Jiang, K. Galatsis, and KL Wang. **Low writing energy and sub nanosecond spin torque transfer switching of in-plane magnetic tunnel junction for spin torque transfer RAM**. *Journal of Applied Physics*, **109**:07C720, 2011.

[50] Haakon Dybdahl and Per Stenstrom. **An adaptive shared/private nuca cache partitioning scheme for chip multiprocessors**. *HPCA*, pages 2–12, 2007.

[51] Alessandro Bardine, Manuel Comparetti, Pierfrancesco Foglia, Giacomo Gabrielli, Cosimo Antonio Prete, and Per Stenström. **Leveraging data promotion for low power d-nuca caches**. *Digital System Design Architectures, Methods and Tools*, pages 307–316, 2008.

[52] Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. **Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0**. *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 3–14, 2007.

[53] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. **Reactive NUCA: near-optimal block placement and replication in**

distributed caches. *ACM SIGARCH Computer Architecture News*, **37**(3):184–195, 2009.

[54] J Jaehyuk Huh, C Changkyu Kim, Hazim Shafi, L Lixin Zhang, Doug Burger, and Stephen W Keckler. **A NUCA substrate for flexible CMP cache sharing**. *IEEE Transactions on Parallel and Distributed Systems*, **18**(8):1028–1040, 2007.

[55] Alessandro Bardine, Manuel Comparetti, Pierfrancesco Foglia, Giacomo Gabrielli, and Cosimo Prete. **Way adaptable D-NUCA caches**. *International Journal of High Performance Systems Architecture*, **2**(3):215–228, 2010.

[56] Alessandro Bardine, Manuel Comparetti, Pierfrancesco Foglia, Giacomo Gabrielli, and Cosimo Antonio Prete. **A power-efficient migration mechanism for D-NUCA caches**. *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 598–601, 2009.

[57] Madhu Mutyam and Vijaykrishnan Narayanan. **Working with process variation aware caches**. *Design, Automation & Test in Europe Conference & Exhibition, 2007.*, pages 1–6, 2007.

[58] Mohammed Abid Hussain and Madhu Mutyam. **Block remap with turnoff: a variation-tolerant cache design technique**. *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pages 783–788, 2008.

[59] Serkan Ozdemir, Arindam Mallik, Ja Chun Ku, Gokhan Memik, and Yehea Ismail. **Variable latency caches for nanoscale processor**. *Supercomputing*, pages 1–10, 2007.

[60] Seokin Hong and Soontae Kim. **AVICA: An access-time variation insensitive L1 cache architecture**. *Design, Automation & Test in Europe Conference & Exhibition*, pages 65–70, 2013.

[61] Aarul Jain, Aviral Shrivastava, and Chaitali Chakrabarti. **La-lru: A latency-aware replacement policy for variation tolerant caches**. *VLSI Design*, pages 298–303, 2011.

[62] Bo Zhao, Yu Du, Youtao Zhang, and Jun Yang. **Variation-tolerant non-uniform 3D cache management in die stacked multicore processor**. *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 222–231, 2009.

[63] Wangyuan Zhang and Tao Li. **Characterizing and mitigating the impact of process variations on phase change based memory systems**. *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 2–13, 2009.

[64] Yi Zhou, Chao Zhang, Guangyu Sun, Kun Wang, and Yu Zhang. **Asymmetric-access aware optimization for STT-RAM caches with process variations**. *Proceedings of the 23rd ACM international conference on Great lakes symposium on VLSI*, pages 143–148, 2013.

[65] Zhigang Hu, Stefanos Kaxiras, and Margaret Martonosi. **Let Caches Decay: Reducing Leakage Energy Via Exploitation of Cache Generational Behavior**. *ACM Transaction on Computer Systems (TOCS)*, **20**(2):161–190, 2002.

[66] Stefanos Kaxiras, Zhigang Hu, and Margaret Martonosi. **Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power**. *ISCA-28*, pages 240–251, 2001.

[67] Sparsh Mittal, Zhao Zhang, and Yanan Cao. **CASHIER: A Cache Energy Saving Technique for QoS Systems**. *VLSID-26*, pages 43–48.

[68] Karthik T Sundararajan, Vasileios Porpodas, Timothy M Jones, Nigel P Topham, and Franke. **Cooperative Partitioning: Energy-Efficient Cache Partitioning for High-Performance CMPs**. *HPCA-18*, pages 1–12, 2012.

[69] D. Albonesi. **Selective Cache Ways: On-Demand Cache Resource Allocation**. *MICRO-32*, pages 248–259, 1999.

[70] S Yang, Michael D Powell, Babak Falsafi, Kaushik Roy, and TN Vijaykumar. **An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-Caches**. *HPCA-7*, pages 147–157, 2001.

[71] Se-Hyun Yang, Michael D Powell, Babak Falsafi, and TN Vijaykumar. **Exploiting Choice in Resizable Cache Design to Optimize Deep-Submicron Processor Energy-Delay**. *HPCA-8*, pages 151–161, 2002.

[72] X. Wang, W. Zhu, H. Xi, and D. Dimitrov. **Relationship between symmetry and scaling of spin torque thermal switching barrier**. *IEEE Transactions on Magnetics*, **44**:2479–2482, 2008.

[73] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu. **New paradigm of predictive MOSFET and interconnect modeling for early circuit design**. *IEEE Custom Integrated Circuit Conference*, pages 201–204, 2000. http://www-device.eecs.berkeley.edu/ptm.

[74] CACTI. **http://www.hpl.hp.com/research/cacti/**.

[75] X. Wang and Y. Chen. **Spintronic memristor devices and application**. *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 667–672, 2010.

[76] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. **Statistical design and optimization of SRAM cell for yield enhancement**. *IEEE/ACM International Conferernce on Computer-Aided Design (ICCAD)*, pages 10–13, 2004.

[77] M. Hu, H. Li, Y. Chen, X. Wang, and R.E. Pino. **Geometry variations analysis of TiO 2 thin-film and spintronic memristors**. *DATE*, 2007.

[78] Marss86. **http://www.marss86.org/**.

[79] NVsim. **http://www.rioshering.com/nvsimwiki/index.php**.

[80] R. Heald and P. Wang. **Variability in sub-100nm SRAM designs**. *Proc. of ICCAD*, pages 347–352, 2004.

[81] **Wind River Simics**. *http://www.windriver.com/products/simics/*.

[82] Parsec. **http://parsec.cs.princeton.edu/index.htm**.

[83] Chikako Yoshida, Takao Ochiai, and Toshihiro Sugii. **Correlation between microstructure and electromagnetic properties in magnetic tunnel junctions with naturally oxidized MgO barrier**. *J. of Applied Physics*, **111**:07C716, 2012.

[84] Takayuki Seki, Akio Fukushima, Hitoshi Kubota, Kay Yakushiji, Shinji Yuasa, and Koji Ando. **Switching-probability distribution of spin-torque switching in MgO-based magnetic tunnel junctions**. *J. of Applied Physics*, **99**:112504, 2011.

[85] 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**. *Proc. of Micro*, pages 329–338, 2011.

[86] A Driskill-Smith, D Apalkov, V Nikitin, X Tang, S Watts, D Lottis, K Moon, A Khvalkovskiy, R Kawakami, and X Luo. **Latest advances and roadmap for in-plane and perpendicular STT-RAM**. *IEEE InternationalMemory Workshop (IMW)*, pages 1–3, 2011.

[87] RH Koch, JA Katine, and JZ Sun. **Time-resolved reversal of spin-transfer switching in a nanomagnet**. *Physical review letters*, **92**(8):88302, 2004.

[88] Y. Joo, D. Niu, X. Dong, G. Sun, N. Chang, and Y. Xie. **Energy-and endurance-aware design of phase change memory caches**. *Proc. of the DATE*, pages 136–141, 2010.

[89] A. Brataas, A.D. Kent, and H. Ohno. **Current-Induced Torques in Magnetic Materials**. *Nat. Mat.*, **11**(5):372–381, 2012.

[90] SPEC2006. *http://www.spec.org/cpu2006/*.

[91] MacSim. *http://code.google.com/p/macsim/*.