博士論文

Cache Design Optimization for Energy-Efficient Processors (高電力効率プロセッサのための キャッシュの設計最適化)

Eishi Arima
有間　英志

高電力効率プロセッサのための
キャッシュの設計最適化

和文概要

近年，高性能計算機システムから組み込みシステムに至るまで，プロセッサの電力効率の改善が強く求められている．プロセッサの中でも特にキャッシュはプロセッサの性能，電力消費を大きく左右する構成要素であり，特に携帯端末においてはその設計の最適化は極めて重要となっている．

キャッシュの設計最適化を通してプロセッサの電力効率を改善するにあたって，以下の 2 通りの方針が有力視されている．一方は既存の SRAM キャッシュに対して電源制御を最適化することで，性能を維持しつつ消費電力を削減するというものであり，他方は STT-MRAM に代表されるデータ保持に電力供給を必要としない不揮発性メモリ素子をキャッシュに利用するというものである．

キャッシュの電力効率を最大化するには，システムに与えられる負荷，システムの性能，コスト制約，各メモリ素子の特性等様々な条件を考慮して，上記の中から適切な方針を適用する必要があり，従って最適解はこれら全ての要因に大きく依存する．この様な背景の下，キャッシュの電力効率を様々なシステムに対して最大化するため，本論文では以下の 3 種類の提案を行った．

まず，*Lost Data Prefetch* と呼ばれる，SRAM キャッシュ向け電源制御手法を提案した．現行のプロセッサの SRAM キャッシュにおいては，電源遮断によってリーク電力を大幅に削減できる一方で，それによって生じるデータ損失によって追加のメモリアクセスが発生し，性能が大幅に下がるという問題があった．この問題に対処するため，本論文では損失したデータの内，必要なものをアクセスされる前にメインメモリからプリフェッチする手法を提案している．本論文では，提案手法によってこの性能低下を大幅に抑えることができるということを詳細な評価によって示している．

次に，STT-MRAM キャッシュ向け電力削減手法を提案した．STT-MRAM キャッシュ

においては，書き込みアクセスのために大電流を駆動する必要があるため，その周辺回路に幅広でリーク電力の大きいトランジスタが必要となり，結果として周辺回路のリーク電力が大きくなるという問題があった．この問題に対処するため，本研究では *Immediate Sleep* と呼ばれる電源制御手法を提案している．具体的には，リーク電力の大きいローカルな周辺回路を含むサブアレイごとの電源制御を行う状況を想定しており，あるサブアレイに対する次のアクセスがクリティカルでないと予測されれば，そのサブアレイを直ちに電源遮断するというものである．そうすることで，性能を維持しつつ周辺回路のリーク電力の大幅な削減が可能となる．

さらに，上記提案手法を勘案した上で，性能，コスト制約の下，エネルギーを最小化するようにキャッシュの設計最適化を行うためのフレームワークの提案及び評価を行った．STT-MRAM においては，素子の性能と製造コストは密接な関係にあるため，上記最適化を行う上でその要求性能を明らかにすることは極めて重要である．また，この要求性能はシステムに与えられる負荷状況にも大きく依存する．そこで，本章では負荷状況に応じて要求性能を明らかにするための評価フレームワークを導出し，さらにそれを用いた定量的な評価を行っている．

以上の成果により，様々な制約条件等を考慮しつつキャッシュに対して最適なメモリ素子を選択し，それに対して適切な電源制御を適用することで，プロセッサの電力効率を大幅に改善できることが分かった．本貢献は様々な計算機システムの電力効率を将来に渡って大きく左右するキャッシュの設計を最適化する上で極めて重要である．

Abstract

It is critical to improve the energy-efficiency of microprocessors for many computing systems from HPC (High Performance Computing) systems to embedded devices. In a microprocessor, caches are major contributors to performance and also major power consumers, especially in mobile devices, thus it is critical to optimize cache designs for both performance and power.

To improve the energy-efficiency of microprocessors through the optimization of cache design, the following two approaches are regarded as the most promising. One is to optimize the power control to conventional SRAM based caches in order to reduce their energy while suppressing performance degradation. The other is to utilize non-volatile memories like STT-MRAM for caches because they do not require any power supply for data retention unlike the conventional SRAM based ones, thus their leakage power is quite small.

To maximize the energy-efficiency of caches, we need to choose the best approach from above in consideration of various factors such as the characteristics of typical workloads utilized on a certain system, the performance/cost constraint of the system and the parameters of each memory, thus the best choice highly depends on all of them. To maximize the energy-efficiency of caches for various systems like mobile systems this thesis proposes following three proposals.

First, this thesis proposes an energy-efficiency improvement technique called *Lost Data Prefetch* for SRAM caches during executing I/O bound workloads. In a conventional micro-

processor, turning off an SRAM cache can save the leakage power significantly but introduces data loss which causes significant performance degradation because of additional main memory accesses. To overcome this problem, the proposed technique fetches the necessary lost data from the main memory before they are accessed. The evaluation result shows that our methodology can save large part of the total leakage energy of an SRAM cache while suppressing performance degradation.

Second, we proposed an energy reduction technique for STT-MRAM caches. In an STT-MRAM cache, large and leaky transistors are required for its peripheral circuits to drive large current to its memory cells, thus the peripheral circuits consume large leakage power. To cope with this problem, this thesis proposes a novel power management scheme called *Immediate Sleep* which immediately turns off a subarray that contains leaky local peripheral circuits when the next access will be non-performance-critical. By doing so, the leakage energy can be significantly reduced with minimal performance impact.

Third, we clarified the parameter requirement of STT-MRAM caches for various systems depending on the typical load state. For systems on which only I/O bound applications are usually executed, the cache access performance/energy are less critical to the overall system performance/energy because caches are less frequently accessed. As the cost of STT-MRAM caches highly depends on their parameters, we have to know the parameter requirement to optimize the design. Therefore, this thesis evaluated the parameter requirement of STT-MRAM in consideration of various factors like the average load state of a system, the performance constraint of the system and the required energy reduction.

From above results, it is turned out that the energy-efficiency of microprocessors can be significantly improved by choosing the optimal memory technology for caches and applying the proposed techniques to them when taking various factors into consideration. This contribution is quite important to optimize the cache design which determines the energy-efficiency of various computing systems through the future.

Acknowledgments

Through my five years journey as a graduate student, many people contributed to this dissertation. Here, I would like to express my sincere gratitude to all of them.

First and foremost, I would like to thank my advisor, Prof. Hiroshi Nakamura for his technical and educational supports. If it had not been for his insightful advices and great encouragement, I could not have produced any meaningful results and not have finished this work. He also improved my skills like questioning, analyzing, writing and presenting through discussions, all of which are indispensable for a researcher in my opinion.

I am quite grateful to the other committee members of this dissertation, Prof. Shinji Hara, Prof. Masatoshi Ishikawa, Prof. Hidetsugu Irie, Prof. Takahiro Shinagawa and Prof. Masaaki Kondo for taking their precious time, reading this dissertation, attending my presentation and giving me quite thoughtful comments to complete this work.

I would like to thank all of the past and present members of Nakamura Kondo laboratory for their technical support as well as their friendliness in real-life. Prof. Masaaki Kondo gave me a lot of insightful and meaningful hints about my work at both global and local meetings. Prof. Shinobu Miwa advised and encouraged me many times while he was in this laboratory for four years. Especially, every time I wrote a paper, he gave me a lot of fundamental and critical comments on that, which made me better thinker and writer. I am very grateful to Prof. Takashi Nakada for constructing and managing our simulation infrastructures. In addition to that, he gave me some advices on this work. Special thanks to Dr. Hiroshi Sasaki for his

intelligent and sensible comments on my work and strong encouragement. I really thank Dr. Toshiya Komoda for many meaningful discussions from technical issues to general topics. I thank Dr. Yuan He from the bottom of my heart for technical suggestions, encouragement, proofreading of this thesis and sharing some hobbies. I thank also the other members of the laboratory for their friendship as well as technical contributions.

I am very grateful to many researchers at other affiliations: Dr. Shinobu Fujita, Dr. Hiroki Noguchi, Dr. Kumiko Nomura, Mr. Susumu Takeda and so on. Especially, Dr. Hiroki Noguchi greatly helped me with finishing this work from the viewpoint of devices and circuits. Because he is such a man of ideas, I really enjoyed discussing with him.

Finally, I can never thank my family: my father, my mother and my brother enough for their assistance through my life.

Eishi Arima

March 2016, Tokyo, Japan

Contents

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Chapter 1

Introduction

1.1 The problem: performance/power impact of caches

It is quite important to improve the energy-efficiency, which is given by performance per power consumption, for microprocessors in various computing systems such as HPC (High Performance Computing) systems, data center servers, personal computers and embedded devices [10, 11, 12, 13]. Particularly, it is critical for mobile devices like smart phones, tablets and mobile PCs because of an increasing demand for higher performance and longer battery life. In a general-purpose microprocessor utilized in mobile devices, caches are major contributors to performance and also major leakage power consumers. Therefore, we need to focus on optimizing caches of microprocessors in mobile devices for both system performance and power.

First of all, caches have been utilized to hide the performance gap between processors and main memories thus the design decision strongly affects the system performance. As shown in Figure 1.1, the gap has been expanded for past 30 years. In the figure, the X-axis indicates year, while the Y-axis shows relative performance of processors and memories which is normalized to that in 1980. Because of the significant performance gap between them, caches are implemented in microprocessors and the design decision for them is quite critical for system performance.

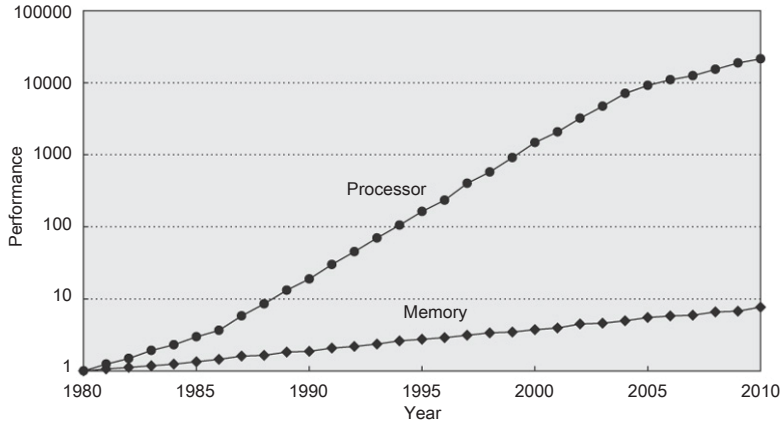


Figure 1.1: The performance gap between processors and memories (extracted from Figure 5.2 in J. L. Hennessy *et al.*(2006) [1])

Figure 1.2: The area of the LLC in a commercial mobile processor and the power breakdown of the processor (remade from Figure 4 in V. George *et al.*(2007) [2] and Figure 1 in S. Li *et al.*(2011) [3] respectively)

Second, conventional caches, particularly an LLC (Last Level Cache), occupy large area in a microprocessor chip as shown in the left figure of Figure 1.2, thus its leakage power consumption is also large as shown in the right figure. In the right figure, the Y-axis shows the power breakdown including dynamic/leakage power of all the components of the microprocessor. The dynamic power is consumed by a microprocessor component when it operates, while the leakage is always consumed if it is powered on. In the figure, *PG* stands for Power-Gating with which a component is turned off and the leakage power is reduced significantly. As shown in the figure, the leakage power of the LLC accounts for large part of the total power consumption of the microprocessor if PG is not applied (N/PG), thus its reduction is

very important. On the other hand, PG can greatly save the leakage power of an LLC as shown in the figure. However, applying PG to caches causes data loss thus it brings significant performance degradation [14, 15]. Therefore, the techniques which can reduce the leakage power of caches while suppressing performance degradation are eagerly required in modern microprocessors.

Moreover, these trends will continue through the future, thus improving energy-efficiency of the caches is quite important. One reason for this is the leakage power of a chip increases as VLSI technology scales [16]. Because a chip can operate lower supply voltage at the same frequency as VLSI technology scales and the dynamic power is proportional to the square of the voltage, thus the voltage to a chip has been decreasing for past few decades. However, the leakage power of a chip increases exponentially as the voltage becomes smaller and smaller (particularly, it will be dominant when the voltage is smaller than around 0.5[V] [17]). Although some device technologies like Fin-FET [18] are proposed to suppress the leakage power, they cannot change the fundamental characteristic of leakage. Because of this reason, it is predicted that the leakage power of a mobile device will continue to grow for at least the next decade [19]. Moreover, as the required size of working memory and number of cores continuously ramp up for mobile devices, the capacity of caches also need to be increased, thus their performance and leakage impacts will be more and more critical in the future. Thus, improving energy-efficiency of caches is quite important.

To improve the energy-efficiency of microprocessors through optimizing caches, following two approaches can be regarded as the most promising. One is to apply a sophisticated power control methodology to caches which can reduce their power significantly while suppressing performance degradation. The other is to implement caches with non-volatile memories which do not require any power supply for data retention unlike the conventional SRAM based ones thus their leakage power is quite small. Especially, STT-MRAM based caches have been regarded as the most promising because of their outstanding properties

such as high write endurance, fast access performance, higher density than SRAM based ones [20, 21, 22, 23, 24, 25].

To maximize the energy-efficiency, we need to choose the best approach from above in consideration of various factors such as the characteristics of typical workloads utilized on a certain system, the performance/cost constraint of the system and the parameters of each memory, thus the best choice highly depends on all of them. For example, in terms of performance, the best design highly depends on the characteristics of the executed workloads such as working set size and data reuseness. This is because an SRAM cache enables faster accesses but has smaller capacity than an STT-MRAM cache having the same area. In terms of energy, an STT-MRAM cache generally consume small energy compared to an SRAM cache because of its non-volatility. On the other hand, the manufacturing cost of an STT-MRAM cache is much higher than that of an SRAM cache because an STT-MRAM memory cell contains non-silicone element called MTJ.

1.2 Contribution

To overcome this problem, this thesis proposes following three proposals to optimize energy-efficiency of caches in consideration of various workloads and system performance/cost constraints.

First, this thesis proposes an energy reduction technique for SRAM caches during executing I/O bound workloads. In a conventional microprocessor, its components are turned off in stages during idle state. However, turning off an SRAM cache causes data loss, which causes significant performance degradation because of additional main memory accesses introduced by the data loss. Therefore, caches, especially last level caches, are rarely turned off for some systems even though they consume large leakage energy. To overcome this problem, this thesis proposes a data management technique called *Lost Data Prefetch*, by which the necessary lost data are fetched from the main memory before they are accessed to mitigate the

performance degradation. The evaluation result shows that our methodology can save large part of the total leakage energy of an SRAM cache during executing I/O bound workloads while suppressing performance degradation.

Second, we proposed an energy reduction technique for STT-MRAM caches. In an STT-MRAM cache, large and leaky transistors are required for its peripheral circuits to drive large current to its memory cells. Therefore, the peripheral circuits consumes large leakage power while the memory cells consumes near-zero leakage due to the non-volatility. To cope with this problem, this thesis applies subarray-level power-gating to the peripheral circuits. Moreover, this thesis proposes power management scheme called *Immediate Sleep*, by which a subarray is turned off immediately when the next access will be non-performance-critical. By doing so, the leakage energy can be significantly reduced with minimal performance impact. The experimental results shows that those optimization can save most part of the leakage energy consumption of an STT-MRAM cache at negligible performance degradation.

Third, we analyzed the parameter requirement of STT-MRAM caches for various systems depending on the typical load state. For systems on which only I/O bound applications are usually executed though operation time, the cache access performance/energy less affect the overall system performance/energy because caches are less frequently accessed. Because the cost of STT-MRAM caches highly depends on their parameters, we have to know the parameter requirement to optimize the design. Therefore, this thesis evaluates the parameter requirement of STT-MRAM caches based on models in consideration of various factors like the average load state of a system, the performance constraint of the system and the required energy reduction rate. By doing so, we can choose the best memory technology for various systems having various factors.

From above results, it is turned out that the energy-efficiency of microprocessors can be significantly improved by choosing the optimal memory technology for caches and applying the proposed techniques to them when taking various factors into consideration. This contri-

bution is quite important to optimize the cache design which determines the energy-efficiency of various computing systems, especially mobile devices, through the future.

1.3 Organization of this dissertation

The rest of this dissertation is organized as follows. Chapter 2 covers the background information of this dissertation such as caching in conventional microprocessors, power-management scheme utilized in caches and various memory technologies. Chapter 3 describes the overall problem definition. Chapter 4 presents the technique called *Lost Data Prefetch* for mitigating performance degradation caused by turning off caches. Chapter 5 describes the methodology called *Immediate Sleep* which can improve the energy-efficiency of STT-MRAM caches. Chapter 6 clarifies the parameter requirement of STT-MRAM caches. Chapter 7 introduces related work and describe the differences from ours. Chapter 8 concludes and generalizes this work.

Chapter 2

Background

2.1 Cache architectures

To hide the performance impact of a main memory access that takes few hundred cycles in today's computer systems, caches are usually implemented in conventional microprocessors [1]. Moreover, to meet the growing working set size of various applications, the capacity of caches has been increased and they occupy large area in a conventional processor die [26, 2]. Because the capacity and access latency of a cache are generally in the relationship of trade-off, they are usually multi-leveled in a processor chip and the last level cache (LLC) occupies the largest area. Thus the leakage power reduction for LLCs is very important.

Although each cache level has different characteristics, all of them usually have the same structure called set-associative cache as shown in Figure 2.1. As shown in the figure, the address which indicates the physical location of the data on main memory is divided into three parts: tag, index and offset. The index is utilized to identify the location of the data on the cache, while the tag is utilized to confirm whether the data is stored in the cache or not. In caches, data are allocated at the granularity of cache line which consists of some words and the offset is utilized to access the requested word within a cache line. The valid bit (V) indicates whether the cache line has valid data or not. As shown in the figure, a cache is

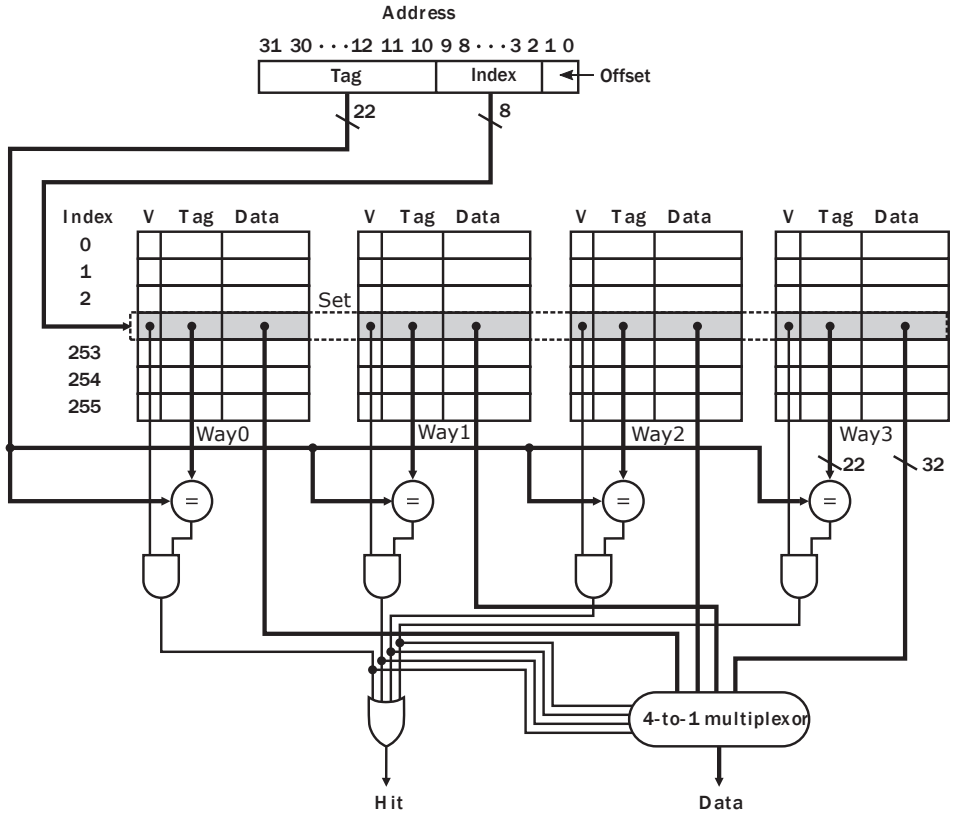


Figure 2.1: Structure of a conventional set-associative cache (remade from Figure 5.17 in D. A. Patterson *et al.*(2008) [4])

divided into several arrays called ways and they are accessed in parallel. A set of cache lines having the same index is called a cache set and all the cache lines belonging to the same set are accessed at the same time. An incoming cache line can be placed on the associated set and if the set is filled up with data, then a victim line is selected from the set based on a certain cache replacement algorithm.

2.2 Power-gating: a circuit technique for reducing leakage power

To reduce the leakage power for various components of microprocessors, a circuit technique called power-gating is widely adopted [5, 27, 28, 29, 30, 31, 32]. With the technique,

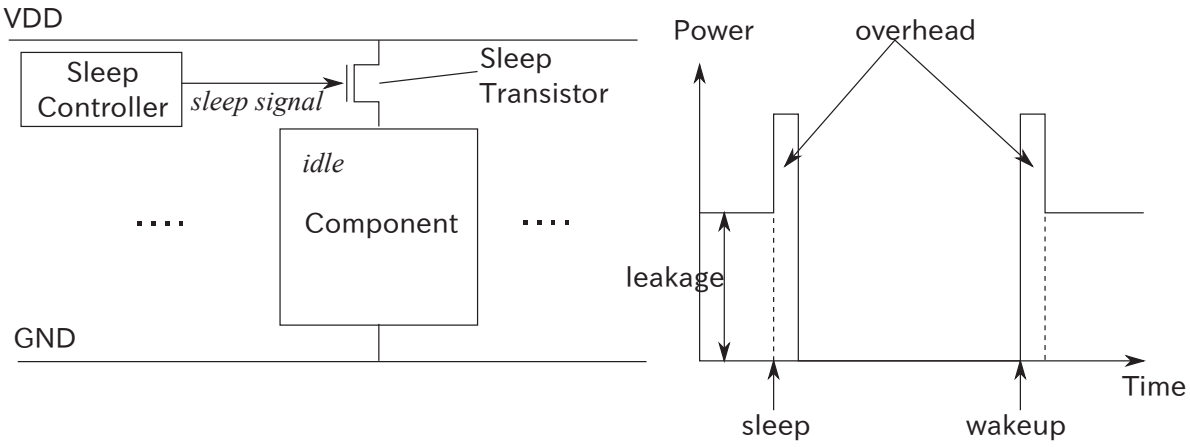


Figure 2.2: Power-gating circuit and state transition

the power supply to a component is cut, thus the component consumes very small leakage power during power-gated state. Figure 2.2 illustrates the technique and an example of state transition. As shown in the figure, a sleep transistor is inserted between a component and VDD (or GND). If the sleep controller shown in the figure finds out an idle component, it sends a sleep signal to the sleep transistor connected to the component, then the component turns into the sleep state, which means it is turned off. Although the technique can save leakage power significantly, it also incurs energy/performance overheads for wakeup operations. Therefore, it is necessary to avoid too frequent state transitions. Moreover, applying power-gating to conventional SRAM-based caches also causes data loss, thus the energy/performance overheads of power-gating can be more critical for them.

2.3 Leakage reduction of caches during system active state

2.3.1 Power management framework utilized in conventional systems

In various modern systems, power management techniques such as DVFS and power-gating are utilized based on a power management standard like ACPI [33]. The framework is supported by various operating systems, which monitors the activities of various components and turns them into one of the sleep states while they are idle. In general, a component

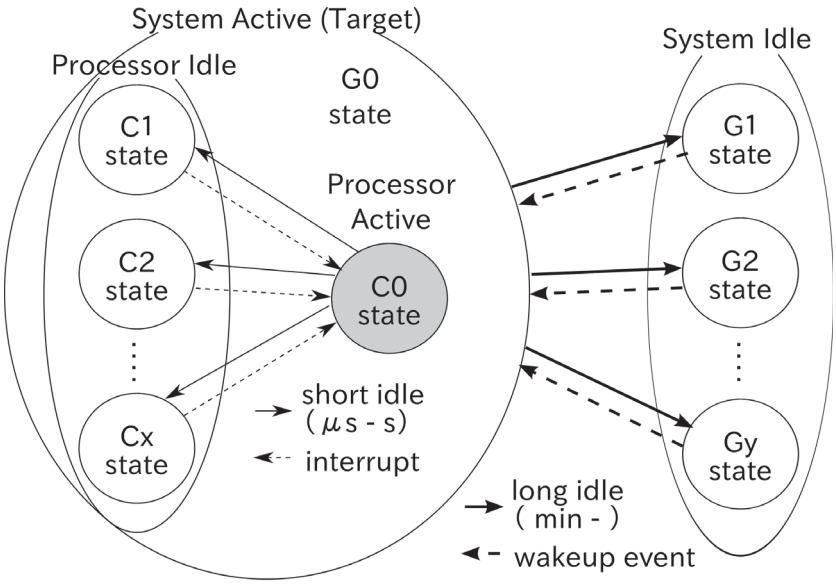


Figure 2.3: The power management framework

consumes lower power when it stays in deeper sleep state having higher transition overheads.

The overview of the state transition utilized in the power management framework is shown in Figure 2.3. A processor turns into a sleep state (chosen from C1 to Cx) when it has no executable process and stays in idle (particularly short idle). The operating system selects an appropriate sleep state with taking various factors like idle length into consideration. If the processor receives an interrupt from I/O devices during idle state, then it returns to the active state and restarts the execution of the associated process. Although the processor stays in idle in these sleep states, the overall system is regarded as active (G0 state). On the other hand, the system turns into sleep state (G1 to Gy), if the processor has not executed any processes for a long period such as a few 10s minutes. In such states, overall system turns into low power mode like hibernation in which all the main memory contents are moved to the storage device and the system is turned off. The system returns to G0 state if it receives a wake up event like pushing the power button.

Figure 2.4 is a schematic figure that shows the power consumption of a processor in various sleep depths. In the figure, the X-axis shows the idle length, while the Y-axis indicates

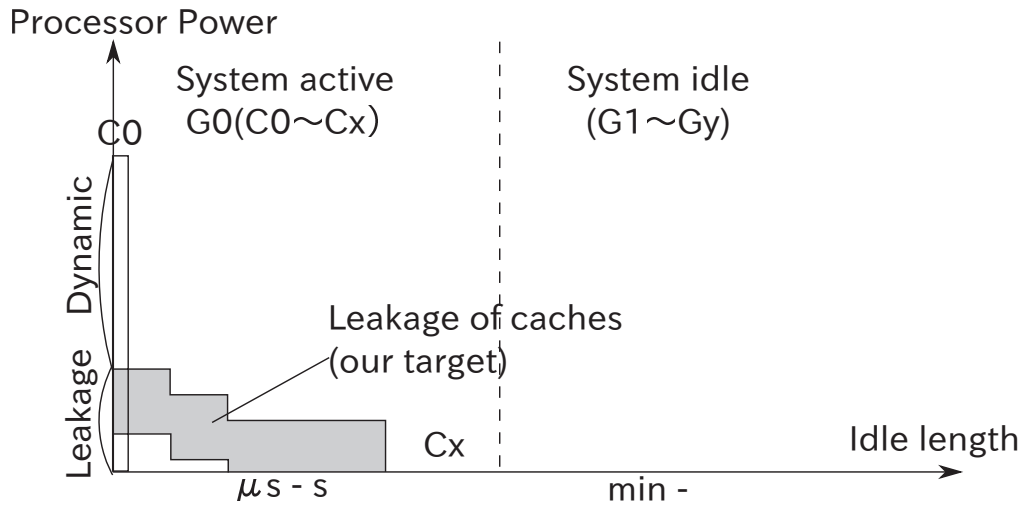


Figure 2.4: Power consumption as a function of idle length

the total power consumption of the processor. As shown in the figure, we do not have to consider the power consumption of a processor during a system idle state because it is completely turned off and consumes near-zero power. On the other hand, we need to consider the leakage power of caches in the system active states. Especially, it is dominant in the total power consumption of the processor in most of the processor idle states. This is because turning off caches causes data loss, which harms performance, thus they are less frequently turned off in the processor components. Thus reducing leakage of caches during the system active state is one of the critical concern in modern systems.

Figure 2.5 shows an example of the sleep states implementation utilized in conventional processors, which is based on the framework described above [5]. A core of a processor chip has states named Cx(x=0to7) and the entire chip can also go into the deeper sleep state called PC7 if all of the cores stay in idle. In C1 state, the clock of the core is stopped thus it can significantly save dynamic energy. In C3 state, the clock generator is shut down and the local caches are flushed. Here, the flush means all of the data stored in the cache are invalidated and the dirty data stored in it are written back to the memory. In C6/C7 states, power-gating is applied to the core. In PC7 state, the LLC is flushed and then the entire chip is turned off.

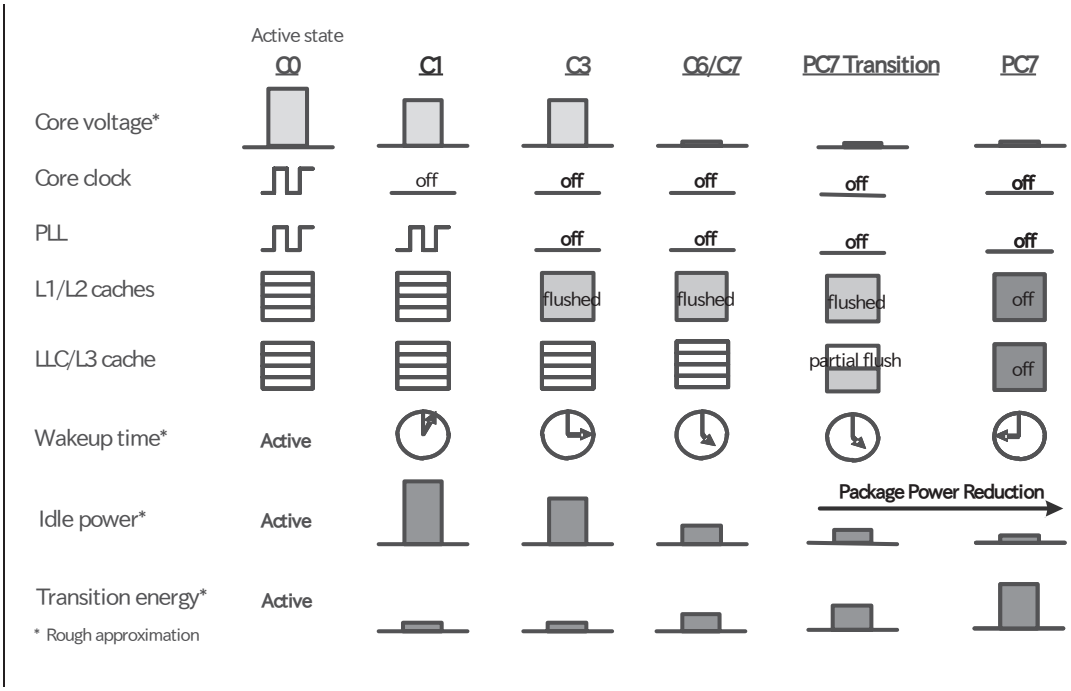


Figure 2.5: Various sleep state in Sandy Bridge microarchitecture (extracted from Figure 4 in E. Rotem *et al.*(2012) [5])

Because turning off LLCs causes significant performance degradation as shown later, thus LLCs are rarely turned off in various modern microprocessors.

An example of the active/idle transition of a microprocessor is shown in Figure 2.6. In the figure, the X-axis shows the time, while the Y-axis indicates each core of the CMP (Chip Multi-Processor). In the active state, a core executes a program. The arrows shown in the figure indicate the interrupts from I/O and the core has to execute the associated program after receiving one. In the package idle state, all of the cores are idle thus the chip can be turned into PC7 state as described above.

2.3.2 Processor idle during system active state

In this section, we introduce the prior work which clarified the idleness of processors in various systems from servers to mobile devices [34, 35, 36, 37, 38, 6]. In their evaluations, it

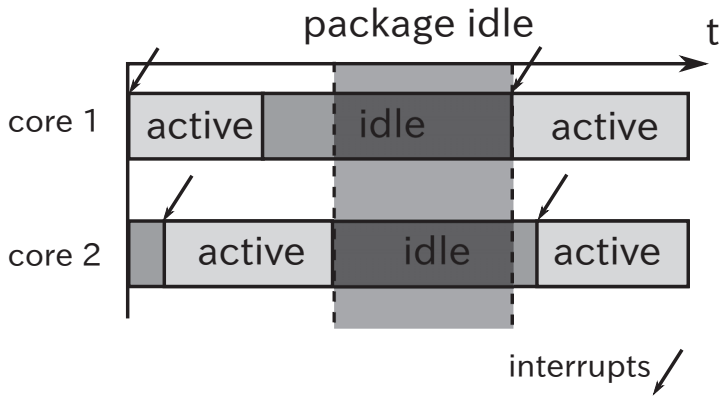


Figure 2.6: State transition of a microprocessor

was turned out that processors usually stay idle during system active state in various systems due to I/O bound applications.

The load state of data center servers are evaluated in [34, 35]. According to the evaluation in [34], all of the CPU cores in a DNS server stays idle for 80% of the total operation time. In [35], the load state of web search servers are disclosed. According to the paper, each web search server stays in idle in 80% of the total operation time.

The CPU utilization in mobile devices is also evaluated in several work [38, 6]. In [38], the CPU usage of mobile PC during executing office softwares is evaluated. According to the evaluation, the processor stays in idle in 70% of the total execution time on average. This is because these applications are I/O bound ones which utilize user interfaces. The CPU usage during executing modern mobile workload is shown in Figure 2.7, which is based on the evaluation in [6]. The X-axis indicates executed workloads, while the Y-axis shows the processor idle rate. As shown in the figure, the mobile processor stays idle most of the time during executing these workloads because these applications cause very frequent network accesses thus it often turns into idle state.

As described above, the CPU usage is usually very low in many systems from servers to mobile devices due to the I/O bound applications. Therefore, we have to care not only CPU or memory bound applications like SPECCPU 2006 [39] but also I/O bound applications to take

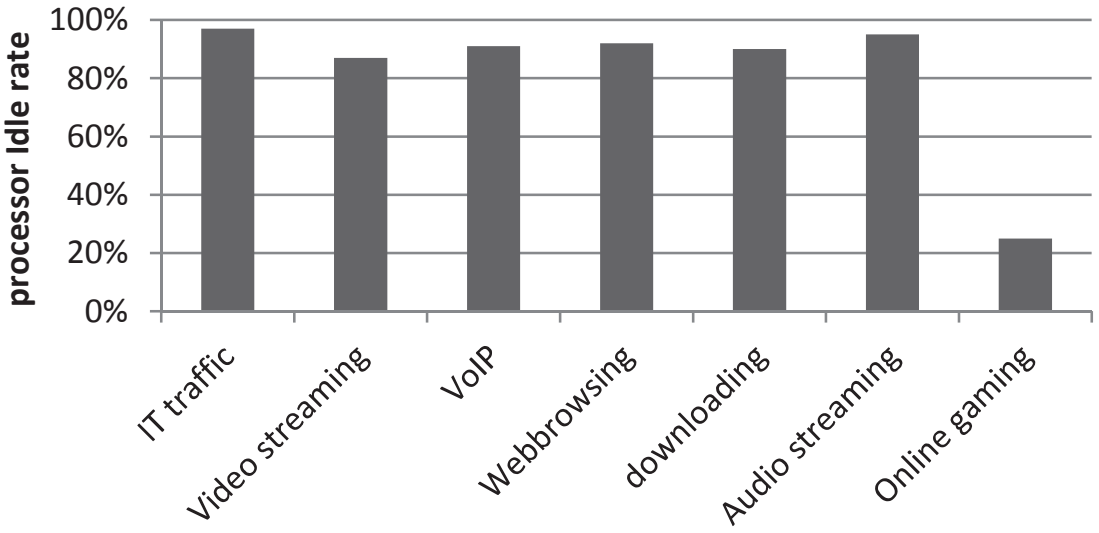


Figure 2.7: Processor idle rate during executing mobile workloads (remade from Table II in R. WANG*et al.*(2011) [6])

the processor idle state into consideration, though only CPU or memory bound applications are usually utilized for cache evaluation. Moreover, even if conventional processors stay idle in the large part of system active time, caches are usually kept awake and waste significant leakage power due to the intolerable performance degradation caused by data loss. Therefore, mitigating the performance impact of turning off caches is required for designing energy-efficient microprocessors.

2.4 Various memory technologies and their characteristics

To mitigate the performance impact of data loss in caches brought by power-gating, implementing caches with non-volatile memories has been regarded as one of the promising approaches recently. Thus, in this section, we introduce various emerging non-volatile memories and describe their characteristics including their merits and demerits.

Figure 2.8 shows the comparison of various memory technologies. The X-axis shows the capacity while the Y-axis indicates the access speed. Fast access operation and high

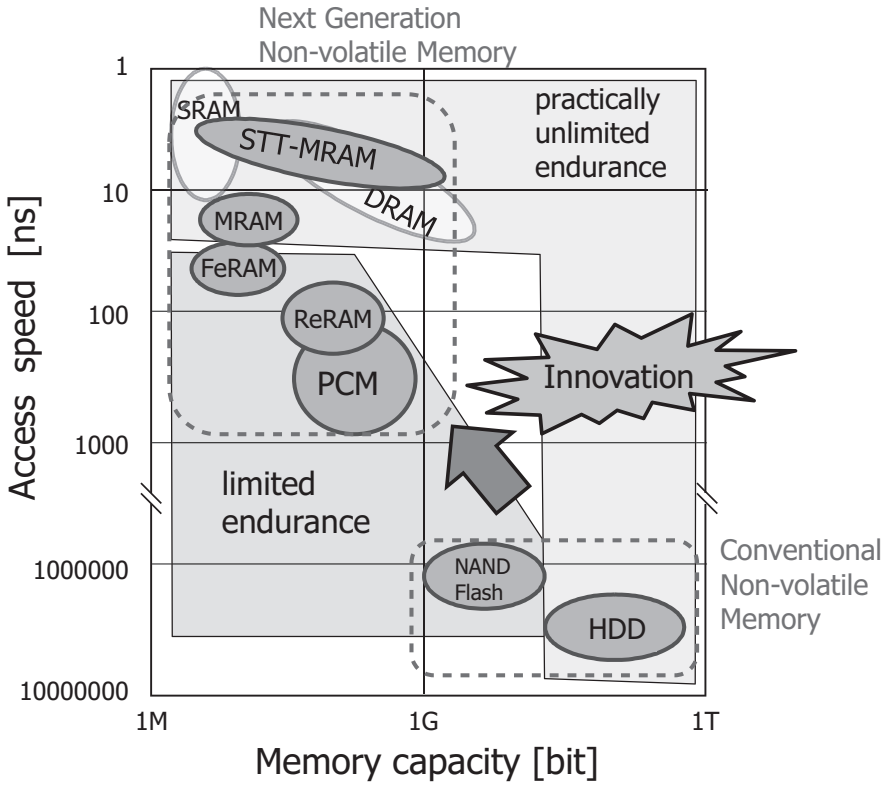


Figure 2.8: Comparison of various memory technologies (extracted from Figure 2 in H. Nakamura *et al.*(2014) [7])

write endurance are required for a cache memory. If we consider such conditions, only STT-MRAM is suitable for cache memories among these emerging non-volatile memories, thus this thesis focuses on only SRAM and STT-MRAM to optimize cache design.

The detailed cell structure of STT-MRAM is shown in Figure 2.9. As shown in the figure, an STT-MRAM memory cell consists of an MTJ cell and a transistor, thus it is much denser than a conventional 6T-SRAM cell. The MTJ cell functions as a variable resistance that keeps one bit. The value of the MTJ is determined by the orientation of the magnetic field in the free layer. The write operation to the cell is conducted by driving large write current, thus the write energy of an STT-MRAM cache is generally large.

The comparison of characteristics between these memory technologies is shown in Table 2.1. The access parameters of STT-MRAM caches used to be much higher than those

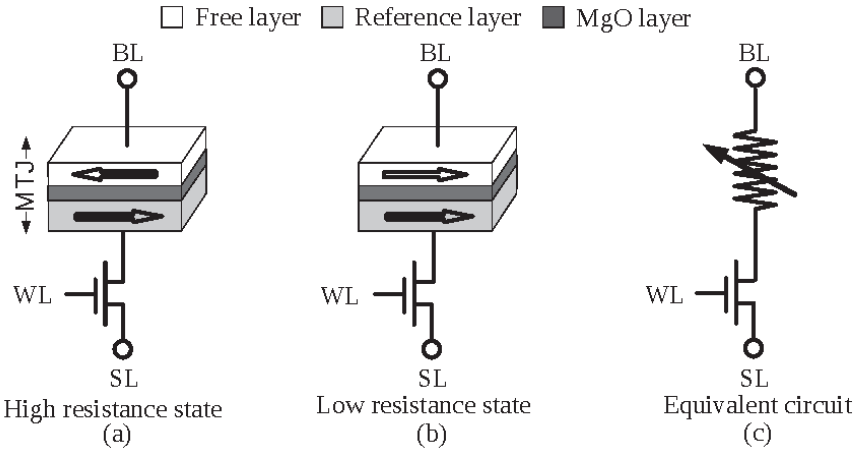


Figure 2.9: Structure of an STT-MRAM memory cell (extracted from Figure 1in Z. Sun *et al.*(2011) [8])

of SRAM caches. However, in a state-of-the-art STT-MRAM cache, these parameters are comparable to those of SRAM caches thanks to the improvement of device technologies as shown in Figure 2.10. In Figure 2.10, the trend of access energy in STT-MRAM for past several years is shown based on various prior researches [40, 41, 42, 9]. The trend of access latency is similar to that shown in the figure because it can also be smaller if the required access energy become smaller in STT-MRAM caches [20]. On the other hand, the manufacturing cost grows larger and larger if we utilize various sophisticated device technologies for STT-MRAM to improve its access parameters. In addition to that, the manufacturing cost of an STT-MRAM cache is higher than that of an SRAM cache in general, because it contains non-transistor elements (MTJ cells). Therefore, we have to chose an appropriate STT-MRAM cache design having the parameters which meet the cost/performance/power requirements of

Table 2.1: Parameter comparison of two different memory technologies (lower is better)

	access latency	access energy	cost	leakage	power-gating overhead	cell size
SRAM	low	low	low	high	high	high
STT-MRAM	low←high	low←high	***high←mid***	***mid***	low	mid

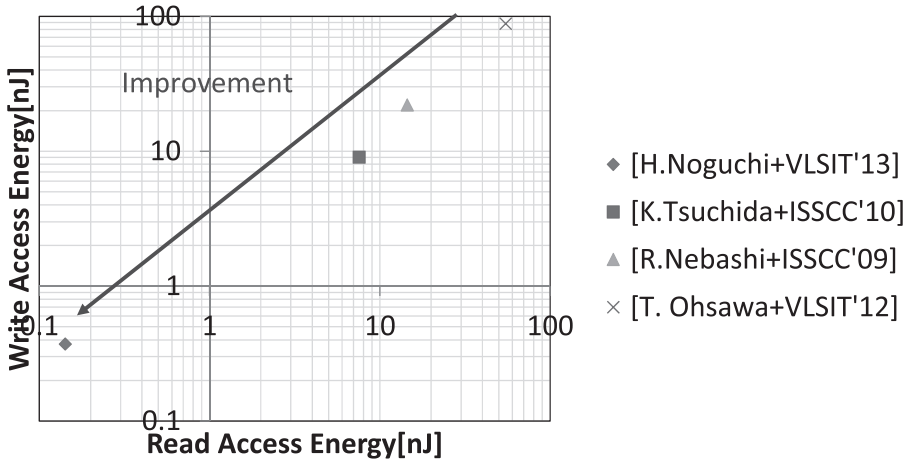


Figure 2.10: Access energy reduction through device technology improvement (remade from TABLE 2 in H. Noguchi *et al.*(2013) [9])

the system.

Furthermore, the leakage of STT-MRAM caches is not quite small compared with that of SRAM caches even though they are non-volatile memory based caches as shown in the table. This is because the leakage power of their peripheral circuits is quite high since large and leaky transistors are required for them as shown later. Therefore, to achieve large leakage reduction with STT-MRAM, power-gating is also required for STT-MRAM based caches. However, the energy/latency overheads of turning off STT-MRAM caches are much smaller than those of powering off SRAM caches because turning off STT-MRAM caches does not cause data loss due to their non-volatility. Therefore, power-gating can be applied to STT-MRAM caches even in the processor active state though that is usually applied only in the deep processor sleep states for conventional SRAM based caches.

Chapter 3

Overview of this work

In this chapter, we present the overall objective of this work in section 3.1 firstly. Then we consider various STT-MRAM designs in section 3.2. Finally, we give a qualitative analysis for the problem in section 3.3.

3.1 Overall problem definition

To improve the energy efficiency of caches, we have to optimize their design in consideration of several factors like performance and manufacturing cost constraints. In this section, we formulate the optimization problem.

The overall problem can be defined as described in the equations (3.1)-(3.3). The first equation expresses the objective function of this work, which minimize the energy consumption of a cache (E) and the second equation shows the constraint condition for the system performance (P), while the third equation indicates the constraint condition of the manufacturing cost of the chip (C). **D** specifies the device utilized for the cache (SRAM or STT-MRAM) and its parameters. **W** shows the workloads utilized on the system. These parameters and the others are also summarized in Table 3.1.

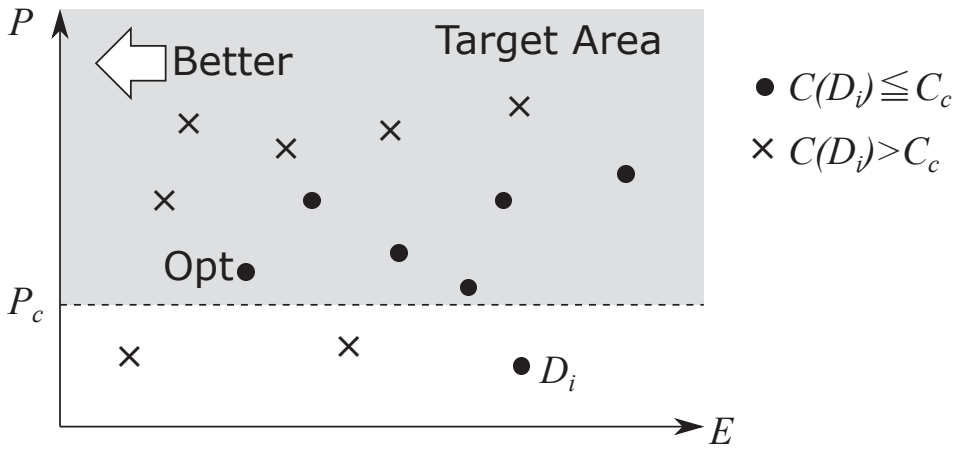


Figure 3.1: Visualization of the optimization problem (**W** is fixed)

$$\text{minimize} \quad E(\mathbf{D}, \mathbf{W}) \tag{3.1}$$

$$\text{subject to} \quad P(\mathbf{D}, \mathbf{W}) \geq P_c \tag{3.2}$$

$$C(\mathbf{D}) \leq C_c \tag{3.3}$$

Name	Remarks
E	Energy consumption of the cache (LLC)
P	Performance of the system
C	Manufacturing cost of the chip
P_c	Performance constraint of the system
C_c	Cost constraint of the chip
D	A vector specifies device utilized in the LLC
W	A vector specifies workloads utilized on the system

Table 3.1: Specification of parameters

The problem is visualized as shown in Figure 3.1. In the figure, the X-axis shows the energy consumption of the cache (E), while the Y-axis indicates the performance (P). Each plot shows energy and performance for various memory parameters D_i. A dot means that the cost of the design subjects to the cost constraint. In the figure, the workloads specification **W**

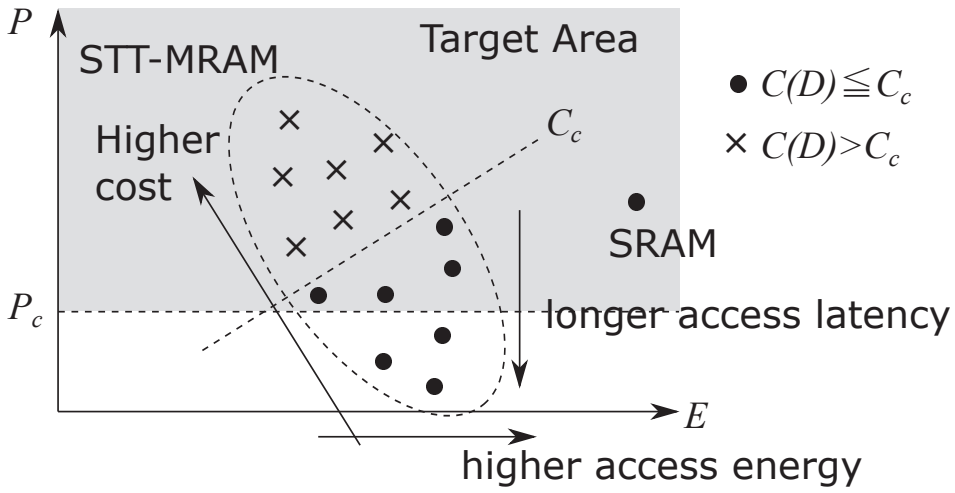


Figure 3.2: Performance/energy of various STT-MRAM designs

is given. As shown in the figure, we have to choose the optimal device to minimize the energy of an LLC in consideration of performance/cost constraints for the given typical workloads on the system.

3.2 Performance/energy of various STT-MRAM designs

Figure 3.2 shows SRAM and various STT-MRAM design on the *E*-*P* plane. In the figure, the workloads **W** are fixed. In this thesis, we consider that the device parameters of the SRAM cache are already optimized or fixed and thus we try to choose the best design for the STT-MRAM cache and then compare it with the SRAM cache. As shown in the figure, various STT-MRAM cache designs show various energy/performance, which depends on the MTJ cell designs like MTJ cell size and materials. In general, the device having higher access energy and/or longer access latency shows lower manufacturing cost [15]. In addition to that the manufacturing cost of STT-MRAM cache is higher than that of SRAM cache because an STT-MRAM memory cell contains a non-silicone MTJ cell.

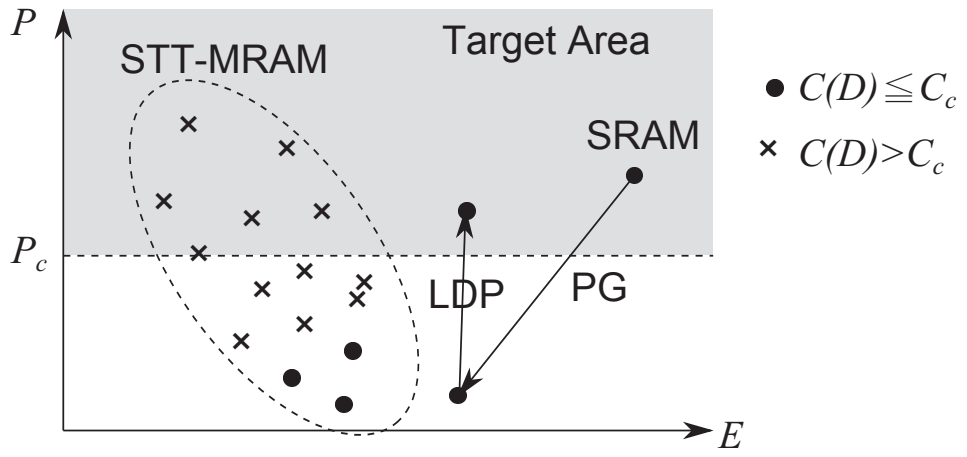


Figure 3.3: Improving energy-efficiency of SRAM caches with LDP

3.3 Qualitative analysis and solution

To optimize the design, we execute following approaches. Firstly, we propose low power techniques for both SRAM and STT-MRAM caches. Then, we optimize **D** for various system workload **W** with considering such techniques.

Figure 3.3 shows the energy-efficiency improvement of an SRAM cache brought by the proposed methodology called *Lost Data Prefetch (LDP)*. As shown in the figure, applying Power-Gating (PG) to SRAM caches causes significant performance degradation due to the additional main memory references caused by data loss brought by PG. Therefore, LLCs are rarely turned off for some systems. In contrast to that, our methodology can mitigate the performance degradation greatly by prefetching the lost data. As shown in the figure, the methodology is effective especially for systems whose cost constraint is relatively strict and most STT-MRAM designs do not subject to the constraint. The detailed information of LDP will be shown in Chapter 4.

Figure 3.4 shows energy reduction of STT-MRAM caches brought by our proposal called *Immediate Sleep(IS)*. In STT-MRAM caches, their peripheral circuits consume large leakage energy because they are implemented with large and leaky transistors to drive large write

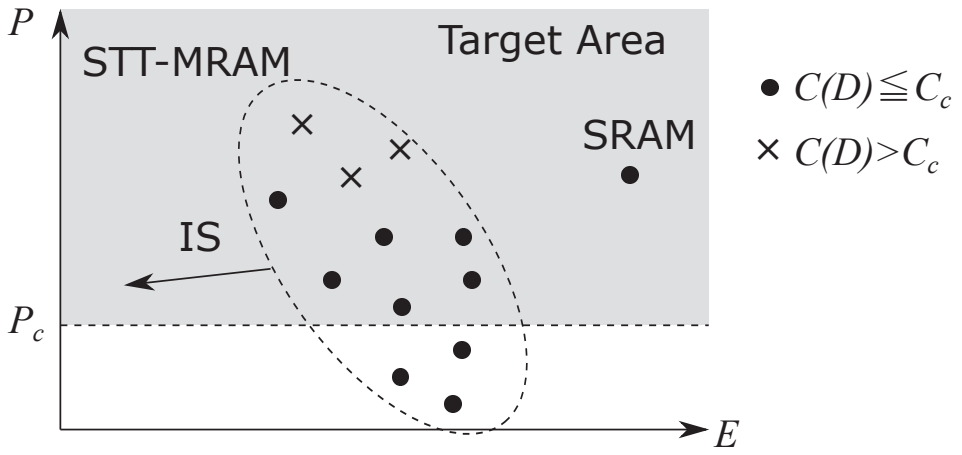


Figure 3.4: Improving energy-efficiency of STT-MRAM caches with IS

current. As shown in the figure, IS can save the leakage energy of the peripheral circuits with negligible performance impact. The detailed information of IS will be shown in Chapter 5.

Finally, we attempt to optimize the cache design. First of all, to solve the problem shown in the equations (3.1)-(3.3), we clarify the energy E and performance P as functions of **D** and **W**. On the other hand, we do not attempt to quantify the manufacturing cost of STT-MRAM because that depends on various factors like the amount of production and the yield of the technology node thus the equation is quite complex. Moreover, the detailed information of the cost of STT-MRAM has not been released from the industry. Alternatively, we attempt to find out the parameter requirement **D** of STT-MRAM which meet the following equations for various (E_c, P_c, **W**). In the first equation, E_c expresses the target energy reduction brought by STT-MRAM compared with the SRAM cache.

$$E(\mathbf{D}, \mathbf{W}) \leq E_c$$

$$P(\mathbf{D}, \mathbf{W}) \geq P_c$$

$$\cancel{C(\mathbf{D}) \leq C_c}$$

After acquiring the parameter requirement D_i for several meaningful $E_c(i)$ from above

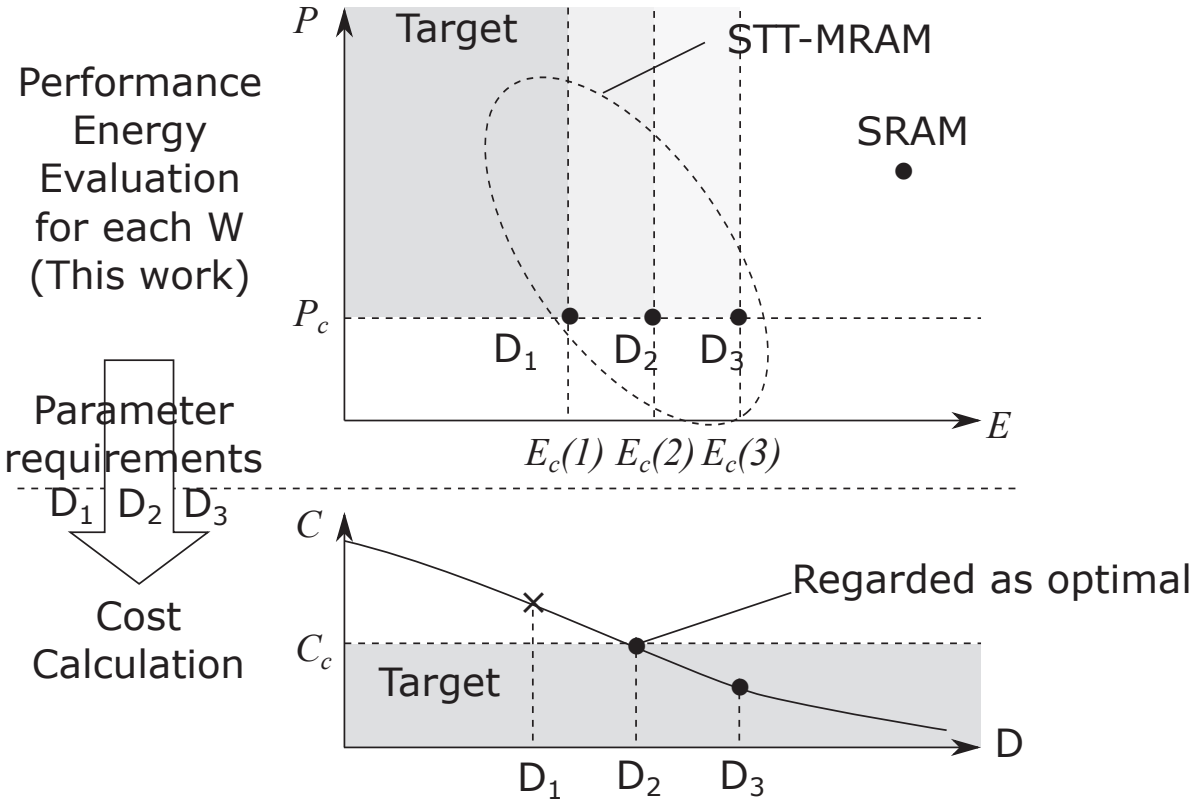


Figure 3.5: The overall optimization process

equations, the industry should only execute the following steps to find out the optimal design if it would finally be able to quantify the manufacturing cost. First, it should calculate the manufacturing costs for all D_i. Second, if there are D_i which meet the cost constraint, then the D_s that has the smallest $E_c(s)$ of them can be regarded as the optimal, otherwise the SRAM based design can be regarded as the optimal. The overall optimization process is illustrated in Figure 3.5. The detailed information of this evaluation will be shown in Chapter 6.

Chapter 4

Lost Data Prefetch

In this chapter, this thesis proposes a novel prefetch methodology called *Lost Data Prefetch* for improving energy-efficiency of SRAM caches. More specifically, the proposed method prefetches the lost data from main memory before they are re-referenced. By doing so, we can mitigate the performance impact of turning off caches, thus can save leakage energy of caches with small performance impact. In section 4.1, this thesis analyzes the performance impact of data loss caused by powering off caches. In section 4.2, we explain the overview, algorithm and implementation of the proposed method. In section 4.3, the evaluation environment and results are shown.

4.1 Performance impact of turning off SRAM caches

As described before, when a processor stays in idle, the operating system tries to change the state of the processor into one of the low-power modes. Although caches are powered off in some low-power states, turning off them can cause significant performance degradation because of data loss.

As a preliminary evaluation, we quantified the performance impact of turning off caches during idle state. In the evaluation, we assumed that only I/O bound applications are executed on the machine and caches are turned off when the processor turns into idle state. The processor has 2-level cache hierarchy. The detailed evaluation environment will be shown in

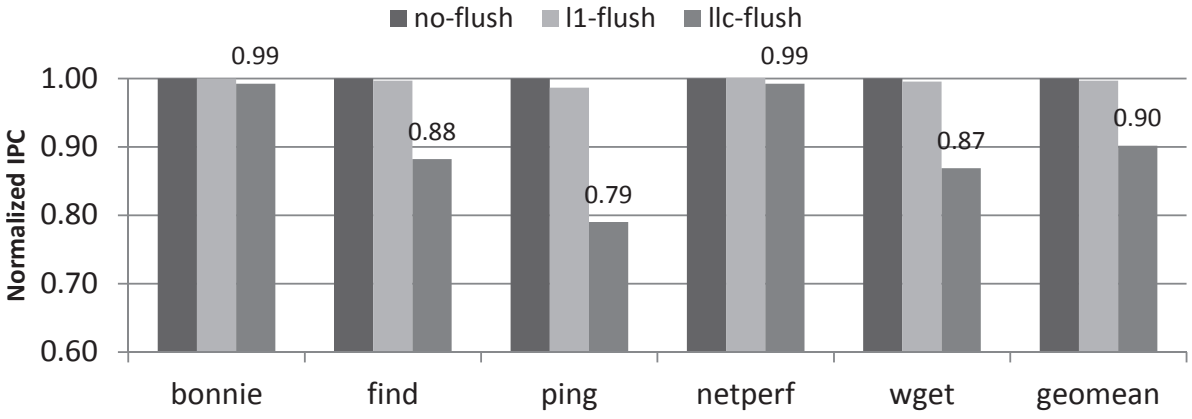


Figure 4.1: Performance impact of cache flushes

a later section of this chapter.

Figure 4.1 shows the performance impact of turning off caches. The X-axis shows the executed workloads, while the Y-axis indicates the relative performance. We compared 3 policies: *no-flush* means both L1cache and LLC are not turned off; *l1-flush* means only L1 caches are turned off during idle state; *llc-flush* means both L1 and LLC are turned off during idle state. The performance is normalized to that of *no-flush*. As shown in the figure, the performance impact of turning off L1 cache is negligible in most cases. On the other hand, flushing LLC causes significant performance degradation. For `wget`, the performance degradation is 21%, which means the total execution time also increases around 20%. Because of this intolerable performance impact, LLCs are usually turned on even in the idle state in most modern systems.

The performance degradation is caused by re-references to lost data because these accesses bring additional cache misses. Therefore, we quantified the reuse rate that is given by the average number of re-referenced cache lines in each active state per the total number of cache lines as shown in Figure 4.2. The evaluation environment will be described later. In the figure, the horizontal axis shows the executed workload, while the vertical axis shows the reuse rate of the LLC.

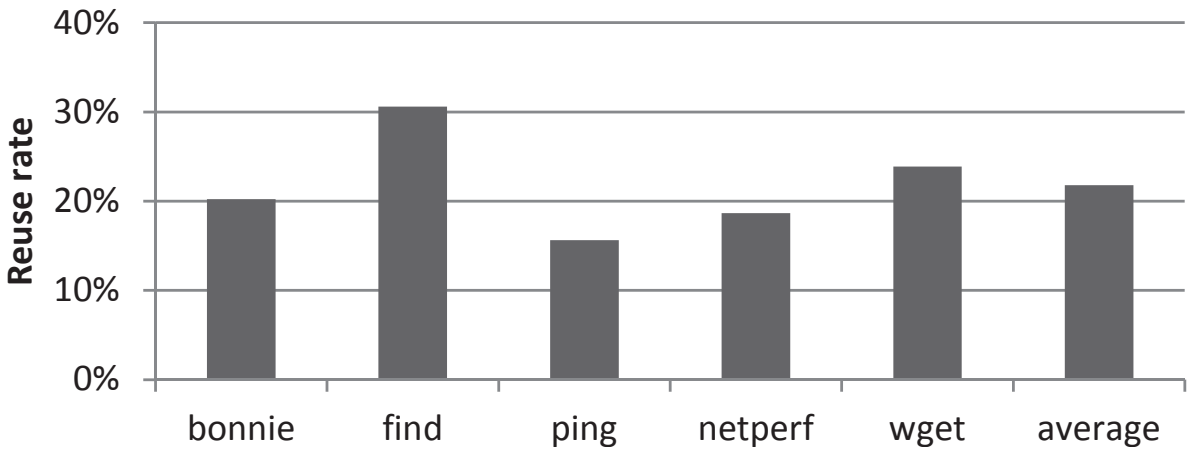


Figure 4.2: Reuse rate in a last level cache

As shown in the graph the reuse rate is only 16% for `ping` and 22% on average, thus small part of data loss of cache lines causes performance degradation. In other words, we can save the performance degradation if we can take only 20% of the total cache lines from the main memory before they are re-referenced.

These results motivate us to propose a novel prefetch scheme called *Lost Data Prefetch* that prefetches the lost data predicted to be re-referenced. In the next section, we will explain the overview, algorithm and implementation of the proposal.

4.2 A methodology for mitigating the performance degradation

In this section, we will explain the detailed information of our proposal called *Lost Data Prefetch*. In section 4.2.1, we will explain the overview of the proposed prefetch. In section 4.2.2, we will describe the algorithm utilized in our prefethcer. In section 4.2.3, we will show the implementation of the prefetcher.

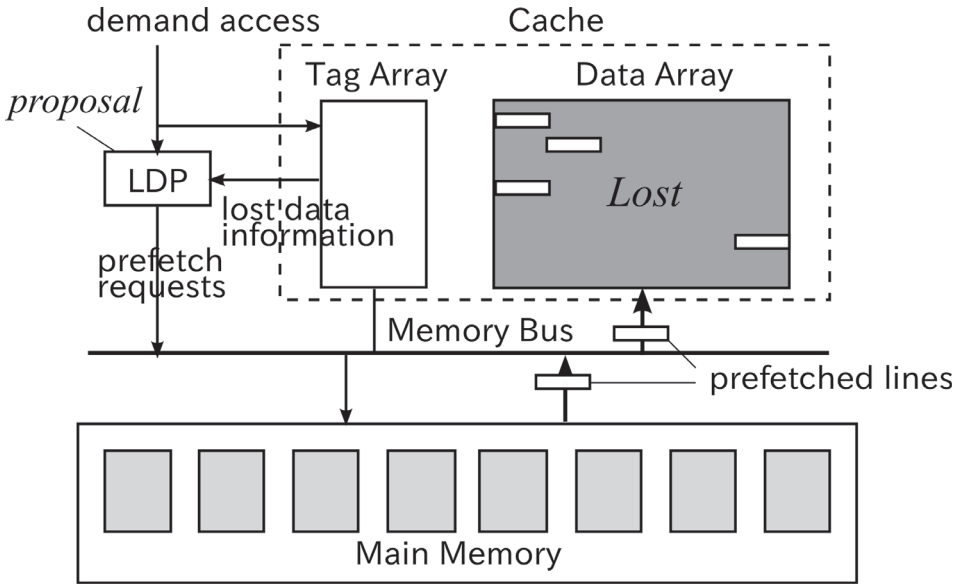


Figure 4.3: Overview of lost data prefetch

4.2.1 Prefetching lost data with tag array

To mitigate the performance degradation caused by the data loss brought by turning off caches, this thesis proposes a novel data management scheme. More specifically, we propose a novel prefetch technique which fetch the lost data from the main memory before they are accessed. To do so, it is necessary to identify the address of the lost data stored in the cache before turned off. Because the information is stored in the tag array, thus it is necessary to keep supplying power to the tag array to retain it. Therefore, we divide the power domain into the tag array and the data array and apply power-gating only to the data array.

Figure 4.3 shows the overview of our prefetch technique utilizing the information of the tag array. Here, we assume the situation where the information stored in the data array is lost, while that of tag array is still retained. We introduce a novel hardware called *Lost Data Prefetcher* (LDP) as shown in the figure. If a cache access request arrives after turning on the cache, the LDP receives the address and returns the addresses of the lost data which will be accessed near future and the data are fetched from the main memory.

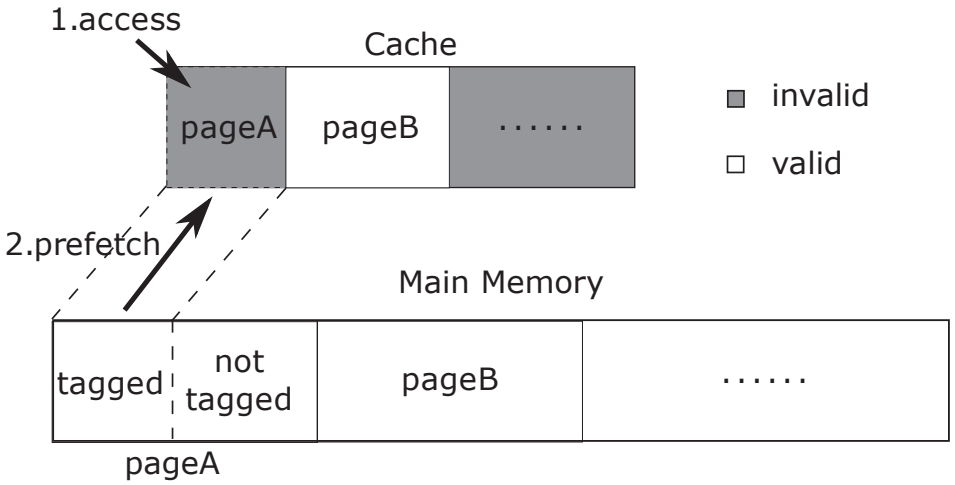


Figure 4.4: Overview of the page level prefetch

Although the tag array need to be kept awake in our method, the energy overhead is negligible because its size is much smaller than that of the data array. In 32-bit memory space, the size of an entry of the tag array is about 20 bits, thus it accounts for only about 4% of the total cache capacity having 64B cache lines. In general, the leakage power consumption of a component is proportional to its area, thus the leakage of the tag array is negligible compared with the total leakage consumption of a cache.

4.2.2 Prefetch algorithm

In this section, we explain the prefetch algorithm utilized in LDP. Fetching all of the lost data from the main memory can significantly reduce the performance degradation. However, it also increases the energy consumption because the energy overhead of a main memory access is not negligible for many systems. Thus, our algorithm tries to fetch only the lost data which are probably reused in the future.

In our prefetch algorithm, if data within a certain amount of memory space is accessed, then the all of the lost data included in the space is prefetched. This is because it is well-known that memory references generally have spatial locality for many applications [43].

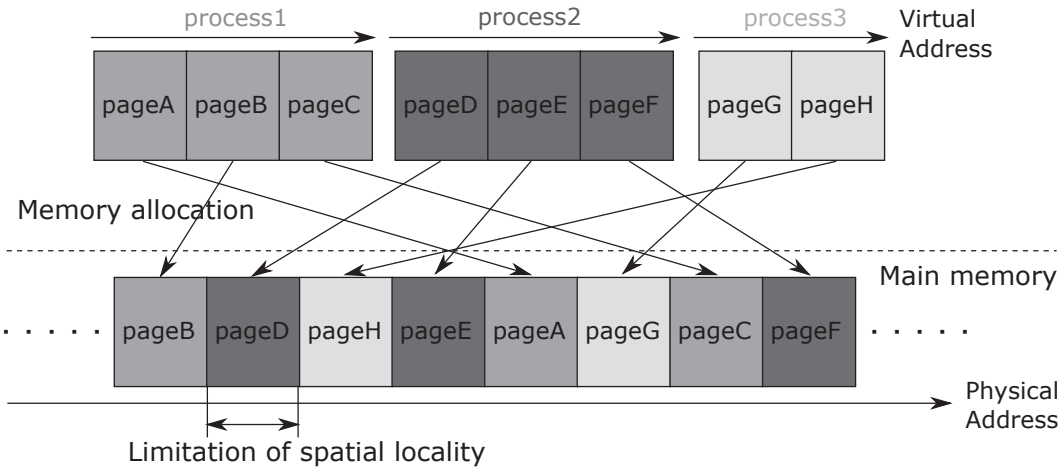


Figure 4.5: An example of memory space allocation by an operating system

More specifically, data elements within relatively close address is accessed intensively within a certain amount of time in many cases.

In this thesis, we apply the prefetch at the granularity of pages as shown in Figure 4.4. More specifically, if a cache line belonging to a page is accessed (1) then the lost data that are tagged in the cache is fetched from the main memory (2) as shown in the figure. In terms of reducing the performance degradation, coarser granularity is better because the first access to a memory space always causes cache miss in our methodology. On the other hand, in terms of energy, finer granularity is better because fetching non-reused data is just a waste of energy. We regard the page granularity as the best because a page is the coarsest segment which can show the spacial locality while the energy waste is negligible according to our evaluation as shown later. Because an operating system allocates memory space to a process at the granularity of pages, two consecutive pages on the physical memory space have generally no relationship with each other as shown in Figure 4.5.

4.2.3 Implementation of the prefetcher

Figure 4.6 shows the implementation of our prefetcher. To conduct our prefetch, two hardware called *Requested Line Address Register*(RLAR) and *Prefethch Queue*(PQ) are re-

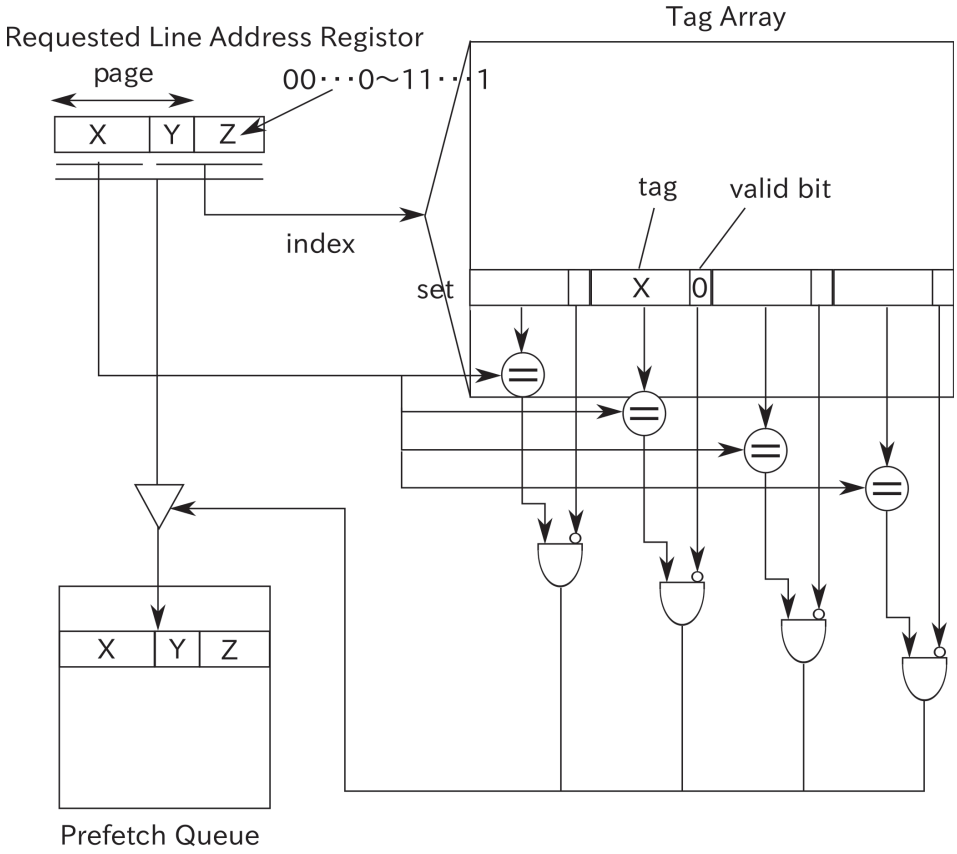


Figure 4.6: Implementation of LDP

quired. The former is utilized for looking up the lost data in the tag array and the latter is utilized for storing the addresses of lost data to prefetch. Because the PQ is usually implemented in conventional processors for various prefetch schemes [44, 45, 46, 47, 48], thus we also utilize the queue for LSD. Therefore, the additional hardware required for our prefetch is only RLAR. In the RLAR, X shows the tag address of the data, while Y and Z indicates the index to the cache. In addition to that X and Y shows the page address to which the data belongs.

In our implementation, a request to the cache is regarded as a trigger for the prefetch operation. Namely, after receiving a request, the address of the request is stored in RLAR, then the prefetcher calculates the page address to which the requested data belongs (XY in

the figure), then look up the lost data included in the page with the address in the tag array and stores the address of the lost data in PQ. More specifically, the values from (Y00...0) to (Y11...1) are utilized as indexes to the tag array and if one of a cache line in the accessed set has the tag address X and the cache line is invalid then the address of the cache line is stored in PQ. By doing so, we can identify the addresses of the lost data which belong to the page and have not been reused yet. The lookup operation is suspended when a demand access arrives or PQ is filled up. Therefore, this operation does not affect cache access performance at all.

The prefetch instructions are issued to the main memory bus when the bus has plentiful bandwidth as other prefetchers do [44, 45, 46, 47, 48]. If the bandwidth is plentiful, an entry is taken from the bottom of PQ and a data access command is issued for the address.

The tag look up operations consume extra energy thus we should avoid useless lookups. For example, if most of the lost data have already been prefetched, then the look up operation is just a waste of energy. To avoid this wastage, we limit the number of pages for which the prefetch operations are conducted as shown in later. Moreover, the lookups for already prefetched pages are also waste of energy. To avoid such lookups, the addresses of already prefetched pages should be registered in a table. Fortunately, such page addresses are usually stored in TLBs and they are accessed before each cache access, thus the additional storage for them is not required.

4.3 Evaluation

In this section, we will explain the evaluation settings and the experimental results.

4.3.1 Evaluation settings

We implemented our methodology on a full system simulator Gem5 [49] and evaluated our proposal with it. The detailed settings are shown in Table 4.1. The processor has 2 cores

and 2-level caches. The proposed methodology is applied to the last level cache because turning off it causes significant energy/performance overheads due to the additional main memory accesses. In our evaluation, the operating system issues halt instruction when the processor turns into the idle state, and when the simulator catches the instruction, both l1 and l2 caches are flushed in turn, which means they are turned off. The benefit of our methodology depends on the main memory bandwidth and the limitation of the number of prefetched pages. Therefore, we evaluate our methodology for different settings having different main memory bandwidth and limitation. The energy parameters utilized in our evaluation are also listed in Table 4.1. These parameters are calculated by CACTI [50].

Name	Remarks
OS	linux 2.6.27, 8KB page
CPU	2core, 1.6GHz, alpha, in-order 2-way fetch/decode/issue
L1 D/I cache	32KB, 2-cycle latency 4-way set assoc., 64B line
L2 cache	512KB, 10-cycle latency 8-way set assoc., 64B line 8-entry write buffers, 1read/write port, 0.373W leakage power, 0.153nJ/line dynamic energy
Main memory	160-cycle latency, 6.4GB/s bandwidth, 51nJ/line dynamic energy
Disk	10ms latency
Ethernet	100μs latency, 100Mb/s bandwidth
LDP	256-entry (PQ), 256 (maximum # of prefetched pages)

Table 4.1: Simulation parameters utilized for the evaluation of LDP

We utilized the following applications for the evaluation.

bonnie `bonnie++` is a widely utilized benchmark software for evaluating disk I/O performance [51]. This software issues frequent disk requests thus the processor also frequently turns into the idle state.

find `find` command is executed at the root directory of the file system. During directory search, disk I/O requests are frequently issued.

ping `ping` command is also utilized for our evaluation. In our simulation, two computers

are connected with an ethernet and one machine sends ping to the other. During waiting for the response from the other machine, the computer stays in the idle state.

`netperf` `netperf` is a tool for checking the status of a network [52]. During execution, the machine frequently goes to the idle state.

`wget` `wget` is a command to fetch contents from web sites. In our evaluation, a client machine tries to fetch a file having few MB from the other http server.

4.3.2 Evaluation results

Performance and energy analysis

Figure 4.7 shows the prefetch hit rate for various workloads. The hit rate is defined as the rate of the reused lines that is successfully prefetched before accessed compared with the total reused lines. Thus, higher prefetch hit rate is better for performance. In the figure, horizontal axis indicates the executed workloads, while the vertical axis shows the prefetch hit rate. As shown in the figure, the prefetch hit rate is around 60-80% thus it should work well for reducing the performance impact.

Figure 4.8 demonstrates the performance improvement brought by our prefetcher. In our evaluation, we compared following 4 methodologies: *no-flush* means caches are not turned off; *l1-flush* means only L1caches are flushed; *llc-flush* means both L1caches and LLC are flushed; *LDP* means both L1caches and LLC are flushed and LPD is applied to LLC. In the figure, the X-axis shows executed workloads while the Y-axis indicates relative performance which is normalized to that for *no-flush*. Because *l1-flush* causes cache misses brought by data losses only in L1 caches, this policy shows the limitation of our methodology. Since LDP shows high prefetch hit rate as shown in Figure 4.7, it can save large part of performance penalty caused by data losses.

Figure 4.9 shows the efficiency of our prefetcher, which is given by the rate of prefetched and reused lines compared with total prefetched lines. The X-axis indicates the executed

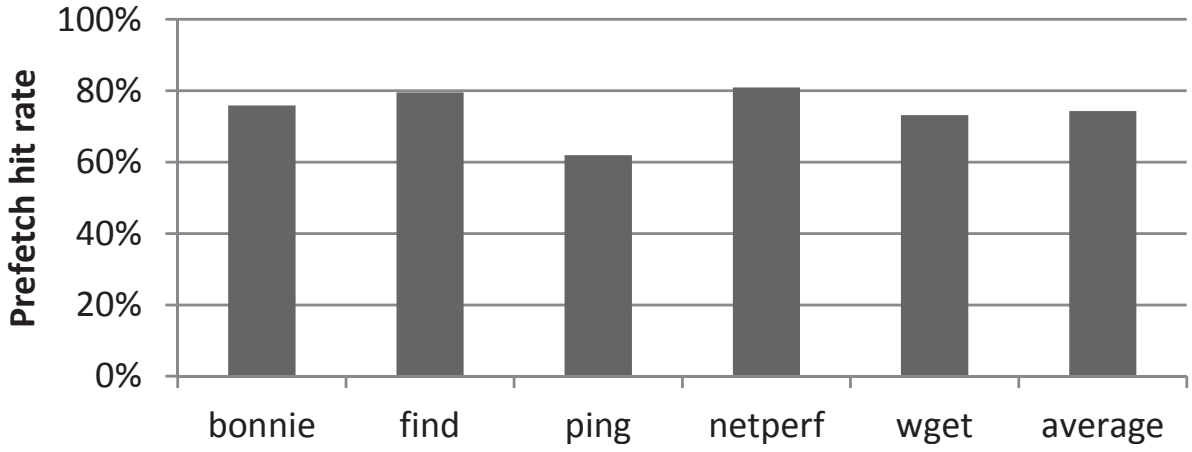


Figure 4.7: Prefetch hit rate

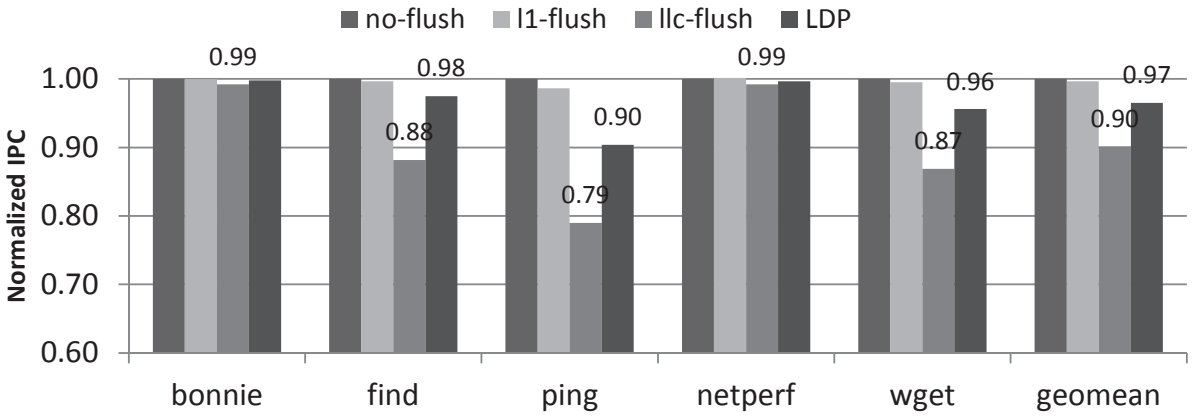


Figure 4.8: Performance comparison among various methodologies

workloads while the Y-axis shows the efficiency. Because fetching non-reused data is just a waste of energy, thus higher efficiency is better for energy reduction. As shown in the figure, the efficiency is about 60% on average and about 70% for `wget` that has relatively high spatial locality. For `ping`, the efficiency is relatively low. However, the energy overhead of prefetching is still small compared with leakage energy reduction even for the workload as shown the next figure.

Figure 4.10 indicates the total energy consumption of the LLC for various methodologies. *llc-static*, *llc-dynamic* and *overhead* shows the leakage energy of the LLC, the dynamic energy consumption of the LLC and the energy overheads of the proposed prefetcher, respec-

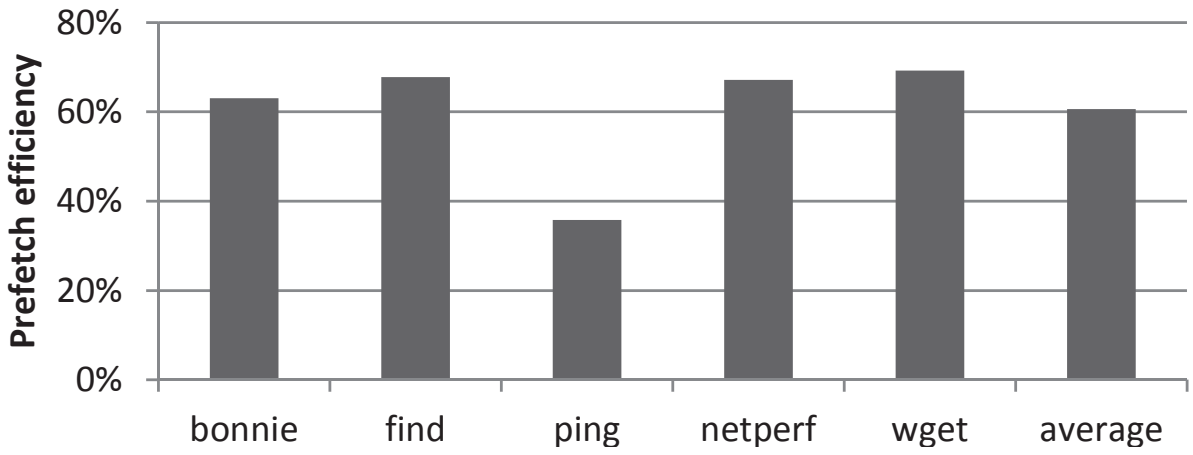


Figure 4.9: Efficiency of LDP

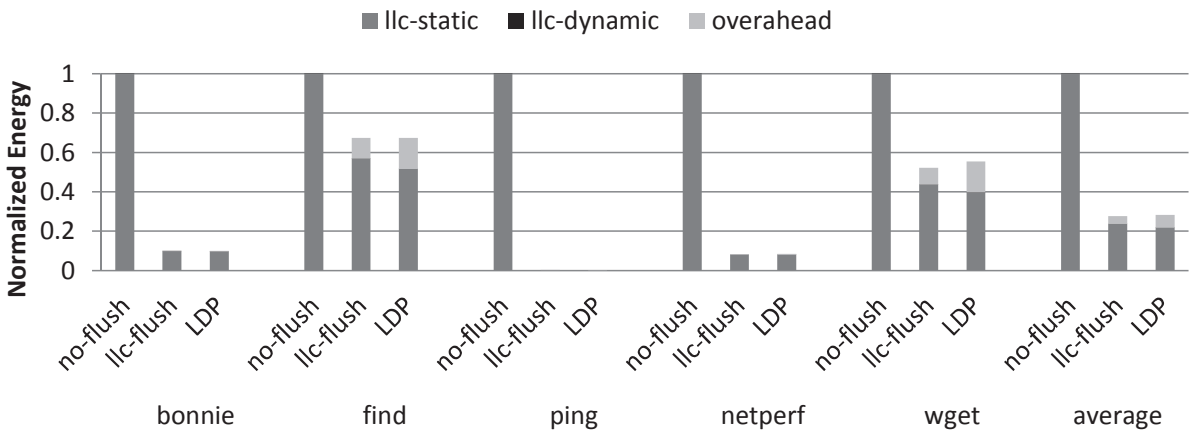


Figure 4.10: Energy comparison among various methodologies

tively. We compared 3 methodologies: *no-flush* means the LLC is not turned off; *llc-flush* shows the LLC is flushed during idle state; *LDP* shows that the LLC is turned off during idles state and our prefetch is applied during active state. The horizontal axis shows various methodologies per executed workloads. The Y-axis shows the relative energy that is normalized to the total LLC energy consumption for *no-flush*. In this evaluation, the energy overhead contains only the extra memory/LLC access energy caused by both prefetching and turning off the cache.

As shown in the figure, turning off the LLC helps reducing the total energy. This is

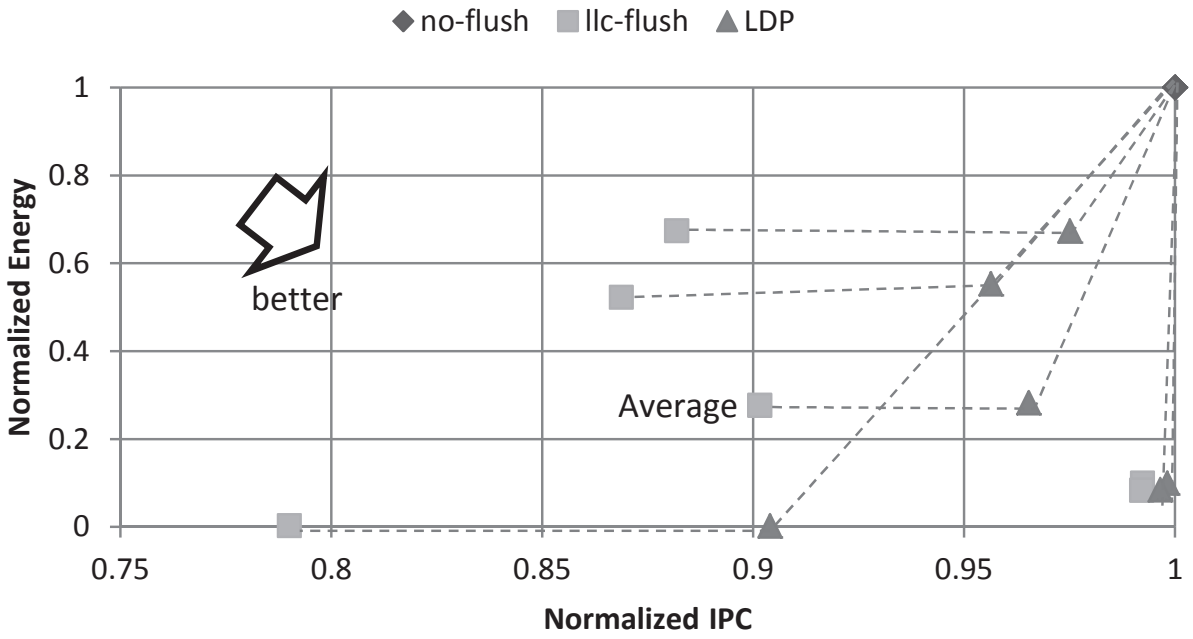


Figure 4.11: Performance and energy comparison among various methods for various applications

because the leakage energy is dominant in the total energy consumption of LLCs. The effectiveness of leakage reduction highly depends on the total idle duration compared with the total execution time. For example, it achieves almost 100% of the total leakage reduction for `ping` and the processor is mostly idle during executing the application.

For some workloads like `wget`, *LDP* can save large leakage compared with *llc-flush*. This is because *LDP* can reduce the total execution time compared with *llc-flush*, thus it can also save leakage energy.

The energy overhead of *LDP* is larger than that of *llc-flush* because some of the prefetched data are not reused as shown in Figure 4.9. If the prefetch efficiency is 100%, which means all of the prefetched data are re-referenced, then the energy overhead of *LDP* is same as that of *llc-flush*. However, for some workloads like `netperf`, the difference is quite small, because the leakage energy is dominant in the total energy consumption of the LLC. On average, the energy reduction brought by *LDP* is almost same as that by *llc-flush*.

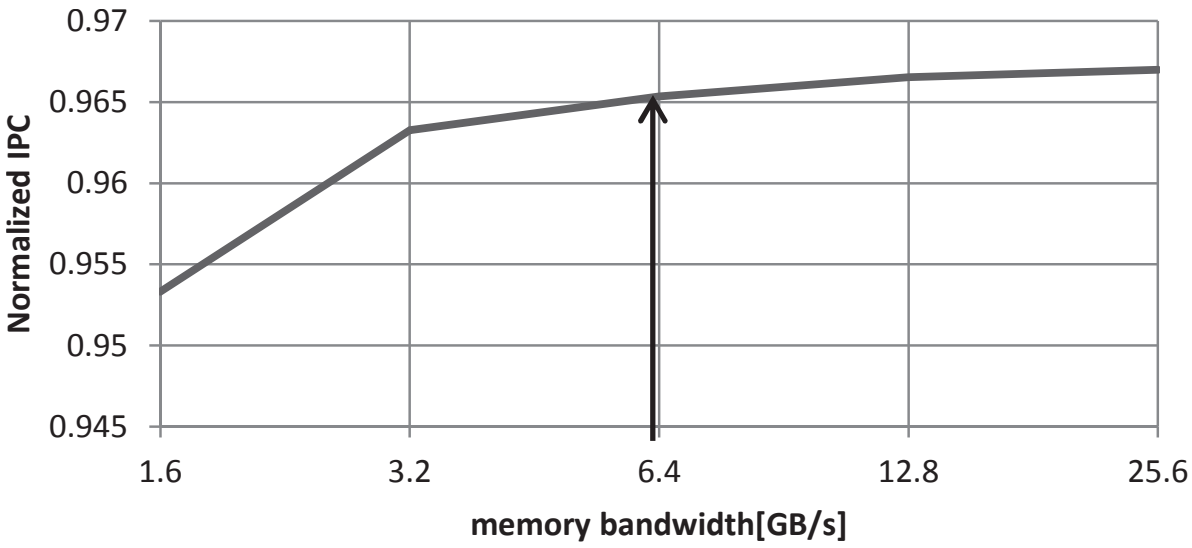


Figure 4.12: Performance as a function of memory bandwidth

Figure 4.11 shows the performance/energy comparison among various methodologies for various workloads. The X-axis shows the relative performance which is normalized to that of *no-flush*, while the Y-axis indicates the relative LLC energy which is normalized to that of *no-flush*. Intuitively, lower right area is better for energy-efficiency. Each plot shows the performance/energy brought by a method for a workload. The results of various methods for the same workload are connected with a broken line. As shown in the figure, *LDP* can improve performance compared with *llc-flush* but the energy reduction is almost same as that of *llc-flush*. If we accept 10% of performance degradation, the LLC can be turned off every idle with *LDP* for all the executed workload. On the other hand, the LLC without *LDP* should be kept awake in some idle duration for some workloads, if the acceptable performance degradation is 10%.

Relationship between performance and some parameters

In this section, we clarify the relationship between performance brought by *LDP* and some parameters. More specifically, we show the performance for various memory bandwidth and various limitation on the number of prefetched pages.

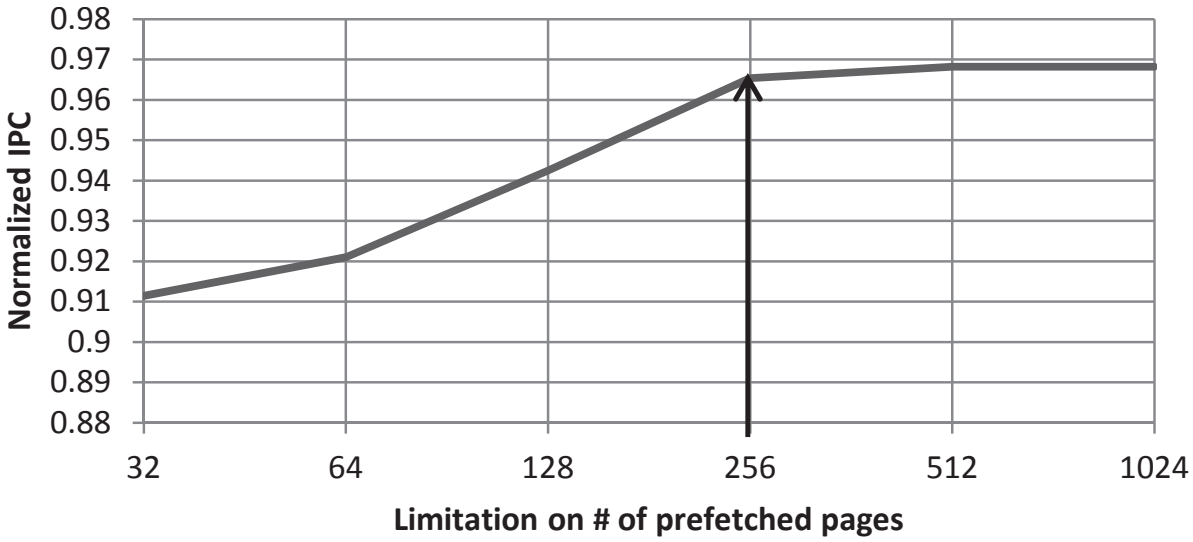


Figure 4.13: Relationship between performance and limitation on # of prefetched pages

Figure 4.12 shows the performance as a function of memory bandwidth. The X-axis shows memory bandwidth, while the Y-axis indicates the average performance brought by *LDP* which is normalized to that by *no-flush*. As shown in the figure, the impact of bandwidth on performance is relatively small. Even at 1.6GB/s, the performance degradation is about 1% compared with that at 6.4GB/s. The bandwidth of one channel DDR-200 based main memory is 1.6GB/s, which is installed in quite old systems. Thus memory bandwidth does not affect the efficiency of *LDP* in modern systems.

Figure 4.13 demonstrates the performance as a function of the maximum number of prefetched pages. In the figure the X-axis indicates the limitation on the number of prefetched pages, while the Y-axis shows average performance brought by *LDP*. The performance is normalized to that brought by *no-flush*. As shown in the figure, the performance is saturated at around 256. Therefore, we set the value 256, which means *LDP* stops its operation after fetching 256 pages.

Chapter 5

Immediate Sleep

Implementing last level caches (LLCs) with STT-MRAM is one of the promising approaches for designing energy efficient microprocessors due to its outstanding properties such as non-volatility, low leakage power, high density and high write endurance [20]. However, the peripheral circuits of STT-MRAM LLCs are still suffering from the leakage problem though their memory cells do not consume leakage power due to their non-volatility [24]. To overcome this problem, in this thesis, we apply power-gating to STT-MRAM caches at the granularity of subarray, especially during active state. However, it is ineffective to simply apply conventional runtime power management schemes to STT-MRAM subarrays. Therefore, we propose a novel power management scheme called Immediate Sleep (IS) which turns off a subarray of STT-MRAM caches immediately if the next access is not critical in performance. In order to predict such non-performance-critical accesses for each subarray during active state, we propose a next-access predictor.

The contribution of this chapter is summarized as follows.

- This thesis presents the subarray level power gating to reduce the leakage energy of an STT-MRAM LLC, especially during active state, based on the fundamental difference of performance degradation between SRAM and STT-MRAM power gating.

- We analyzed the subarray accesses, then found that major part of subarray accesses is

not performance-critical in STT-MRAM caches. According to Figure 5.8, the ratio is almost 80%.

- Based on the observation, we propose a new scheme of subarray power management called Immediate Sleep (IS). IS turns off a subarray immediately if the next access to the subarray is not critical in performance. This is going to significantly reduce the static power of STT-MRAM LLCs.

- In order to realize IS, we developed next-access predictor. This predictor helps finding out whether the next access to a subarray is critical in performance. According to our evaluations, it is highly accurate while consuming only a very small amount of power and area.

The rest of the chapter is organized as follows. Section 5.1 covers the leakage problem in STT-MRAM peripheral circuits and subarray-level power-gating to reduce it. Section 5.2 presents the limitation of timeout based scheme and motivation for Immediate Sleep. Section 5.3 describes our technique for improving the energy efficiency of STT-MRAM LLCs. Section 5.4 presents the experimental methodology and result.

5.1 Energy impact of STT-MRAM peripheral circuits

5.1.1 Leakage problem in STT-MRAM peripheral circuits

To enable low-energy and fast memory accesses, driving large write current to its memory cells is required for an STT-MRAM cache. It is well-known that the write current can be relatively smaller if a wider write pulse is utilized for the write operations, which makes its write latency longer [20]. However, in that case, the write access energy will also be higher, because the write current can be half at best, even if the write pulse is extended 10 times wider [20]. Moreover, to avoid read disturbances, the read current must also be smaller, which also makes the read access latency longer [20]. This is unacceptable because read

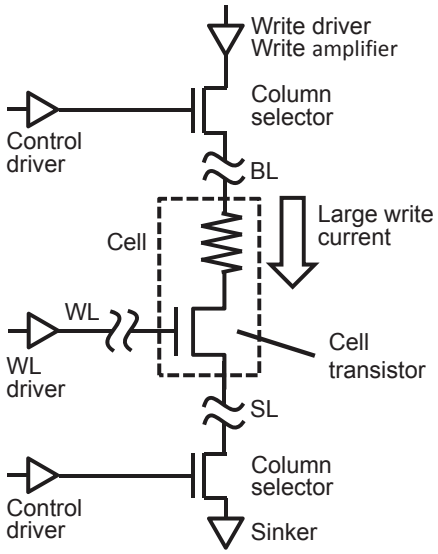


Figure 5.1: STT-MRAM memory cell

accesses are generally critical for the performance of many systems [53]. For these reasons, driving large write current is required for energy efficient STT-MRAM caches.

To drive large write current, large and leaky transistors are required for peripheral circuits in STT-MRAM caches [23]. First of all, the cell transistor shown in Figure 5.1 must be designed with enough drive ability margin, thus it must be large [54]. To drive large write current to the memory cell, the write driver and write amplifier are also large [55]. Moreover, the column selectors shown in Figure 5.1 must also be manufactured with large and leaky transistors because they are on the same current path as the cell transistor. Because the word line driver and column control drivers are connected to these large transistors as shown in Figure 5.1, they are also large and leaky. Thus, the peripheral circuits of STT-MRAM caches consume much leakage power. Furthermore, as VLSI technology is scaled down, the current drive ability of a transistor decreases. Therefore the peripheral circuits will be relatively larger and consume larger leakage power in the future.

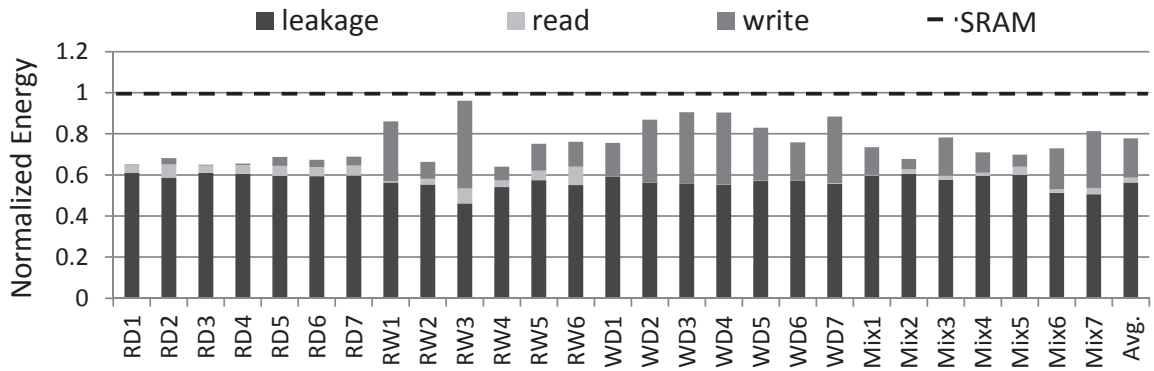


Figure 5.2: Energy breakdown of an STT-MRAM LLC

5.1.2 Quantification of leakage impact of STT-MRAM caches

Figure 5.2 presents the energy breakdown of an STT-MRAM cache used as an LLC. The Y-axis represents energy, while the X-axis indicates executed workloads. The energy of the STT-MRAM LLC is normalized to that of an SRAM LLC which has the same area. The parameters of the STT-MRAM are based on a state-of-the-art device with 32-nm HP CMOS technology and the perpendicular MTJs, which has relatively small write access energy [24, 56], while the parameters of SRAM were calculated by CACTI with 32-nm CMOS technology [50]. Note that our experiment does not include power of tag arrays in both caches. This is because power of a tag array is negligible due to its small size (around 1/10 of the capacity of a data array in the SRAM cache). The detailed simulation parameters and workloads will be presented in Section 5.4.

As shown in the figure, leakage energy is dominant in the state-of-the-art STT-MRAM LLC for many workloads. On average, it accounts for 72% of the total energy consumption of the STT-MRAM LLC. Because of the large leakage energy, the total energy consumption of the STT-MRAM LLC is 78% compared to that of the SRAM LLC on average. The impact of write energy is also large for some workloads such as `WD4`, however it can be reduced by the existing techniques such as EWT (Early Write Termination) [22]. Thus, we need to focus

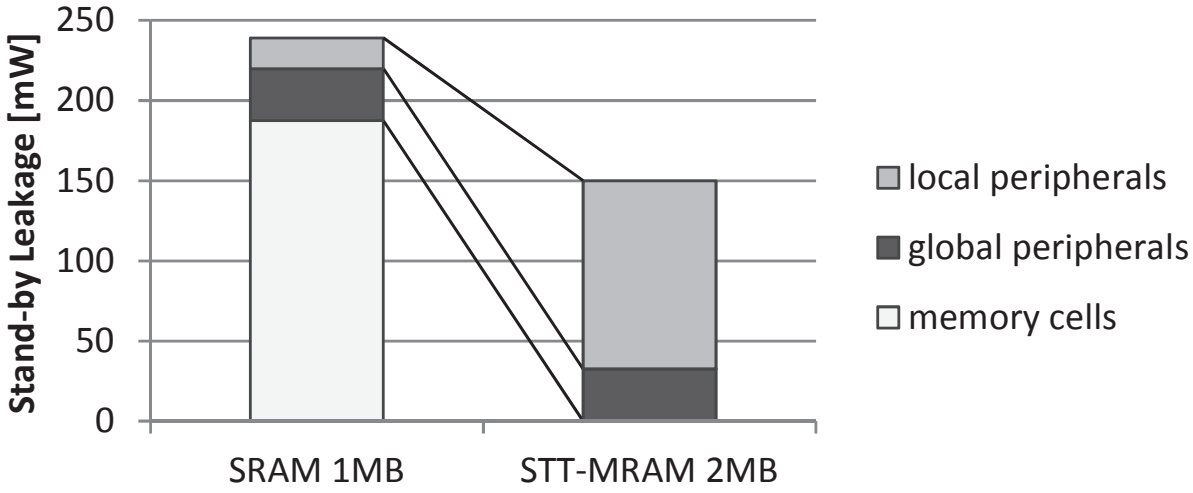


Figure 5.3: leakage breakdowns of SRAM/STT-MRAM LLCs

on reduction of leakage energy of STT-MRAM caches, though the write access energy has been regarded as the most critical problems [21, 23, 22, 57].

Figure 5.3 presents the leakage power breakdowns of an 1MB SRAM LLC and a 2MB STT-MRAM LLC which have the same area. The local peripherals represent such as write drivers and word line drivers, while the global peripherals contain such as pre-decoders, global output drivers, htrees. The parameters of the SRAM and STT-MRAM LLC are the same as those utilized in the evaluation shown in Figure 5.2.

As shown in Figure 5.3, since the local peripheral circuits of the STT-MRAM LLC consume large leakage power, total leakage power of the STT-MRAM LLC is comparable to that of the SRAM LLC, though the leakage power of memory cells is almost zero due to its non-volatility. The leakage power of the global peripheral circuits of the STT-MRAM LLC is small compared to its local peripheral circuits because they do not need large transistors utilized for driving large current. Thus we focus on the leakage reduction of the local peripheral circuits of STT-MRAM LLCs.

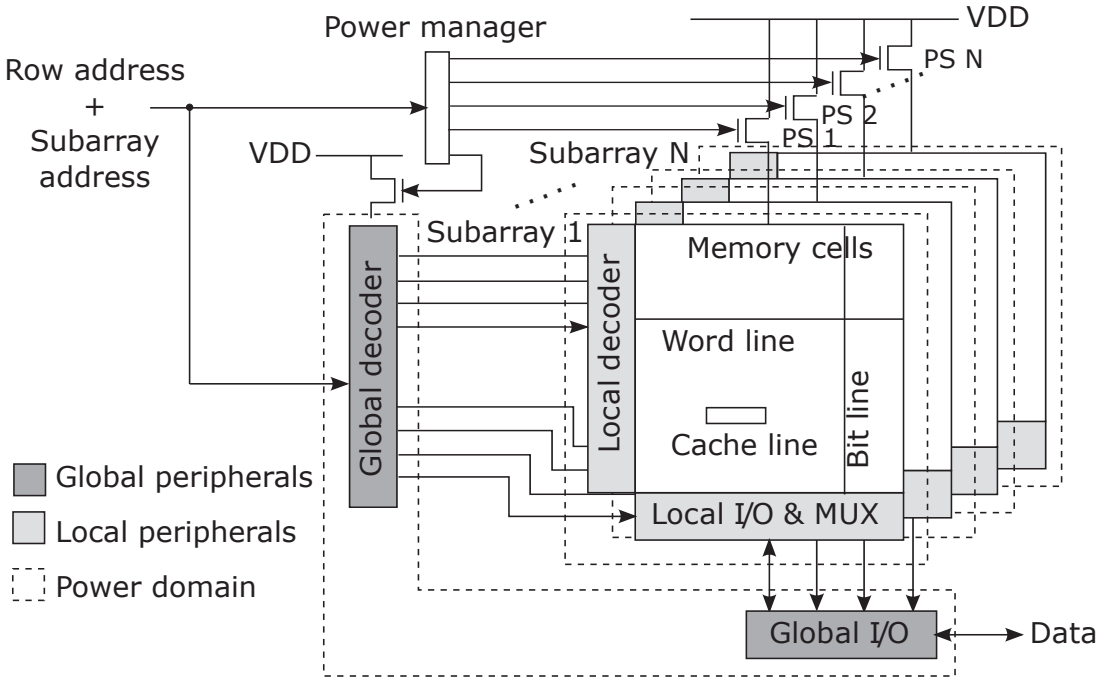


Figure 5.4: Subarray-level power gating for a data array

5.1.3 Runtime subarray-level power gating

Figure 5.4 illustrates architecture of a data array in an STT-MRAM cache under the assumption that tag and data arrays are sequentially accessed in the cache. As shown in the figure, the data array consists of some subarrays and global peripheral circuits. In addition to memory cells, each subarray contains local peripheral circuits such as a local decoder, sense amplifiers, local multiplexers and I/O drivers. The local decoder asserts a word line, and then read current flows into the MTJ devices within the word through bit lines. The amplitude of read current is sensed by sense amplifiers connected to bit lines, and then amplified signals are transferred to global peripheral circuits.

To reduce leakage power of local peripheral circuits, this thesis applies a power-gating technique to STT-MRAM LLCs at the granularity of subarrays. As shown in Figure 5.4, a power switch is inserted between each subarray and VDD. A power manager controls this switch depending on cache-access patterns during active state. It is possible to implement

power gating per each component of a subarray such as decoder, write-driver and multiplexer as shown in [24]. However, this implementation increases the area of the local peripheral circuits, though they occupy larger area as VLSI technology is scaled down as described before. Moreover, most of the components such as write-driver and multiplexer circuits need to be activated at the same time if the subarray is accessed. Therefore, we consider subarray-level power management in this thesis.

Although subarray-level power gating is widely used on SRAM LLCs in modern microprocessors [5], it is not used during active state because of its large performance/energy overheads. For example, if a subarray of an SRAM cache is powered off, all data stored in the subarray are lost and then it dramatically increases cache-miss rate and data-transfer energy. However, STT-MRAM LLCs do not suffer from this problem. Moreover, powering on/off an SRAM subarray is much slower than STT-MRAM subarray because of the large charge-discharge capacity of SRAM memory cells. Therefore, we propose a new power-management scheme which is dedicated for STT-MRAM LLCs having small performance/energy sleep overheads.

5.2 Motivation for an alternative approach to timeout

A timeout based power management, by which a power domain is turned off when it has not been accessed for a certain amount of time called timeout interval, is well known as an effective way for power management of SRAM caches [32]. However, using conventional timeout for runtime subarray-level power gating still wastes the significant amount of leakage energy of subarrays. Section 5.2.1 quantifies this problem. To alleviate this problem, this thesis focuses on cache accesses that have less performance impact of subarray-level power gating as described in Section 5.2.2 and 5.2.3.

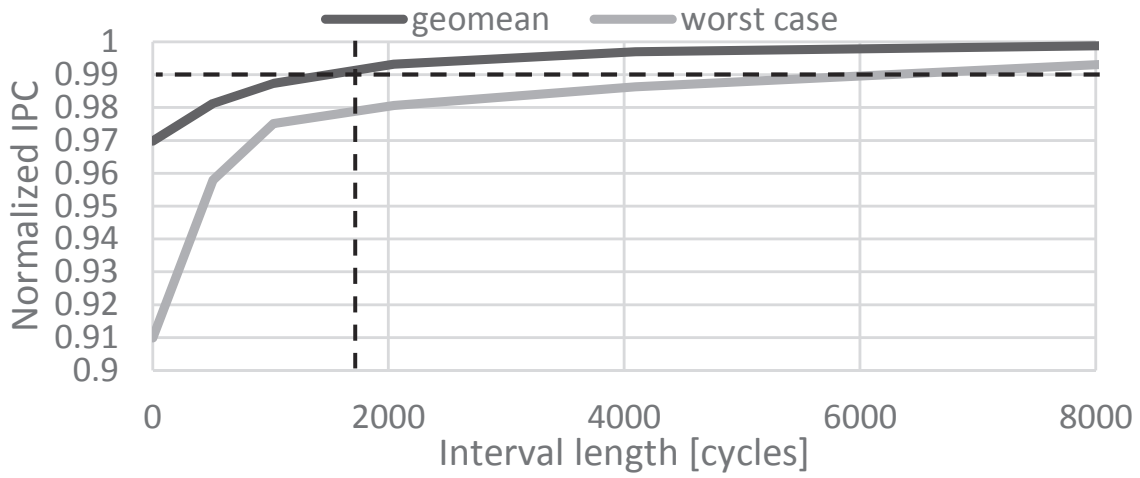


Figure 5.5: Relationship between performance and timeout interval

5.2.1 Waste of energy under timeout controls

Figure 5.5 shows the performance degradation caused by a timeout based power gating as a function of the timeout interval. The X-axis shows the timeout interval, while the Y-axis shows IPC which is normalized to IPC without power gating. In the figure, the geometric mean and the worst case value of all the executed workloads are reported for each interval. As shown in the figure, if we assume that 1% performance degradation on average is acceptable, then the optimal timeout interval is about 2K cycles.

Figure 5.6 shows the relationship between performance and leakage reduction rate achieved by runtime subarray-level power gating with a conventional timeout scheme and the other methodologies. "TO" means the timeout based power control. The other methodologies show the room for improvement of "TO" and the detailed information of them will be shown in the later part of this section. In the figure, the X-axis represents leakage reduction rate of all the executed workloads, while the Y-axis represents relative IPC. Intuitively, upper right area represents that performance is better and leakage is further reduced. A value plotted in the figure is averaged over all workloads. The number at each plot represents the associated

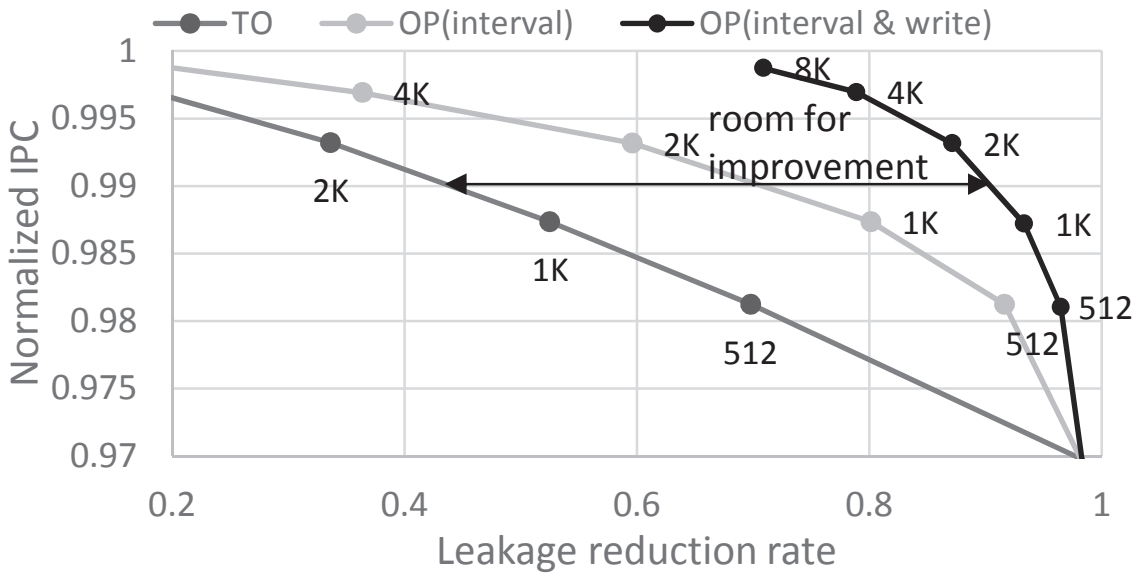


Figure 5.6: Relationship between leakage reduction rate and performance for various power management schemes

interval length. The detailed information of the executed workloads will be described later. As shown in Figure 5.6, if we try to suppress the performance degradation within 1%, the conventional timeout scheme achieves only 44% leakage reduction. Thus, runtime subarray-level power gating with conventional timeout has the limited capability to save leakage power of an STT-MRAM LLC.

5.2.2 Non-performance-critical accesses and their energy impact

If we consider subarray-level power gating for STT-MRAM LLCs, there are two types of cache accesses which can be regarded as non-performance-critical. One is a write access because it is well known that delaying write accesses to an cache has negligible impact on processor performance [53]. The other is an access which arrives with long time interval since the previous access to the subarray because turning off a subarray in this case causes relatively small impact on performance as shown in Figure 5.5. More specifically, we can save relatively large leakage energy with small performance impact by turning off subarrays

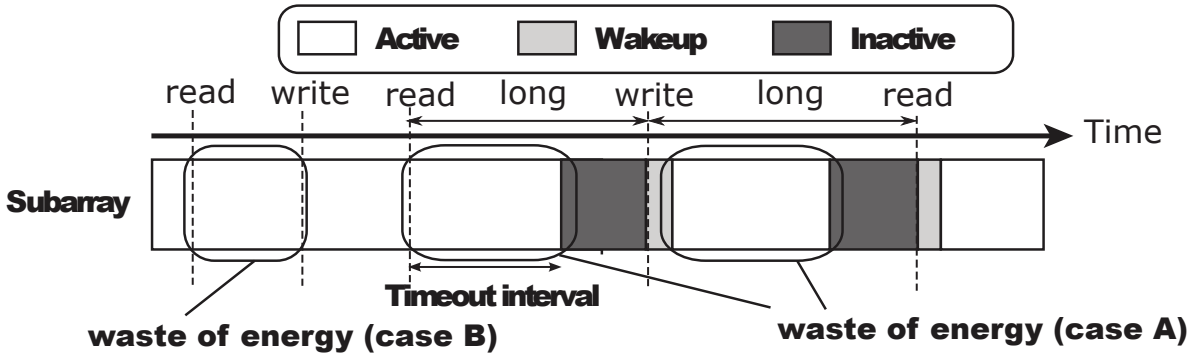


Figure 5.7: Waste of energy under conventional timeout

during such long access intervals because we can minimize the number of wakeup operations to save a certain amount of leakage energy.

As the conventional timeout scheme can not identify such non-performance-critical accesses efficiently, it would waste significant amount of leakage energy if the scheme is applied to STT-MRAM LLCs as shown in Figure 5.7. Although the conventional timeout scheme can identify accesses with long interval, the subarray needs to be kept awake and wastes leakage energy during the timeout interval (case A). Furthermore, a subarray is not turned off until meeting the timeout even if the next access is a write, which leads to the waste of energy because turning off the subarray has negligible impact on performance (case B). As seen from this figure, it is important to turn off a subarray immediately after an access has been completed if the next access is non-performance-critical. We call this control IS (Immediate Sleep).

5.2.3 Quantification of the energy impact of non-performance-critical cache accesses

Figure 5.8 shows the breakdown of accesses to an LLC. We classified all the cache accesses into 4 categories with two criterions: read/write, and long/short. A long access means that the interval between the access and the previous access to the same subarray exceeds a

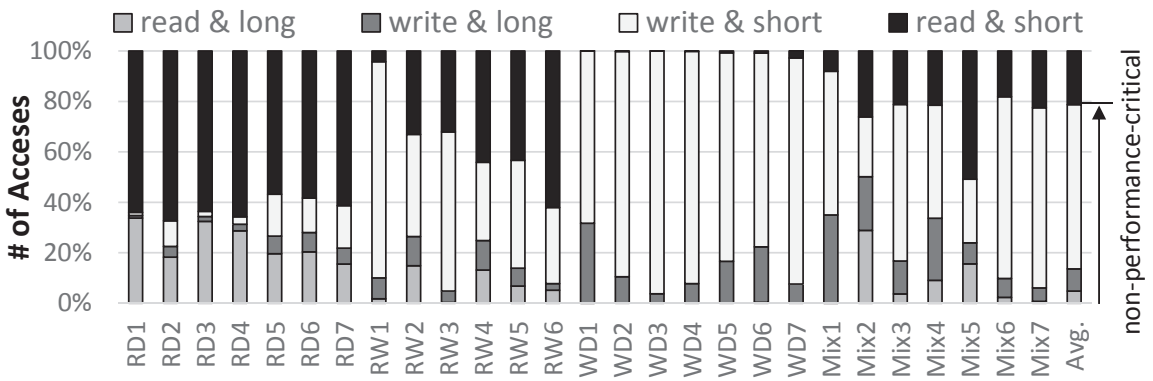


Figure 5.8: Breakdown of LLC accesses

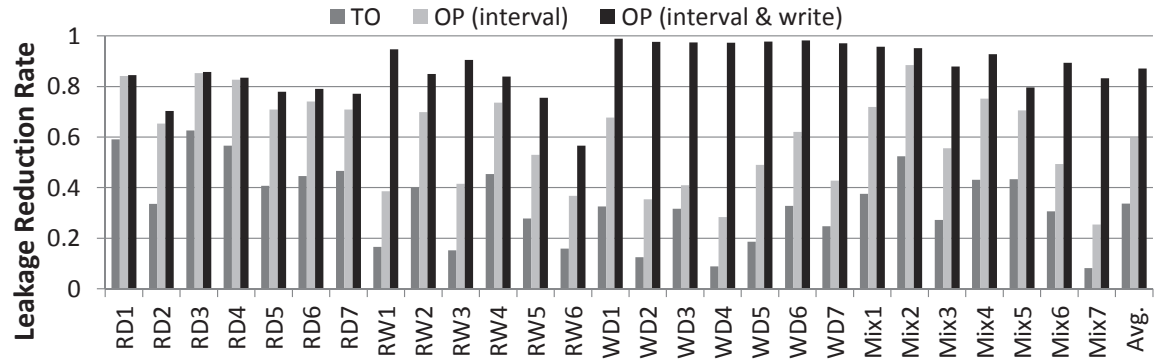


Figure 5.9: Leakage reduction caused by runtime subarray-level power gating with oracle prediction

certain interval, and a short access means vice versa. In this experiment, the interval is set to 2K cycles. The figure shows that non-performance-critical accesses (i.e., write or long accesses) account for 80% of cache accesses. Therefore, there still remain many chances to save leakage energy of subarrays without performance degradation.

Figure 5.9 illustrates leakage reduction brought by runtime subarray-level power gating with the ordinary timeout and oracle prediction. The oracle prediction represents the situation where the power manager ideally tells the type of the next access beforehand. If the next access is non-performance-critical, the manager applies IS (Immediate Sleep) to the

subarray. The figure has 3 bars for each workload: “TO” means the conventional timeout; “OP(interval)” represents the oracle prediction that can only tell whether the next access is long or not; and “OP(interval & write)” means the oracle prediction that can tell both long or not and write or not. In this evaluation, an access which arrives after 2K cycles is regarded as a long access for all of the schemes. Therefore, the timeout interval is also set 2K cycles for “TO” to identify long accesses. On average, “OP(inverval & write)” improves leakage reduction rate by more than 50% compared to “TO”. “OP(interval)” is useful for `RD1-RD7` which are workloads having many long read accesses. On the other hand, the write prediction is effective for `WD1-WD7` which are write intensive workloads as described later.

Finally, we explain the relationship between performance impact and leakage reduction by the oracle prediction based power managements “OP(interval)” and “OP(interval & write)” shown in Figure 5.6. The number at each plot represents the associated interval length for determining each access long or not for these methods. “TO” also utilizes these values to identify each access whether long or not by setting the timeout interval as one of the values. As shown in the figure, these oracle based predictions show large leakage reduction compared to “TO” at the same performances.

These results motivate us to develop a new control scheme IS, which turns off a subarray immediately if the next access to the subarray is non-performance-critical. The next section describes our control scheme in detail.

5.3 Immediate Sleep: reducing the energy impact of peripheral circuits

The proposed scheme turns off a subarray of STT-MRAM LLC immediately after an access if its next access is non-performance-critical. This section describes the details of the scheme and its implementation.

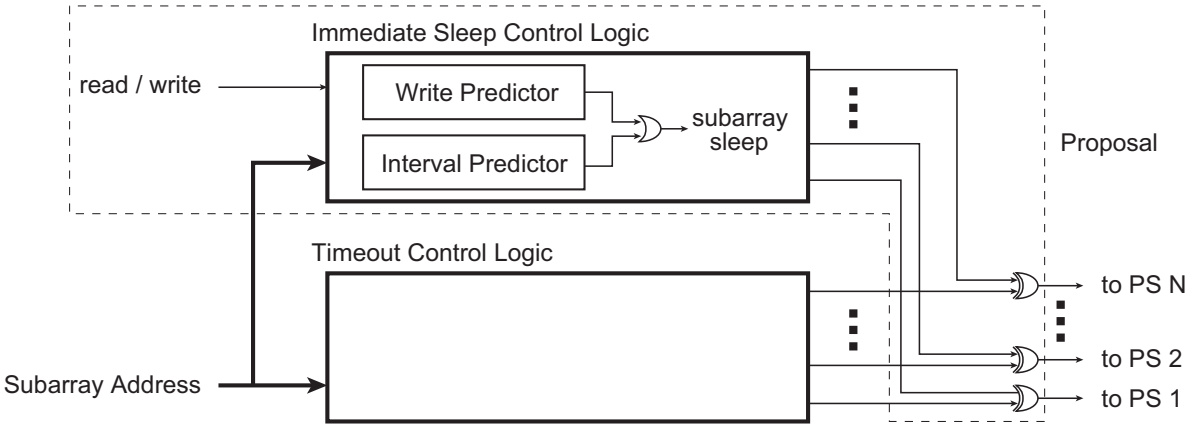


Figure 5.10: Overview of Proposed Power Management

5.3.1 Overview

Figure 5.10 illustrates the overview of the proposed power manager. The immediate sleep control logic generates a signal to assert the power switch of the subarray when the next access to the subarray is predicted to be non-performance-critical. The signal is immediately negated when the subarray is accessed.

To predict the type of the next access, this logic has two predictors: Write Predictor (WP) and Interval Predictor (IP). WP predicts whether the next access is a write operation or not, whereas IP predicts whether the next access is long or not for each subarray. Both predictions are carried out every access in parallel with the subarray access. If either of the prediction results is true, the subarray is turned off immediately after the access finishes.

If non-performance-critical accesses are predicted as critical, chance of power gating is lost which leads to the waste of significant leakage power. In order to avoid such energy waste, the power manager also includes an ordinary timeout-control logic which turns off a subarray based on the timeout. We can also utilize the logic to confirm whether the last access to a subarray was long or not and feedback the information to IP.

The effectiveness of our power manager relies on the accuracy of the two predictors. We extend a history-based approach to these predictors, which is widely used for branch

prediction [58] and cache-hit/miss prediction [59]. The next section explains the predictors.

5.3.2 Prediction of non-performance-critcal subarray accesses

Figure 5.11 shows the concept of our history-based predictor. As the prediction schemes of WP and IP are conceptually identical, the figure illustrates only one predictor. The binary sequence shown in the figure indicates the past access history for a given subarray (note that the history for WP and that for IP are not the same). In WP, "1" represents a write access to the subarray, while "0" represents a read access to it. In IP, "1" represents an access after long interval, while "0" represents an access not fit the case. WP and IP have their own access histories for each subarray. The histories for the subarray are updated every subarray access.

With this access history, it is predicted whether the next access to the subarray is performance-critical or not. Here, x and n represents the next access to the subarray and the length of the history h used for prediction, respectively. First, a predictor compares the probability of $x = 1$ ($P(x = 1|h)$) to the probability of $x = 0$ ($P(x = 0|h)$). Then, if $P(x = 1|h) > P(x = 0|h)$ holds, a predictor outputs 1 (i.e., the next access is not expected to be performance-critical). Otherwise, the predictor outputs 0. To achieve this, we have only to estimate $P(x = 1|h)$, since $P(x = 1|h) + P(x = 0|h) = 1$ is always established for all h.

Figure 5.12 shows the accuracy of static predictors. Here, it is assumed that $P(x = 1|h)$ for all the accesses of each workload are given in an offline manner, and the prediction is statically decided based on the information. The X-axis shows history length n, while the Y-axis shows the prediction accuracy. "WP(Avg.)" and "IP(Avg.)" represents the prediction accuracy of WP and IP averaged over all programs respectively, while "WP(RW3)" and "IP(RD3)" shows the prediction accuracy of WP for `RW3` and IP for `RD3` respectively. In this evaluation, we regard an access which arrives after 2K cycles as a long access. As shown in the figure, the prediction accuracy of both WP and IP exceed 84%, even if the history length n is 1 and the prediction is static. However, as shown in this figure, longer history length

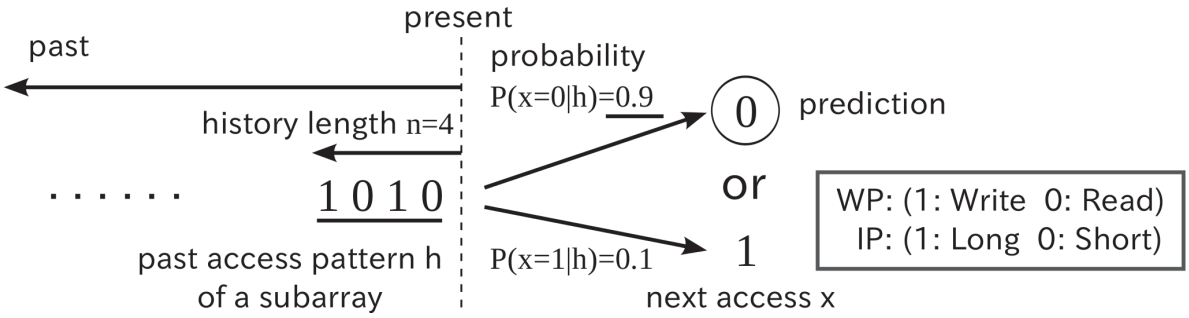


Figure 5.11: History-based prediction of non-performance-critical subarray accesses

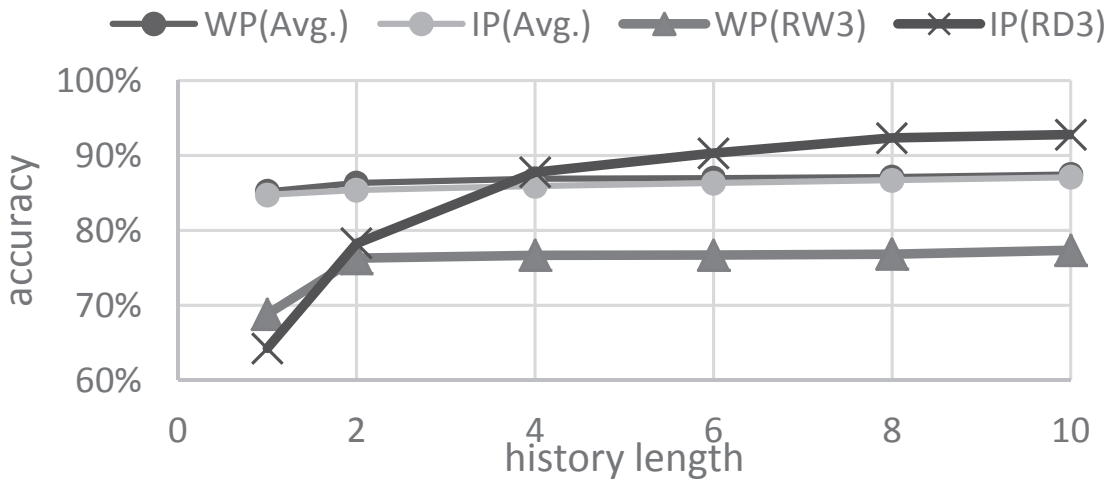


Figure 5.12: Accuracy of static history-based predicion

is required for the workload `RW3` and `RD3` to achieve accurate prediction. This is because a subarray is accessed regularly during executing those workloads. Thus, a history-based approach is also useful for prediction of non-performance-critical subarray accesses, especially for the workloads which produce regular access patterns in subarrays.

5.3.3 Implementation of predictors

Figure 5.13 explains the implementation of each predictors. As previously mentioned, WP and IP are the same except for the actual history data. A predictor consists of two tables as well as a 2-level adaptive branch predictor[58]. One is a history table that consists of shift registers keeping access histories. The number of shift registers is the same as the number of

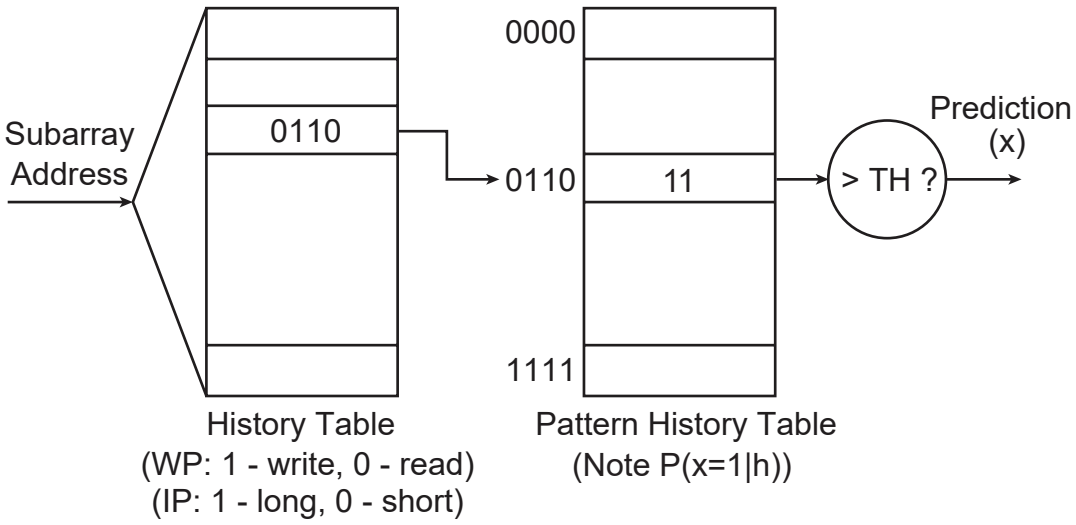


Figure 5.13: Implementation of predictors

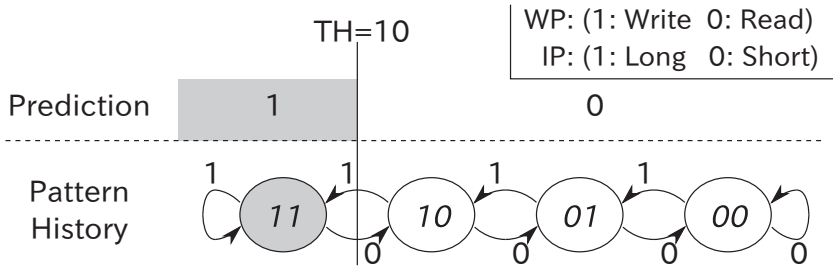


Figure 5.14: State transition and prediction

subarrays. The other is a pattern history table that consists of some saturating counters having access tendencies. When an access reaches the power manager, the history table is referred with its subarray address, and then the access history for the subarray is read out. Next, the pattern history table is accessed with the above history, and then $P(x = 1|h)$ for the history is loaded. If the loaded value is greater than a threshold value (TH), the predictor outputs 1. Otherwise, it outputs 0.

All tables are updated when the above prediction process finishes. Depending on the type of the current access (i.e., write/read, and long/short), the predictor updates two shift registers used for prediction (i.e., the related shift registers in WP and IP, respectively). Moreover, the

predictor counts up or down two saturating counters used for prediction, according to the access type. As shown in Figure 5.14, the value of the related counter is incremented if $x = 1$ follows the access history. It is decremented if $x = 0$ follows the history.

Table 5.1: Simulation paramaters utilized for the evaluation of IS

Name	Remarks
CPU	2core, 2GHz, alpha, in-order 4-way fetch/decode/issue
L1 D/I cache (SRAM)	32KB, 1-cycle latency 4-way set assoc., 64B line
L2 cache (SRAM)	1MB, 14-cycle latency 16-way set assoc., 64B line 8-entry write buffers
L2 cache (STT-MRAM)	2MB, 16-cycle/20-cycle latency(read/write) 16-way set assoc., 64B line, 8-entry write buffers 64KB subarrays, 10-cycle wake-up latency
Main memory	200-cycle latency
predictors	2bits/4bits history(WP/IP), 2bits/2bits counters(WP/IP), $10_2/01_2$ threshold TH(WP/IP)

5.4 Evaluation

For evaluation, we implemented our method on a cycle-accurate processor simulator. This section describes the experimental methodology and result.

5.4.1 Evaluation environment

We implemented the proposed method on Gem5[49], which is the simulator that models detailed memory systems including non-blocking caches and write buffers for lower level caches. The simulation parameters are summarized in Table 5.1. The processors consist of 2-level caches (i.e., level 2 is the last level), so we used an STT-MRAM cache and our power-gating method for the L2 cache. The feature size of a state-of-the art STT-MRAM cell is the half of that of SRAM cell[56]. That is, the memory cells of an STT-MRAM can be doubled under an area constraint. Therefore, we assumed that the STT-MRAM cache has twice the capacity of the SRAM cache in our simulation. Note that the STT-MRAM cache consists of

32 subarrays, with each subarray having 64 KB.

Table 5.2: Energy parameters utilized for the evaluation of IS

Name	Remarks
L2 cache (SRAM)	0.421[nJ/access] dynamic, 239[mW] leakage
L2 cache (STT-MRAM)	0.428/1.20[nJ/access] dynamic(read/write) 118/0.174[mW] total leakage of subarrays(awake/sleep) 0.078[nJ/sleep] PG overhead for a subarray
predictors (IP & WP total)	0.00601[nJ/prediction] dynamic, 0.00629[mW] leakage

Table 5.3: Classification of SPEC2006 benchmarks

Class	Benchmarks
Read Dominant	GemsTDTD, tonto, gcc, namd, sphinx3, h264ref, povray, gromacs
Read-Write	sjeng, astar, soplex, bwaves, hmmer, omnetpp, dealII, gamess, xalancbmk
Write Dominant	libquantum, milc, gobmk, zeusmp, lbm, bzip2, cactusADM, lbm, leslie3d, wrf

The energy parameters of the STT-MRAM LLC and the SRAM LLC are shown in Table 5.2. The parameters of the STT-MRAM LLC such as dynamic energy and leakage power are derived from a state-of-the-art STT-MRAM with 32-nm high-performance CMOS technology[56]. We also estimated the leakage power of peripheral circuits of the STT-MRAM LLC with CACTI 6.5[50], and then it turned out the total leakage power of local peripheral circuits is 118 mW (3.69mW per subarray). The leakage power of a subarray during the sleep state was estimated 0.00544 mW, thus the local peripheral circuits consume only 0.174 mW if they are all turned off as shown in Table 5.2. The recovery latency from sleep state is 10 cycles and the energy overhead of power gating for a subarray is 0.078 nJ as shown in Table 5.1 and Table 5.2. These values were estimated with the RC net-list from a 64KB memory array. Because these parameters depend on the implementation of the power switches, we optimized the switches to minimize the wakeup latency under certain area and

energy constraints with our in-house circuit simulator.

The parameters of WP and IP are listed in the last row of TABLE 5.1. The prediction tables are the major reason for area increase but their size is only 29B for the 2MB STT-MRAM LLC. We also calculated the energy overheads of these predictors with SPICE assuming that they are designed as register files. In the last row of TABLE 5.2, the calculation results are represented as the total dynamic/leakage of all tables in our predictor. Because they are negligible compared to the leakage power of the total local peripherals (118mW), the overheads did not affect the total energy saving caused by our technique in our evaluation shown later. The prediction is carried out in parallel with the cache access, thus it also does not affect the processor performance at all. Therefore we did not take the prediction latency into consideration.

We evaluated our methodology in multiprogrammed environments on the multi-core processor using 27 programs from SPEC CPU 2006 benchmark suits. We did not use `perlbench` and `calculix` because Gem5 cannot execute them normally[49]. As shown in TABLE 5.3, these programs are divided in 3 categories, depending on read/write access frequency. Read intensive programs are classified as *Read Dominant*, while write intensive programs are classified as *Write Dominant*. The others are classified as *Read-Write*. In our evaluation, we utilize workloads called *RD**, *RW**, *WD** and *Mix**. In *RD**, 2 benchmarks are randomly selected from *Read Dominant* and are executed on each core. Similarly in workloads *WD**/*RW**, 2 benchmarks are randomly selected from *Write Dominant*/*Read-Write* respectively and are executed on each core. In workloads *Mix**, 2 benchmarks are selected randomly from different categories and executed on each core. By doing so, we can evaluate various access patterns on the LLC of the multi-core processor. Each program of a workload is fast-forwarded for 500M instructions, and the workload is simulated until all programs have executed 500M instructions. We report aggregated IPC in the evaluations.

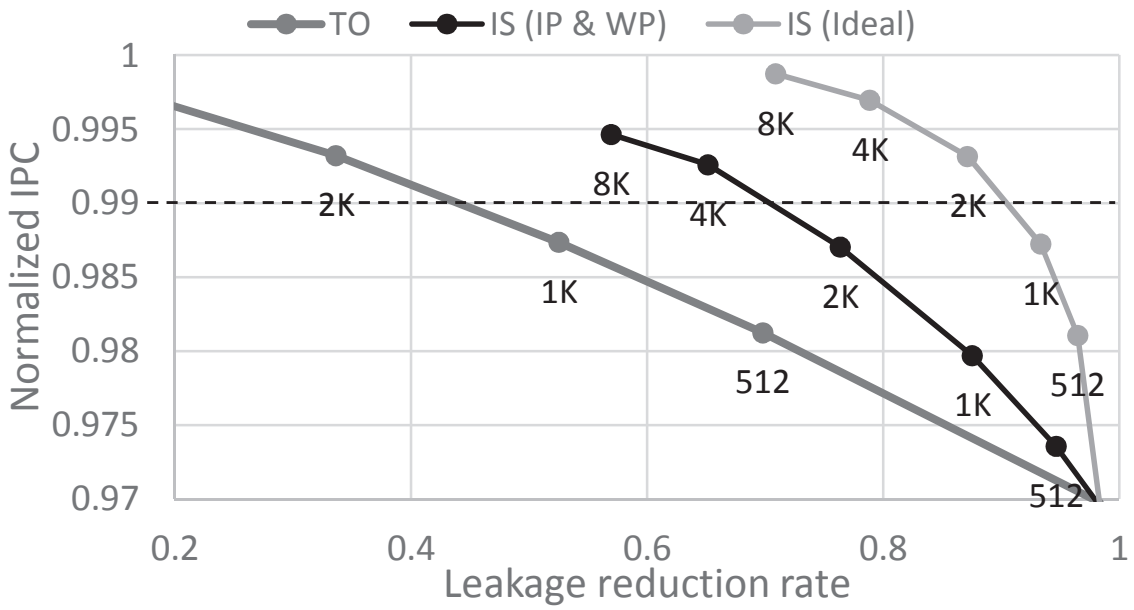


Figure 5.15: Relationship between IPC and leakage reduction rate for various methods

5.4.2 Experimental results

Figure 5.15 shows the relationship between IPC and leakage reduction rate for various methods. We compared 3 methods in this evaluation: "TO" means power gating with conventional timeout; "IS (IP & WP)" means power gating with IS controlled by both IP and WP; and "IS (Ideal)" represents power gating with ideal IS whose predication is always correct. The X-axis represents leakage reduction rate of overall subarrays, while the Y-axis represents relative IPC. Intuitively, upper right area represents that performance is better and leakage is further reduced. A value plotted in the figure is averaged over all programs. We changed the intervals set in the leakage/performance evaluations within the range of 512-8K cycles. The number at each plot represents the associated interval length.

As shown in the figure, "IS (IP & WP)" saves more leakage energy than "TO" at the same performance degradation. For example, "IS (IP & WP)" saves 71% of leakage energy whereas "TO" saves only 44% of that at IPC degradation of 1%. This indicates that "IS

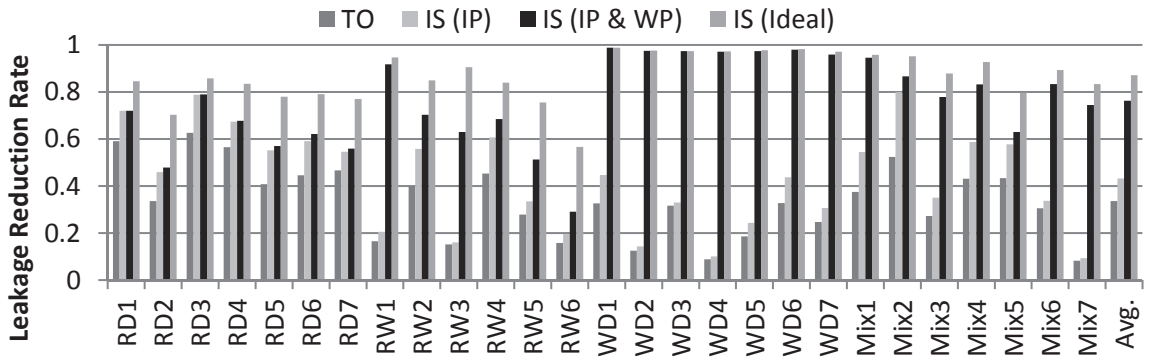


Figure 5.16: Leakage reduction rate for various schemes

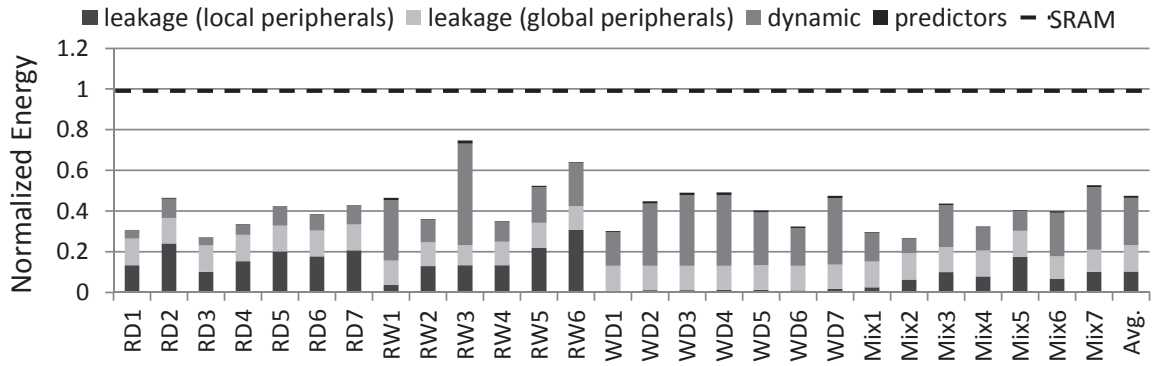


Figure 5.17: Overall LLC energy consumption for "IS (IP & WP)"

(IP & WP)" consumes 48% (1-(1-0.71)/(1-0.44)=0.48) less leakage energy than "TO" (32% total leakage saving of the STT-MRAM LLC) at that performance. We have to select the acceptable performance degradation properly with considering the energy consumption of the other components of the processor because increasing total execution time also increase their energy consumption. However, at any performance, our methodology shows large leakage reduction compared to the conventional timeout based control.

Figure 5.16 shows the leakage reduction rate of overall subarrays of the STT-MRAM LLC for various workloads. The horizontal axis represents benchmark programs, while the vertical axis represents ratio of reduction of leakage energy in the STT-MRAM LLC. The

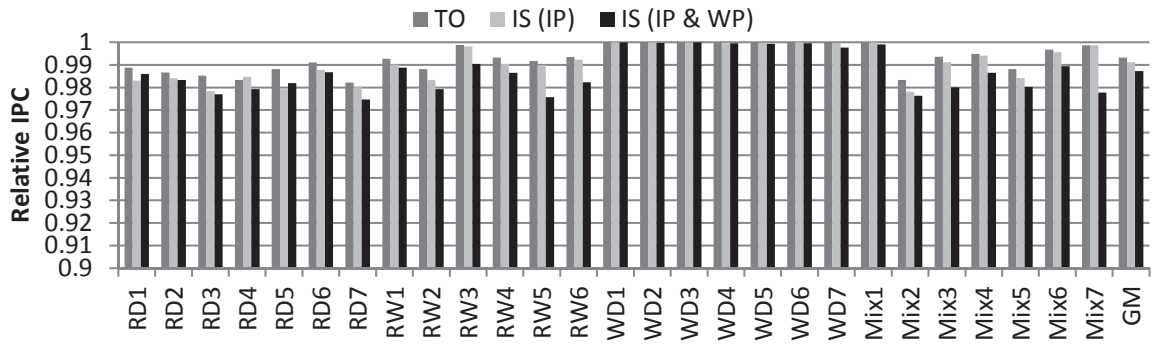


Figure 5.18: IPC for various power-management schemes

reduction of leakage energy is normalized to leakage energy of no power gating. The energy overheads of power gating and predictors are considered in this evaluation. There are 4 bars for each program: “IS (IP)” means power gating with IS controlled by IP; and the others are the same as described above. “TO” use 2K cycles as timeout intervals and the other schemes also regard an access which comes after 2K cycles as long. As shown in the figure, “IS (IP & WP)” achieves leakage reduction of 76% on average. “IS (IP)" (IP based IS) helps reducing leakage power for benchmarks having many long-read accesses like `RD1-RD7`. On the other hand, WP based IS works very well for write intensive workloads like `WD1-WD7`.

Figure 5.17 represents overall energy consumption of the STT-MRAM LLC with “IS (IP & WP)”. The X-axis shows executed workloads. Each bar contains the leakage energy of the local/global peripheral circuits, the dynamic (read/write) energy of the LLC and the energy overheads of the predictors, which are normalized to the energy consumption of the SRAM LLC. The energy overhead of power gating is contained in the leakage energy of the local peripheral circuits. As shown in the figure, most of the energy consumption of the STT-MRAM LLC can be reduced by the proposed methodology compared to Figure 5.2. On average, thanks to “IS (IP & WP)” based power management, the STT-MRAM LLC consumes only 45% energy compared to the SRAM LLC. The dynamic energy is not ignorable for some workloads such as `WD4` and the leakage energy is mainly consumed by the global peripheral

circuits for many workloads. However, they can be significantly reduced by the previous works such as [23, 22, 60].

The processor performance for each scheme is summarized in Figure 5.18. The vertical axis represents relative IPC, which is normalized to IPC of no power gating. As shown in the figure, the performance degradation caused by our scheme is quite small. With the proposed technique, the performance degradation is still around 1% at the average case. This is because our predictors are accurate enough.

Chapter 6

Evaluating parameter requirements of STT-MRAM caches

In this chapter, we evaluate the parameter requirement of STT-MRAM caches depending on the typical load state of a certain system. For systems on which only I/O bound applications are usually executed though operation, the cache access performance/energy less affect the overall system performance/energy because caches are less frequently accessed. Because the cost of STT-MRAM caches highly depends on their parameters, we have to know the parameter requirement to optimize the design. In the first section, the relation between parameter requirement and load state of the system is described. In the second section the models utilized for the evaluation is explained. In the final section, the evaluation methodology and the experimental result are shown.

6.1 Parameter requirements of STT-MRAM for low CPU load systems

In this section, we describe the relation between the parameter requirement of STT-MRAM and the system load state.

Figure 6.1 illustrates the power and execution time of two different processors having an SRAM-based LLC or an STT-MRAM-based one for different system load state. In the figure, the X-axis shows the total execution time, while the Y-axis shows the power, thus

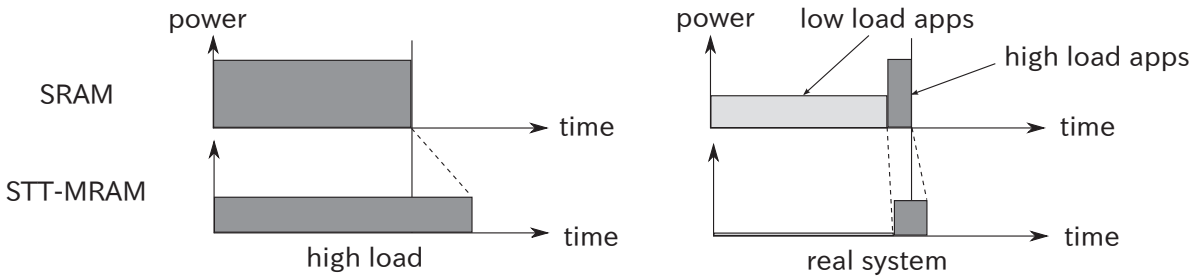


Figure 6.1: Performance and power according to workloads

the total area of the rectangular regions shows the total energy consumption. The *high load* in the figure means only CPU/memory bound applications are executed on the system. On the other hand, both high load applications and low load applications are executed for the *real system* during operating time. Here, the *high load applications* include CPU/memory intensive workloads thus a CPU hardly goes into idle state during executing them. On the other hand , *low load* applications indicate I/O bound applications which make a processor frequently idle. Note that the low load applications and high load ones are executed mixedly along the time axis unlike the graphs shown in Figure 6.1.

As shown in Figure 6.1, the energy reduction and performance degradation brought by STT-MRAM highly depend on the system load state. This is because the energy/performance impact by memory parameters is highly depend on memory access frequency, which is quite different for high/low load applications. More specifically, a processor stays idle for low load applications, during which any memory reference does not occur, thus the impact is small for them. Thus, the performance/energy impact of LLC parameters are relatively small for the systems whose load state is usually low.

Therefore, this thesis clarifies the parameter requirement of STT-MRAM caches according to acceptable performance degradation, target energy reduction rate and typical system load state. In this thesis, the parameter called utilization rate of high load applications are introduced to express the system load state. The parameter is given by the total execution time of the CPU/memory intensive applications per the total execution time of all the applications.

6.2 Formulation and modeling

In this section, this thesis explains the formulation of the problem in 6.2.1. Then, the energy / performance models are described in 6.2.2.

6.2.1 Formulation

To acquire the parameter requirement, we formulated the problem as shown in the equations (6.1), (6.2). In the equations, T_{mts} and E_{mts} indicate the relative execution time and relative energy consumption of the processor having an STT-MRAM LLC which are normalized to those of the processor having an SRAM LLC, respectively. R_{inc}, R_{save} and R_{util} shows the acceptable performance degradation, energy reduction requirement and the utilization rate of high load applications that is described before. For example, $R_{util} = 1$ means only high load applications are executed on the system. T_{lat} and E_{acc} describe the access latency and access energy of the STT-MRAM cache respectively.

$$T_{mts}(T_{lat}, R_{util}) \leq 1 + R_{inc} \tag{6.1}$$

$$E_{mts}(T_{lat}, E_{acc}, R_{util}) \leq 1 - R_{save} \tag{6.2}$$

The first equation explains the performance constraint condition. In the equation, the requirement of latency T_{lat} can be determined when the acceptable performance degradation R_{inc} and the load state R_{util} are given. Note that the relative execution time T_{mts} is independent of the energy parameter E_{acc}.

The second equation describes the energy constraint condition, in which the parameter requirement for (T_{lat}, E_{acc}) are determined when the target energy reduction rate R_{save} and the utilization of high load applications R_{util} are given. The latency T_{lat} is included in the arguments of E_{mts} because increasing execution time also increases leakage energy consumption.

The Figure 6.2 illustrates the area that meets both of the equations above. In the figure,

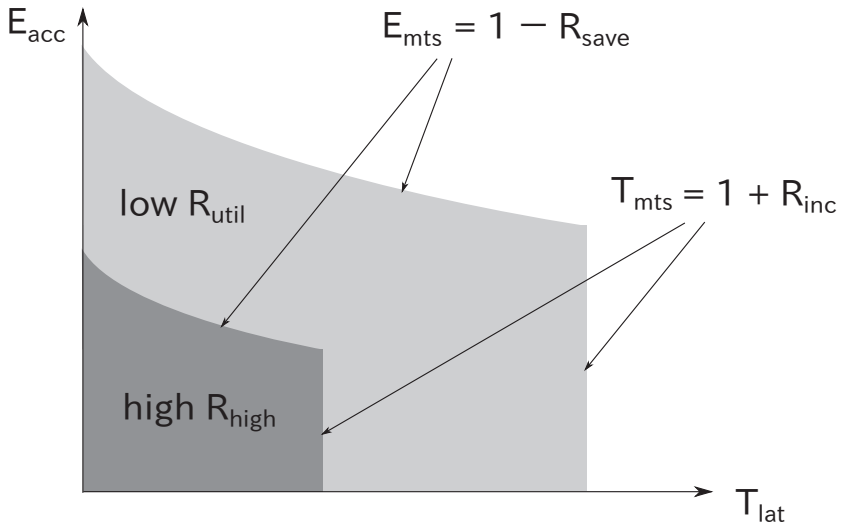


Figure 6.2: The domains to meet the parameter requirements

the horizontal axis indicates the access latency T_{lat}, while the vertical axis shows the energy consumption of the cache E_{acc}. As shown in the figure, if the utilization rate of high load application R_{util} is higher, then the area is smaller under a certain constraints (R_{inc}, R_{save}). To acquire the boundary of the area, we describe the models for execution time and energy consumption in the next section.

6.2.2 Performance / energy calculation

The models utilized in this work are as follows. First of all, the relative execution time and relative energy consumption of an STT-MRAM LLC are expressed as $T^{high}_{mts}(T_{lat})$ / $T^{low}_{mts}(T_{lat})$ and $E^{high}_{mts}(T_{lat}, E_{acc})$ / $E^{low}_{mts}(T_{lat}, E_{acc})$ for high / low load applications. The execution time and energy consumption are normalized to those of an SRAM LLC. Then, $T_{mts}(T_{lat}, R_{util})$ and $E_{mts}(T_{lat}, E_{acc}, R_{util})$ are calculated as shown in the equations (6.3) and (6.4). As shown in the equations, these are expressed as weighted linear sums of above equations. The weight R_{util} or $1 - R_{util}$ show the utilization of high or low load application respectively.

$$
\begin{aligned}
& T_{mts}(T_{lat}, R_{util}) \\
= \;& R_{util} T^{high}_{mts}(T_{lat}) \\
+ \;& (1 - R_{util}) T^{low}_{mts}(T_{lat})
\end{aligned} \tag{6.3}
$$

$$
\begin{aligned}
& E_{mts}(T_{lat}, E_{acc}, R_{util}) \\
= \;& R_{util} E^{high}_{mts}(T_{lat}, E_{acc}) \\
+ \;& (1 - R_{util}) E^{low}_{mts}(T_{lat}, E_{acc})
\end{aligned} \tag{6.4}
$$

The high load applications can be divided into two classes: memory bound applications and CPU bound applications. Because the performance / energy impact of memory parameter is much more significant while executing memory bound application, we consider it in our models. More specifically, for memory / CPU bound applications, the relative execution time and relative energy consumption of an STT-MRAM LLC are expressed as $T^{mb}_{mts}(T_{lat})$ / $T^{cpub}_{mts}(T_{lat})$ and $E^{mb}_{mts}(T_{lat}, E_{acc})$ / $E^{cpub}_{mts}(T_{lat}, E_{acc})$ which are normalized to those of an SRAM LLC. In this thesis, we models $T^{high}_{mts}(T_{lat})$, $E^{high}_{mts}(T_{lat}, E_{acc})$ as the equations shown in (6.5), (6.6). In the equations, R_{mb} is given by the total execution time of the memory bound applications per the total execution time of all the high load applications.

$$
\begin{aligned}
& T^{high}_{mts}(T_{lat}) \\
= \;& R_{mb} T^{mb}_{mts}(T_{lat}) \\
+ \;& (1 - R_{mb}) T^{cpub}_{mts}(T_{lat})
\end{aligned} \tag{6.5}
$$

$$
\begin{aligned}
& E^{high}_{mts}(T_{lat}, E_{acc}) \\
= \;& R_{mb} E^{mb}_{mts}(T_{lat}, E_{acc}) \\
+ \;& (1 - R_{mb}) E^{cpub}_{mts}(T_{lat}, E_{acc})
\end{aligned} \tag{6.6}
$$

To clarify the area, we need to acquire $T^*_{mts}(T_{lat})$, $E^*_{mts}(T_{lat}, E_{acc})$ (* = low, mb, cpub). We got these functions based on the simulation described later.

6.3 Evaluation

In this section, the evaluation methodology and environment are shown firstly. Then, the experimental result is revealed.

6.3.1 Evaluation methodology

First of all, we classify the executed applications memory bound, CPU bound and low load (I/O bound). As described before, memory / CPU bound applications are high load ones. In this research, we utilized SPEC CPU2006 benchmark suite [39] as high load applications. The classification to memory / CPU bound is based on the performance impact of access latency for each workload. The detailed classification will be shown later. For low load applications, we chose some I/O bound workloads as follows.

surge Surge is an I/O bound application installed in the simulator shown later by default. The application issues frequent I/O requests thus a processor often stay in idle during executing the program.

specWeb We executed specWeb benchmark client on the simulator described later. This application utilize the network very frequently thus usually stays in idle to wait for the responses.

bonnie++ bonnie++ is a widely utilized benchmark software for evaluating disk I/O performance. This software issues frequent disk requests thus the processor also frequently turns into the idle state.

find find command is executed at the root directory of the file system. During directory search, disk I/O requests are frequently issued.

netperf netperf is a tool for checking the status of a network. During execution, the machine frequently goes to the idle state.

wget wget is a command to fetch contents from web sites. In our evaluation, a client machine tries to fetch a file having few MB from the other http server.

In the evaluation, we acquire $T^*_{mts}(T_{lat})$ and $E^*_{mts}(T_{lat}, E_{acc})$ (*=mb,cpub,low) for each class. To do so, we evaluate the performance and energy for various cache parameters for each applications then calculate the average performance and average energy for each class with the results.

Furthermore, we input the results $T^*_{mts}(T_{lat})$ and $E^*_{mts}(T_{lat}, E_{acc})$ to the equations (6.3)-(6.6). By doing so, we can acquire $T_{mts}(T_{lat}, R_{util})$ and $E_{mts}(T_{lat}, E_{acc}, R_{util})$. In the evaluation we set R_{mb}, R_{cpub} and R_{low} to various values.

Name	Remarks
OS	linux 2.6.27
CPU	2core, 2.4GHz, alpha, OoO 4-way fetch/decode/issue
L1 D/I cache	32KB, 1ns latency, 8-way set assoc., 64B line
LLC (SRAM)	4MB, 6.4ns latency, 16-way set assoc., 64B line
LLC (STT-RAM)	8MB, 7.8ns latency(read), 9.8ns latency(write), 16-way set assoc. 64B line, 128KB subarrays, 9ns wakeup latency
Main memory	100ns latency, 6.4GB/s bandwidth
Disk	10ms latency
Ethernet	100μs latency, 100Mb/s bandwidth

Table 6.1: Simulation settings

Evaluation settings

We evaluated the performance and energy on the full-system simulator gem5[49]. The parameters utilized in the evaluation are shown inTable 6.1.

The parameters utilized for the energy calculation are shown in Table 6.2. The prameters of SRAM are calculated with CACTI [50], while the parameters of STT-MRAM are based

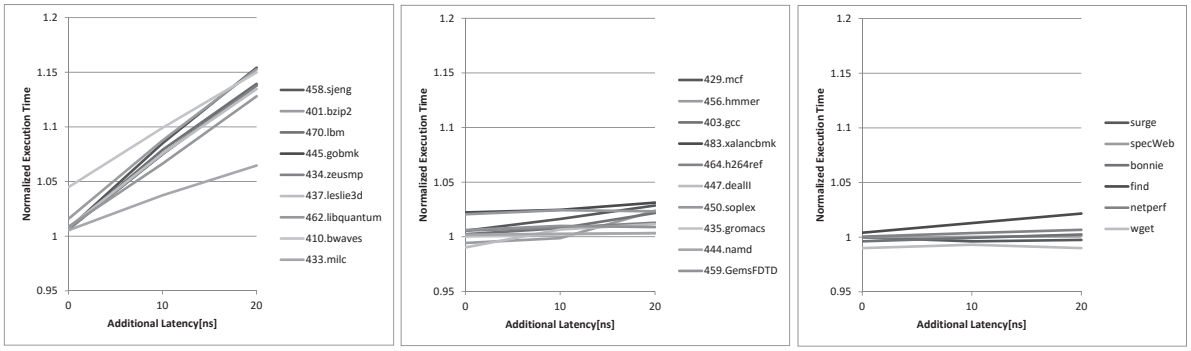
on the previous work [61]. We increased the access energy and access latency of the STT-MRAM cache to acquire the boundary.

In the evaluation, we apply power-gating to both the STT-MRAM cache and the SRAM cache. More specifically, A subarray of the STT-MRAM is turned off based on the decision made by *Immediate Sleep Control Logic* proposed in the last chapter while the processor is active. The interval threshold utilized by the logic is set to 16K cycles in the evaluation with taking the performance impact of power gating into consideration. The other hardware settings of the power manager is based on those shown in the last chapter. During processor idle state, all the subarray of the STT-MRAM cache is turned off thus the leakage during idle is not considered in the evaluation. On the other hand, power-gating is applied to the SRAM cache only when the processor stays idle and the performance degradation is under the performance constraint.

Table 6.2: Energy parameters

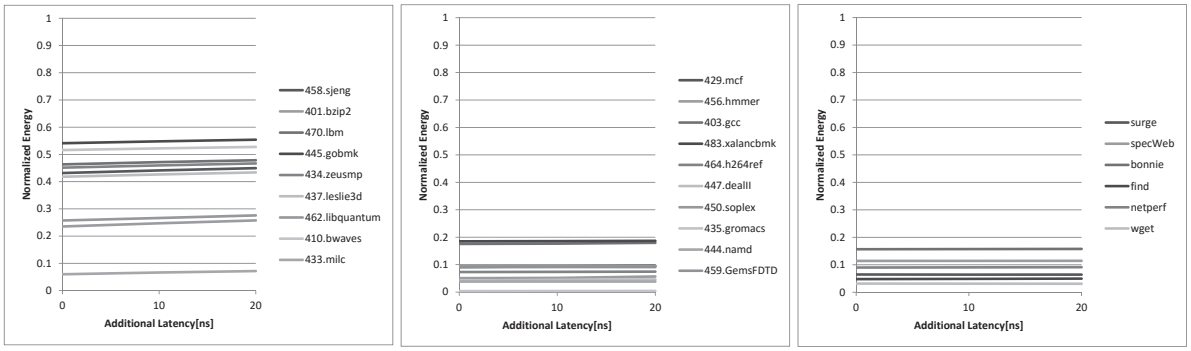
Parameters	Remarks
LLC:	
SRAM:	
Leakage Power:	2.86W
Dynamic Energy:	0.843nJ/line
STT-MRAM:	
Leakage Power:	0.604W
Dynamic Energy:	0.751nJ/line(read)
	1.53nJ/line(write)
main memory:	
Dynamic Energy:	51nJ/line

6.3.2 Evaluation result



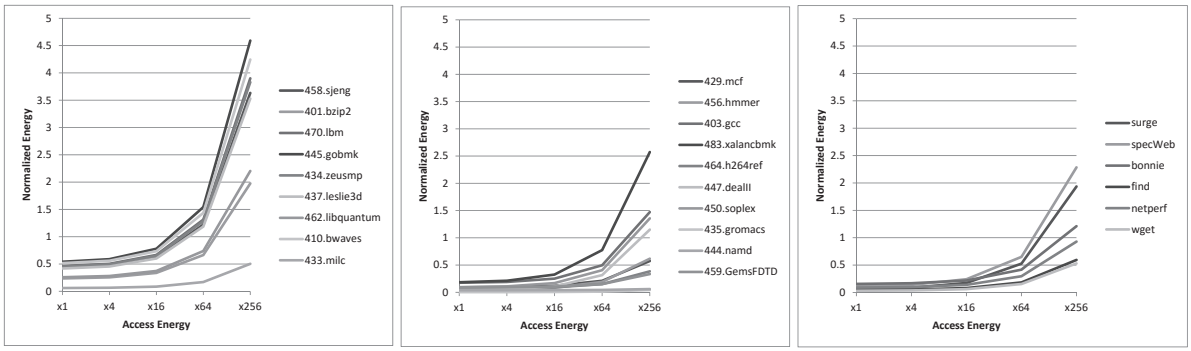
[1]memory bound [2]CPU bound [3]I/O bound

Figure 6.3: Execution time as a function of latency



[1]memory bound [2]CPU bound [3]I/O bound

Figure 6.4: Energy consumption as a function of access latency



[1]memory bound [2]CPU bound [3]I/O bound

Figure 6.5: Energy consumption as a functions of access energy

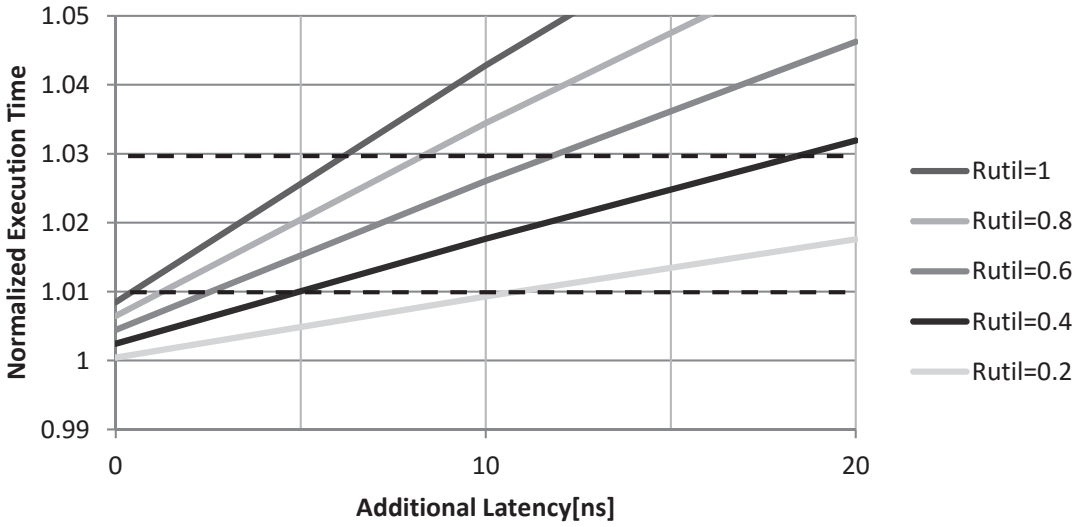


Figure 6.6: The relative execution time as a function of latency for various R_{util}

Figure 6.3-6.5 show the performance / energy evaluation results for various applications. For the evaluation, we add latency to the original STT-MRAM one shown in Table 6.1 and also multiple various values with the original access energy shown in Table 6.2. In this thesis, these original parameters are called top-parameters. In Figure 6.3, the X-axis shows the additional latency, while the Y-axis indicates the relative execution time. In Figure 6.4, the X-axis shows the additional latency, while the Y-axis indicates the relative energy consumption. In Figure 6.5, the X-axis shows the access energy, while the Y-axis describes the relative energy consumption. In the energy evaluation, the dynamic/leakage energy of the LLC and the memory access energy during executing the application are considered. The memory access energy is considered because an STT-MRAM LLC can reduce the number of memory accesses because it has larger capacity than an SRAM LLC. As shown in the figure, the impact of parameters on performance / energy is significant for memory bound applications. We utilized these results for the calculation of parameter requirement. In the later evaluation we set R_{mb} to 0.5.

Figure 6.6 shows the relationship between the total execution time and the latency of the LLC for various R_{util}. The X-axis shows the additional latency, while the Y-axis indicates

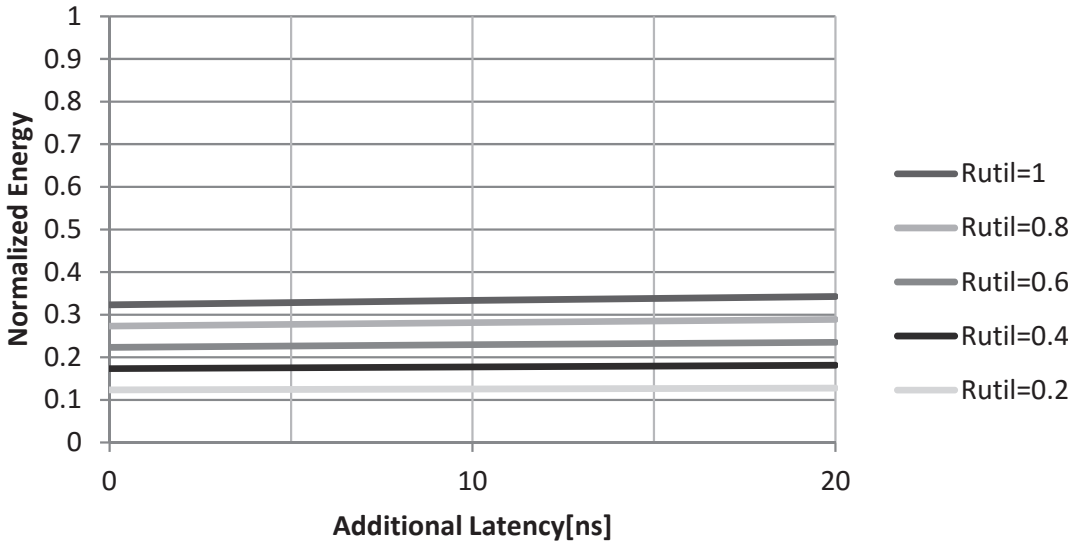


Figure 6.7: Energy consumption as a function of access latency for R_{util}

the relative execution time. The broken line shows examples of performance constraints. By obtaining the intersection point of the performance constraint line and the performance line for a certain R_{util}, we can acquire the parameter requirement of latency for R_{util}. For example, if we accept 1 % performance degradation, the STT-MRAM having 11ns longer latency than the original one is acceptable. The prior work usually evaluate the effectiveness of STT-MRAM only with high load application (R_{util}=1). Therefore, we can also know how much the parameter requirement can be relaxed by considering low load state by comparing the intersection point between $R_{util} = 1$ and the others.

Figure 6.7 shows the relationship between the energy consumption of the LLC and the access latency of the cache for various R_{util}. The X-axis shows the additional latency, while the Y-axis indicates the relative energy consumption. As shown in the figure, the impact of latency increase on the total energy consumption of the cache is quite small. This is because the incrementation of the total static energy consumption caused by performance degradation relatively causes small impact on total energy consumption. The difference of these values are brought by the frequent memory references of memory bound applications.

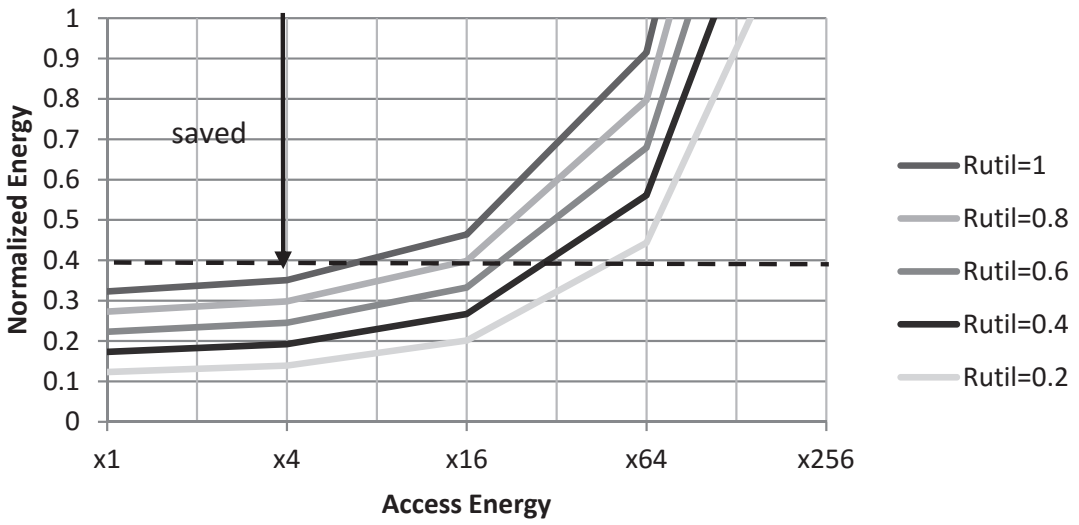


Figure 6.8: Energy consumption as a function of access energy for R_{util}

Figure 6.8 indicates the relation between the energy consumption of the cache and the access energy of it for various R_{util}. The X-axis shows the access energy, while the Y-axis indicates the relative energy consumption. The broken line shows examples of target energy reduction. In the same way as Figure 6.6, we can acquire the parameter requirement of the access energy for R_{util} by obtaining the intersection point of the target energy reduction line and the energy line for a certain R_{util}. For example, if the target is 60% energy reduction, the STT-MRAM having around 40 times higher access energy than the original one is acceptable for $R_{util} = 0.2$. We can also confirm how much the parameter requirement of access energy can be relaxed by considering low load state by comparing the intersection point between $R_{util} = 1$ and the others because previous work considers only such high load state.

Figure 6.9 shows the parameter requirement at the constraints of $(R_{inc}, R_{save}) = (0.01, 0.67)$ for various R_{util}. For $R_{util} = 1$, even the STT-MRAM cache having the top-parameter can not reach the target energy reduction, thus the top-parameter should be improved for the system. If we consider the system whose utilization rate R_{util} is from 20% to 40%, 2.5-5ns longer access latency and 16-22 times higher access energy than the top-parameters are acceptable.

Figure 6.10 shows the parameter requirement at the constraints of $(R_{inc}, R_{save}) = (0.03, 0.50)$

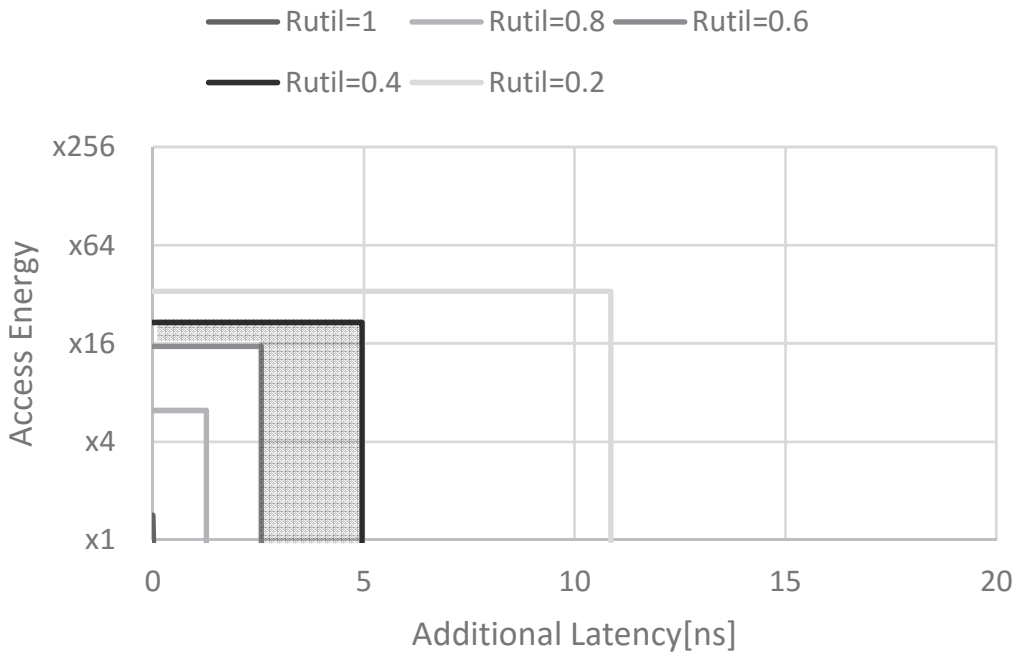


Figure 6.9: Parameter requirement at $R_{inc} = 0.01$, $R_{save} = 0.67$

for various R_{util}. The performance constraint and target energy reduction are relaxed compared with Figure 6.9. In this case, 8ns longer access latency and 20 times higher access energy than the top-parameters are acceptable. For $R_{util} = 1$, even the STT-MRAM cache having the top-parameter can not reach the target energy reduction, thus the too-parameter should be improved for the system. If we consider the system whose utilization rate R_{util} is from 20% to 40%, 12-18ns longer access latency and 32-45 times higher access energy than the top-parameters are acceptable.

From above results, it is turned out that the parameter requirement for an STT-MRAM LLC can be significantly relaxed by taking the system load state into consideration, which allows us to utilize the STT-MRAM having lower manufacturing cost. After utilizing our methodology, we can choose the best design with considering cost constraint from the designs which meet the parameter requirement.

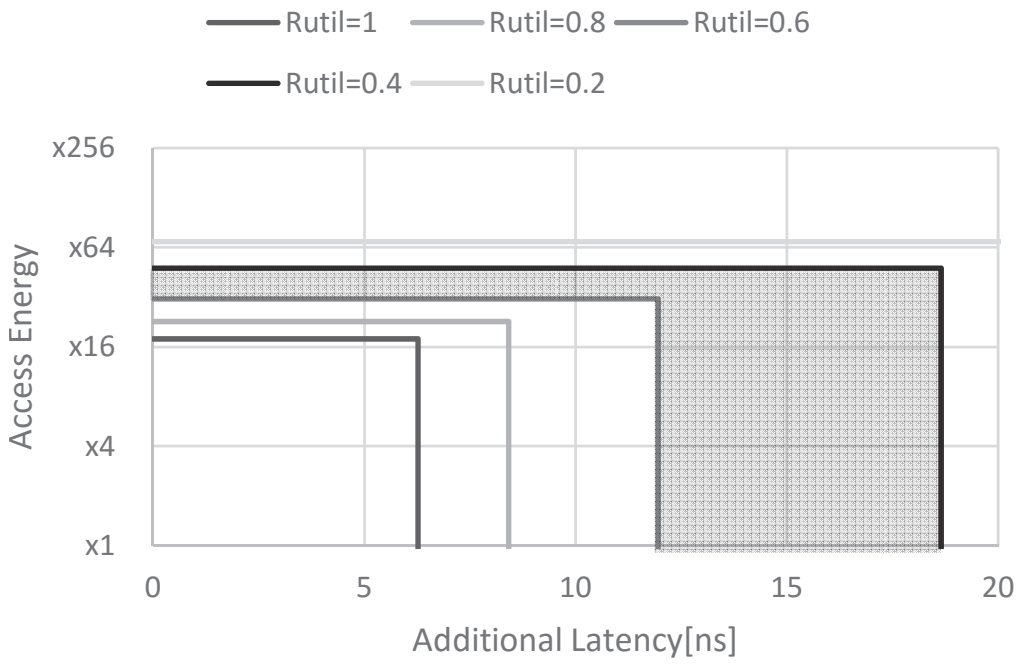


Figure 6.10: Parameter requirement at $R_{inc} = 0.03$, $R_{save} = 0.50$

Chapter 7

Related work

In this chapter, this thesis clarifies the scope of this research in the overall power reduction methodologies utilized for various components in various computing systems. Then, we explain the differences between these studies and ours.

7.1 The scope of this research

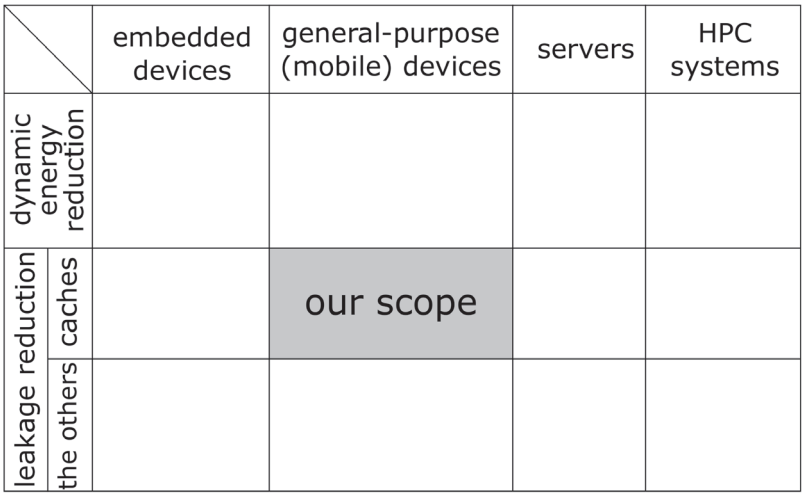


Figure 7.1: The scope of this work in the overall power reduction methodologies

Figure 7.1 shows the scope of this research in various components in various systems. The power consumption of a system comes from two different parts: dynamic and leakage. As described before, the leakage power consumption of computer systems will be larger and

larger as VLSI technology scales, thus this thesis focuses on the leakage reduction. Especially, we focus on the leakage reduction of caches in general-purpose processors utilized in mobile devices. Although our approaches are proposed for the caches of these processors, it can also be extended to other levels of the memory hierarchy or other kinds of computer systems as described later.

7.2 Dynamic energy reduction techniques for various components in various systems

Because the dynamic power of processors is also large for various systems, there have been some researches which attempt to mitigate it. To reduce the dynamic power of several components such as CPUs, GPUs and main memories, DVFS(Dynamic Voltage and Frequency Scaling) is widely utilized. For embedded systems, some DVFS algorithms have been proposed such as ones specialized for energy harvesting systems [62, 63] and temperature-aware one [64]. For servers or HPC systems, various power management techniques utilizing DVFS have also been proposed for CPUs, GPUs and main memories. In [65], an algorithm that co-ordinates the voltage and frequency levels of both processors and DRAM memories in server systems is proposed. In [66], a power management scheme which co-ordinates DVFS and task mapping in CPU-GPU heterogeneous HPC nodes is proposed. Although these techniques can significantly reduce the dynamic energy of systems, they have no capability of mitigating leakage energy.

In various memory systems, dynamic energy has been one of the critical issues. Particularly, non-volatile memory based caches have suffered from write access energy, thus, several architectural energy reduction techniques have been proposed [21, 23, 22]. Wu et al. propose a cache system that is composed of STT-MRAM and SRAM area [21]. This cache system stores frequently-written data in SRAM area to save the write-access energy in STT-MRAM area. Zhow et al. propose EWT (Early Write Termination) for saving the energy for redun-

dant writes [22]. In EWT, if a value of a written bit is equivalent to a retained value, the write operation stops at the early stage. Kim et al. propose a scheme to reduce write energy based on the difference of the amount of write current between memory cells [23]. These techniques achieve the energy saving for write accesses, however they have no capability to reduce leakage power of memory systems.

7.3 Leakage energy reduction techniques for various components except caches

Because the leakage power of processors will be more and more significant problem in the future as VLSI technology scales, there have been a several solutions for reducing the leakage for various components like processors and main memories.

In a microprocessors, functional units like ALUs and FPUs also consume large leakage power, thus there is some prior work which tries to reduce the leakage during active state by fine-grained power-gating [27, 28, 29, 30, 31]. More specifically, these work turns off the functional units when they are idle during executing CPU/memory bound applications by the compilers and operating systems based power management schemes. These techniques can be applied during active state unlike the power control schemes for SRAM-based caches because they do not bring performance/energy overheads caused by data loss. Because of such fundamental difference of performance/energy overheads between memories and logics, the required power management scheme is also different.

Because conventional DRAM based main memories consume large refresh power to avoid the data loss caused by leakage current, there are a lot of work which attempted to reduce it by use of various memory technologies from low-power DRAM to non-volatile memories like PRAM or STT-MRAM [67, 68, 55, 69, 70, 71, 72, 73]. Our methodologies can also be applied to these memories as described later even though the temporal granularity of power management technique is different.

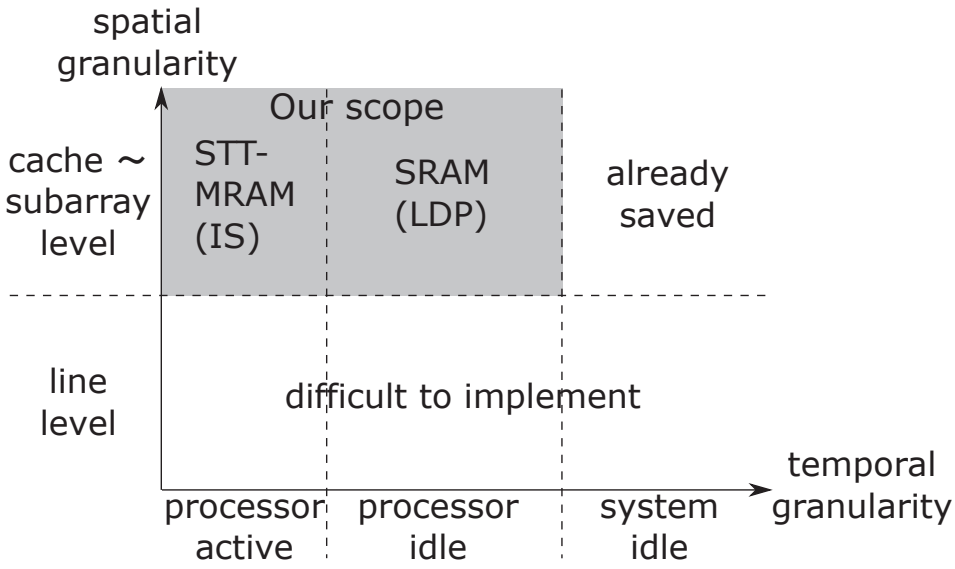


Figure 7.2: The scope of this research in the various power management granularities

7.4 Leakage energy reduction techniques for caches in general-purpose processors

Figure 7.2 shows the scope of this research in the various power management granularities for caches in general-purpose processors. A processor is turned off during system idle state, thus we do not have to consider this state for leakage reduction of caches. Therefore, we focused on the system active state (the processor active/idle state) in this thesis. Moreover, we consider the leakage reduction of the memory cells in SRAM caches and the local peripheral circuits in STT-MRAM based caches because they are the major leakage power consumers in each cache. If we consider them, coarse-grained power-gating like subarray level one is promising for the easiness of implementation, thus we only consider such coarse-grained power-gating in this thesis. Furthermore, due to the performance/energy impact of data loss, power-gating is applied only in processor idle state for SRAM caches in this thesis, while that is applied in processor active state for STT-MRAM caches. Under the assumptions, we proposed *Lost Data Prefetch* for SRAM based caches and *Immediate Sleep* for STT-MRAM based ones, then evaluated the parameter requirement of STT-MRAM based ones.

7.4.1 Fine-grained power management techniques for caches during processor active state

Since SRAM caches have suffered from leakage problem, there have been proposed several solutions, in which very fine-grained such as cache line level power management schemes during processor active state are generally considered. Cache decay and drowsy cache have been well studied, which are the leakage power reduction techniques for SRAM memory cells at the granularity of cache line [32, 74, 75, 76, 77]. However, because of the difficulty for the implementation, much coarser grained power controls like bank-level power-gating are utilized during long processor idle in conventional commercial chips [5].

7.4.2 Coarse-grained power management techniques for caches during processor active state

There exists a few studies about saving the leakage energy of peripheral circuits of SRAM caches at the granularity of entire cache level during active state [60, 78]. Because these papers focus on the leakage reduction of the peripherals in caches, they may look similar to one of our solutions. However, these papers focus on power saving of global peripheral circuits such as pre-decoders and global output drivers, in contrast to us. As described in Chapter 5, leakage power of local peripheral circuits is dominant in total power of a state-of-the-art STT-MRAM cache, thus we have to focused on them. Because the spatial granularity of power management is fundamentally different, thus effective power management algorithm is also different.

Because the leakage power of peripheral circuits will be one of the most significant problem in STT-MRAM caches in the future, we applied power-gating to them at the granularity of subarrays. Because turning off STT-MRAM caches does not cause data loss, thus we can apply it during processor active state. Noguchi et al. [24] proposed the implementation of power gating for an STT-MRAM chip which contains a subarray, which is based on our

power-gating strategy shown in [79]. However, they applied the timeout-based naive power management scheme to the chip, thus it still wastes about half of total leakage power of the subarray. Moreover, the performance degradation caused by power-gating is not considered in the paper. On the other hand, we also proposed a more sophisticated power management algorithm called *Immediate Sleep* which reduces large part of the leakage of subarrays in an STT-MRAM LLC under a certain performance constraint. This power management algorithm was implemented in a real chip as shown in [80].

7.4.3 Coarse-grained power management techniques for caches during processor idle state

In modern microprocessors, operating systems can direct LLCs to shut their banks down during processor idle state [5]. However, the sleep chances are limited since it causes serious performance/energy overheads as a result of data loss in the power gated bank. To mitigate this problem, we propose a novel data management scheme called *Lost Data Prefetch*, which is the first work that reduces the performance overhead caused by data loss.

7.4.4 Evaluation for emerging memory technology based caches

Although there are some work which evaluated the usefulness of STT-MRAM caches [81, 21], our work is the first one which attempts to clarify the parameter requirement for STT-MRAM caches in consideration of both low processor active state and processor idle one. More specifically, the evaluations were executed only in processor active state with the high load applications like SPECCPU2006 benchmark suit [39] and PARSEC benchmark suit [82] even though the modern computer systems usually stay in processor idle state because of the I/O bound low load applications [34, 35, 38]. By considering such low load state, an STT-MRAM cache whose access latency is longer and access energy is larger can be acceptable for many systems, which can save manufacturing cost significantly, though the write access energy has been regarded as the most critical problem in STT-MRAM caches.

Chapter 8

Conclusion and further discussions

8.1 Conclusion

It is essential to optimize the cache design for improving the energy-efficiency of microprocessors under performance/cost constraints in many systems. To this end, this thesis proposed two leakage energy reduction methodologies (*Lost Data Prefetch* and *Immediate Sleep*) and a framework that clarifies the parameter requirement for STT-MRAM caches for the optimization.

First, *Lost Data Prefetch* was proposed to save the performance degradation caused by data loss brought by power-gating. More specifically, the methodology attempts to reduce the performance degradation by fetching the lost data from the main memory before they are re-refereced. By doing so, it can increase the opportunities for turning off SRAM caches. The evaluation result shows that our methodology can successfully mitigate the performance degradation with negligible area/energy overheads.

Second, *Immediate Sleep* was proposed to mitigate the leakage energy impact of peripheral circuits in STT-MRAM caches. More specifically, it turns off a subarray that contains leaky local peripheral circuits immediately when the next access is non-performance-critical. Therefore, it can effectively reduce the leakage with little impact on performance. The experimental results show that our proposal can save most part of the leakage energy consumption

of an STT-MRAM cache at negligible performance degradation.

From above proposals, the energy-efficiency of both SRAM and STT-MRAM caches can be significantly improved. As the next step, to acquire the optimal cache design, this thesis proposed the framework that clarifies the parameter requirement of STT-MRAM caches depending on the system's load state. This is because the manufacturing cost of STT-MRAM caches highly depends on their parameters and thus acquiring such requirement is quite important. Moreover, this thesis also demonstrated with the framework that the STT-MRAM having the top parameters is not necessary for LLCs and thus the parameters can be greatly relaxed for most cases.

We can finally acquire the optimal design if we can clarify the cost model of STT-MRAM caches. This is going to be our future work.

8.2 Further Discussions

This section describes further discussion about this thesis. First, we clarify the applicability conditions of each methodology proposed in the thesis in 8.2.1. Then, we explain the expandability of them to other systems and other levels of memory hierarchy in 8.2.2.

8.2.1 Applicability conditions of the proposed methods

Lost Data Prefetch

Although *LDP* was evaluated for several I/O bound applications in Chapter 4, this methodology can be used for other I/O bound applications too. However, the effectiveness of the prefetch highly depends on the executed I/O bound application. More specifically, our methodology is effective only for I/O bound applications having high spacial/temporal memory access locality at the granularity of pages. This is because the algorithm utilized in the proposal tries to exploit the page level access locality. Namely, it fetches all of lost data in a page that is accessed after turning on the cache. We can confirm such page level data

reuseness by checking TLB hit rate and it is usually high for these applications.

The methodology is also scalable, that is the method is also useful for processors having much larger caches or many cores. Because the reuseness is determined by the executed application, the cache size does not affect the efficiency of the method. Furthermore, the number of cores does not affect the effectiveness because the method is used only when the load is quite low, which means only one core is mostly used during active state.

Immediate Sleep

Although *IS* was evaluated for SPECCPU2006 benchmark suite that is widely utilized for processor performance evaluation, the methodology can also be applied in any active state during executing any applications. However, the usefulness of the method highly depends on the characteristics of executed application such as intensity, regularity and locality of the memory accesses. For the applications having quite low memory access intensity, even the conventional timeout-based power management scheme is effective because subarrays are mostly idle in such case. For the applications which produce irregular memory references, the proposed method may not work well because the prediction used in *IS* attempts to exploit the regularity of subarray level accesses. The temporal locality of executed application also affects the accuracy of the interval predictor used in *IS*. Although the efficiency of *IS* depends on all of these characteristics, we proved the effectiveness in common active cases by use of the benchmark that was developed to evaluate such cases.

The effectiveness of the method also depends on the configuration such as number of cores. Because many threads can be executed concurrently, the cache access intensity of a processor having many cores will be higher. In such case, the conventional timeout-based power management scheme does not work well and a sophisticated algorithm like ours is required to save the leakage of STT-MRAM caches. Moreover, the number of cores in a processor has been increasing in the last decade thus our methodology will be more and more

important in the future.

On the other hand, the regularity of accesses in a subarray of a shared cache decreases as the number of running threads increases, which may make the accuracy of our predictor degrade. This is because a subarray can be accessed by various threads having different characteristics of accesses at the same period. However, we can easily solve this problem by use of *Cache Partitioning* [83] by which a subarray is accessed only one application. Moreover, the technique is considered as one of the most promising approaches to improve the cache hit rate in this multi-core era.

The efficiency of our proposal also depends on the number of subarrays. This is because, generally speaking, the number of accesses to a subarray decreases as the number of subarrays increases. In such case, the conventional timeout-based scheme also works well because a subarray is less frequently accessed. However, the number of subarrays will be almost constant even if the size of an LLC will increase in the future because, for an LLC, capacity and density are the most important parameters and a subarray should be large to implement large and dense caches.

The framework for the evaluation of STT-MRAM caches' parameter requirement

The performance/energy evaluations shown in Chapter 6 are useful to optimize the device parameters of STT-MRAM for general-purpose processors. This is because, to optimize the design of such a processor, we should focus on common cases, thus utilizing the average performance/energy of each application class is appropriate. In addition to that, the applications utilized in the evaluation such as SPECCPU2006 benchmark applications are also suitable for the optimization because they are widely used on such processors. On the other hand, the errors of the performance/energy models used in the evaluations may be large for several special cases which are very different from the evaluated cases. However, we do not have to consider such *rare cases* to optimize the design of a *general-purpose* processor.

The proposed framework can also be applied to other kind of processors having different configurations such as different number of cores, different richness of core resources, different number of cache levels or different cache size. Although re-evaluations are required for such processors, the evaluation result shown in Chapter 6 is useful for optimizing device parameters of STT-MRAM because the configuration utilized for it is based on a commercial processor widely used in conventional mobile devices.

As a next step, we should appropriately set the load state parameter, the utilization rate of high load applications for each system. For example, the parameter should be set lower for low-end general-purpose processors because their load state must be usually low. To do so, we have to know the actual value of the parameter in real systems from low-end to high-end systems, which is our future work.

8.2.2 Extendability of the proposed methods to other systems / levels of memory hierarchy

This section describes further discussion beyond the scope of this thesis. More specifically, although this thesis focused on optimizing caches of general-purpose processors in mobile devices, we can extend our proposals to other levels of memory hierarchy and other computer systems as shown in Figure 8.1. In 8.2.2, this thesis describe the extension to other memory hierarchy, particularly to main memory. In 8.2.2, this thesis extend our methodologies to other computer systems.

Extension to other levels of memory hierarchy

We can extend our strategies to other levels of memory hierarchy like main memories. The only differences we should consider are the temporal/spatial granularity of power/data management and the characteristics of accesses. For example, the power supply to main memory is stopped only when the operating system goes to a system idle state like *Hibernation State* in which all the memory contents are moved to persistent storage devices like the

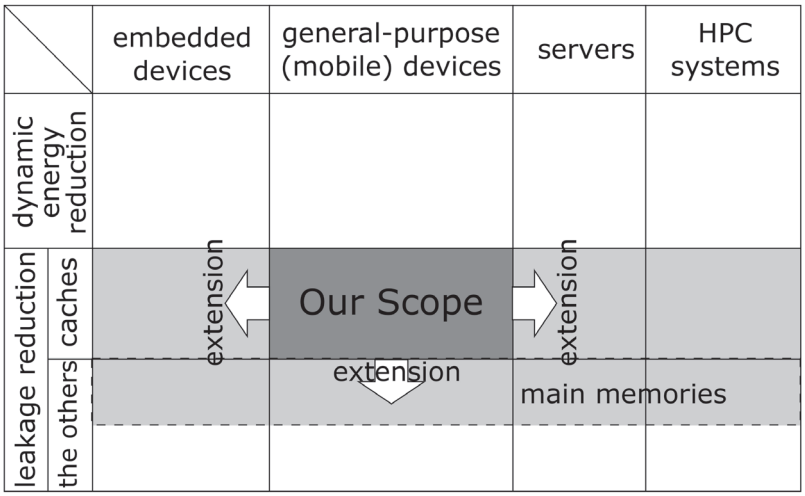


Figure 8.1: The extension of this work to other components or other systems

hard disks. Moreover, the data movement between a main memory and a storage device is usually done at the granularity of pages. In addition to that, the accesses to main memory are generally done less frequently and more burstly compared with those to caches. Although we have to optimize the algorithm because of these differences, prefetch itself can be utilized for main memories to reduce the performance/energy impact of extra data movement caused by state transitions. Moreover, applying power-gating to the peripheral circuits of various non-volatile technologies utilized in main memory will also be useful because these memories generally have to drive large current for write operations, though the power management algorithm utilized for them will need to be updated. Furthermore, we also have to choose the best technology with sophisticated power management schemes to minimize the energy of a main memory in consideration of system load state, performance/cost constraints.

Extension to the other computer systems

We can extend our schemes also to various computing systems from large scale systems HPC systems to embedded devices. Here, we explain such extensions as follows.

If we consider HPC systems, the most significant differences are the performance re-

quirement, cost constraint and the amount of resources a system has. For these systems, performance and fairness are the important factors, thus we rather have to care the worst case performance degradation for each application class unlike the evaluation done in Chapter 7. In addition to this, because these systems generally have enormous number of nodes and the cost constraint is generally loose, we can compose heterogeneous memory systems for them. More specifically, we can install various types of nodes having CPUs and main memories with various memory technologies. If we consider such systems, we should optimize the memory system of a node for a certain application class and allocate an application to the appropriate node unlike the evaluation conducted in Chapter 7. Because power or energy consumption of memory systems in HPC systems is quite high, thus such an optimization is quite important. Moreover, *LDP* is also useful for HPC systems because the usage of processor will be quite low while executing I/O intensive HPC applications. Furthermore, the energy reduction of peripheral circuits of non-volatile memories used through various levels of the memory hierarchy will also be useful for HPC systems.

Our framework can also be applied to servers and is useful for reducing their power consumption. The processors specialized in server applications generally have quite large caches [26], thus their leakage power consumption is also critical [3]. Moreover, such systems generally have quite large main memories thus their power consumption is also large. Therefore, our approach is also useful for these systems. The difference we have to consider for optimizing caches and main memories in these systems is only the workloads utilized in them. Generally, the applications utilized in such systems are I/O bound thus both *LDP* and *IS* are useful for reducing the leakage of caches. We can optimize the caches and main memories in server systems by use of the evaluation framework if we input the server workloads to it.

We can also utilize our approach for embedded systems utilized in various systems like sensor networks because the leakage energy of memory systems installed in such systems

is also critical. Only the differences we need to consider for the optimization are that the applications utilized in them are usually pre-determined and the memories are usually software controlled (scratch pad memories) due to the power constraint. Thus we can optimize the memory design for the specific applications and the algorithm utilized in *LDP* and *IS* for such applications by use of software-based approaches.

Bibliography

[1] John L. Hennessy and David A. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[2] Varghese George, Sanjeev Jahagirdar, Chao Tong, K. Smits, Satish Damaraju, Siers Scott, Ves Naydenov, Tanveer Khondker, Sanjib Sarkar, and Puneet Singh. Penryn: 45-nm Next Generation Intel®Core™2 Processor. In *2007 IEEE Asian Solid-State Circuits Conference ASSCC '07*, pages 14–17, Nov 2007.

[3] Sheng Li, Ke Chen, Jung Ho Ahn, J.B. Brockman, and N.P. Jouppi. CACTI-P: Architecture-Level Modeling for SRAM-based Structures with Advanced Leakage Reduction Techniques. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 694–701, Nov 2011.

[4] David A. Patterson and John L. Hennessy. *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th edition, 2008.

[5] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann. Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge. *IEEE, Micro*, 32(2):20–27, 2012.

[6] Ren Wang, J. Tsai, C. Maciocco, T.C. Tai, and J. Wu. Reducing Power Consumption for Mobile Platforms via Adaptive Traffic Coalescing. *IEEE Journal on Selected Areas in Communications*, 29(8):1618–1629, September 2011.

[7] H. Nakamura, T. Nakada, and S. Miwa. Normally-off computing project: Challenges and opportunities. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 1–5, Jan 2014.

[8] Zhenyu Sun, Xiuyuan Bi, Hai (Helen) Li, Weng-Fai Wong, Zhong-Liang Ong, Xiaochun Zhu, and Wenqing Wu. Multi Retention Level STT-RAM Cache Designs with a Dynamic Refresh Scheme. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 329–338, New York, NY, USA, 2011. ACM.

[9] H. Noguchi, K. Kushida, K. Ikegami, K. Abe, E. Kitagawa, S. Kashiwada, C. Kamata, A. Kawasumi, H. Hara, and S. Fujita. A 250-MHz 256b-I/O 1-Mb STT-MRAM with Advanced Perpendicular MTJ Based Dual Cell for Nonvolatile Magnetic Caches to Reduce Active Power of Processors. In *2013 Symposium on VLSI Technology (VLSIT)*, pages C108–C109, June 2013.

[10] GiorgioLuigi Valentini, Walter Lassonde, SameeUllah Khan, Nasro Min-Allah, SajjadA. Madani, Juan Li, Limin Zhang, Lizhe Wang, Nasir Ghani, Joanna Kolodziej, Hongxiang Li, AlbertY. Zomaya, Cheng-Zhong Xu, Pavan Balaji, Abhinav Vishnu, Fredric Pinel, JohnatanE. Pecero, Dzmitry Kliazovich, and Pascal Bouvry. An Overview of Energy Efficiency Techniques in Cluster Computing Systems. *Cluster Computing*, 16(1):3–15, 2013.

[11] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power Provisioning for a Warehouse-sized Computer. *SIGARCH Comput. Archit. News*, 35(2):13–23, June 2007.

[12] Grace Metri, Abhishek Agrawal, and Sherine Abdelhak. Power Profiling with Real-World Workloads: A Case Study with MobileMark 2012. In *2012 International Conference on Energy Aware Computing*, pages 1–4, Dec 2012.

[13] T. Simunic, L. Benini, and G. De Micheli. Energy-Efficient Design of Battery-Powered Embedded Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1):15–28, Feb 2001.

[14] Eishi Arima, Toshiya Komoda, Takashi Nakada, Shinobu Miwa, and Hiroshi Nakamura. Lost Data Prefetching to Reduce Performance Degradation Caused by Powering off Caches. *IPSJ Transaction of Advanced Computing Systems*, 6(3):629–647, 2013.

[15] Eishi Arima, Toshiya Komoda, Takashi Nakada, Shinobu Miwa, Hiroki Noguchi, Kumiko Nomura, Keiko Abe, Shinobu Fujita, and Hiroshi Nakamura. Analyzing Requirements Specification of STT-MRAM Last Level Cache Considering Low CPU Loads. *IEICE Transactions*, J97-A(10):629–647, 2014.

[16] N.S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan. Leakage Current: Moore's Law Meets Static Power. *Computer*, 36(12):68–75, Dec 2003.

[17] V. De. Energy Efficient Computing in Nanoscale CMOS: Challenges and Opportunities. In *2014 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 121–124, Nov 2014.

[18] M. Guillorn, J. Chang, A. Bryant, N. Fuller, O. Dokumaci, X. Wang, J. Newbury, K. Babich, J. Ott, B. Haran, R. Yu, C. Lavoie, D. Klaus, Y. Zhang, E. Sikorski, W. Graham, B. To, M. Lofaro, J. Tornello, D. Koli, B. Yang, A. Pyzyna, D. Neumeyer, M. Khater, A. Yagishita, H. Kawasaki, and W. Haensch. FinFET Performance Advan-

tage at 22nm: An AC Perspective. In *2008 Symposium on VLSI Technology*, pages 12–13, June 2008.

[19] J.-A. Carballo and A. B. Kahng. ITRS Chapters:System Drivers. In *Future Fab International*, 2011.

[20] Dmytro Apalkov, Alexey Khvalkovskiy, Steven Watts, Vladimir Nikitin, Xueti Tang, Daniel Lottis, Kiseok Moon, Xiao Luo, Eugene Chen, Adrian Ong, Alexander Driskill-Smith, and Mohamad Krounbi. Spin-Transfer Torque Magnetic Random Access Memory (STT-MRAM). *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 9(2):13:1–13:35, 2013.

[21] Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, and Yuan Xie. Hybrid Cache Architecture with Disparate Memory Technologies. In *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA)*, pages 34–45, June 2009.

[22] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. Energy Reduction for STT-RAM Using Early Write Termination. In *2009 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 264–268, Nov 2009.

[23] Yusung Kim, Sumeet Kumar Gupta, Sang Phill Park, Georgios Panagopoulos, and Kaushik Roy. Write-optimized Reliable Design of STT MRAM. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 3–8, 2012.

[24] H. Noguchi, K. Ikegami, K. Kushida, K. Abe, S. Itai, S. Takaya, N. Shimomura, J. Ito, A. Kawasumi, H. Hara, and S. Fujita. A 3.3ns-Access-Time 71.2μW/MHz 1Mb Embedded STT-MRAM Using Physically Eliminated Read-Disturb Scheme and Normally-Off

Memory Architecture. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 136–137, 2015.

[25] Eishi Arima, Hiroki Noguchi, Takashi Nakada, Shinobu Miwa, Susumu Takeda, Shinobu Fujita, and Hiroshi Nakamura. Immediate Sleep: Reducing Energy Impact of Peripheral Circuits in STT-MRAM Caches. In *33rd International Conference on Computer Design (ICCD)*, pages 149–156, Oct 2015.

[26] Jeffrey D. Gilbert, Stephen H. Hunt, Daniel Gunadi, and Ganapat i Srinivas. TULSA, A Dual P4 Core Large Shared Cache Intel Xeon Processor for the MP Server . In *Hot Chips*, 2006.

[27] Toshiya Komoda, Hiroshi Sasaki, Masaaki Kondo, and Hiroshi Nakamura. Compiler Directed Fine Grain Power Gating for Leakage Power Reduction in Microprocessor Functional Units. In *7th Workshop on Optimizations for DSP and Embedded Systems*, pages 39–48, 2009.

[28] N. Seki, Lei Zhao, J. Kei, D. Ikebuchi, Yu. Kojima, Yohei Hasegawa, H. Amano, T. Kashima, S. Takeda, T. Shirai, M. Nakata, K. Usami, T. Sunata, J. Kanai, M. Namiki, M. Kondo, and H. Nakamura. A Fine-Grain Dynamic Sleep Control Scheme in MIPS R3000. In *IEEE International Conference on Computer Design (ICCD)*, pages 612–617, Oct 2008.

[29] D. Ikebuchi, N. Seki, Y. Kojima, M. Kamata, L. Zhao, H. Amano, T. Shirai, S. Koyama, T. Hashida, Y. Umahashi, H. Masuda, K. Usami, S. Takeda, H. Nakamura, M. Namiki, and M. Kondo. Geyser-1: A MIPS R3000 CPU Core with Fine-Grained Run-Time Power Gating. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 369–370, Jan 2010.

[30] Aviral Shrivastava, Deepa Kannan, Sarvesh Bhardwaj, and Sarma Vrudhula. Reducing Functional Unit Power Consumption and Its Variation Using Leakage Sensors. *IEEE Trans. Very Large Scale Integr. Syst.*, 18(6):988–997, June 2010.

[31] Masaaki Kondo, Hiroaki Kobyashi, Ryuichi Sakamoto, Motoki Wada, Jun Tsukamoto, Mitaro Namiki, Weihan Wang, Hideharu Amano, Kensaku Matsunaga, Masaru Kudo, Kimiyoshi Usami, Toshiya Komoda, and Hiroshi Nakamura. Design and Evaluation of Fine-grained Power-gating for Embedded Microprocessors. In *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*, DATE '14, pages 145:1–145:6, 3001 Leuven, Belgium, Belgium, 2014. European Design and Automation Association.

[32] Stefanos Kaxiras, Zhigang Hu, and Margaret Martonosi. Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power. In *Proceedings of the 28th Annual International Symposium on Computer Architecture (ISCA)*, pages 240–251, New York, NY, USA, 2001. ACM.

[33] Compaq Computer Corporation and Revision B. Advanced configuration and power interface specification, 2000.

[34] David Meisner, Brian T. Gold, and Thomas F. Wenisch. Powernap: eliminating server idle power. *SIGPLAN Not.*, 44(3):205–216, March 2009.

[35] David Meisner, Christopher M. Sadler, Luiz André Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. Power management of online data-intensive services. *SIGARCH Comput. Archit. News*, 39(3):319–330, June 2011.

[36] M. Arora, S. Manne, I. Paul, N. Jayasena, and D.M. Tullsen. Understanding Idle Behavior and Power Gating Mechanisms in the Context of Modern Benchmarks on CPU-GPU

Integrated Systems. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 366–377, Feb 2015.

[37] W.L. Bircher and L.K. John. Core-Level Activity Prediction for Multicore Power Management. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(3):218–227, Sept 2011.

[38] Grace Metri, Abhishek Agrawal, and Sherine Abdelhak. Power Profiling with Real-World Workloads: A Case Study with MobileMark 2012. In *2012 International Conference on Energy Aware Computing*, pages 1–4, 2012.

[39] John L. Henning. SPEC CPU2006 Benchmark Descriptions. *SIGARCH Computer Architure News*, 34(4):1–17, September 2006.

[40] T. Ohsawa, H. Koike, S. Miura, H. Honjo, K. Tokutome, S. Ikeda, T. Hanyu, H. Ohno, and T. Endoh. 1Mb 4T-2MTJ Nonvolatile STT-RAM for Embedded Memories Using 32b Fine-Grained Power Gating Technique with 1.0ns/200ps Wake-up/Power-off Times. In *2012 Symposium on VLSI Circuits (VLSIC)*, pages 46–47, June 2012.

[41] R. Nebashi, N. Sakimura, H. Honjo, S. Saito, Y. Ito, S. Miura, Y. Kato, K. Mori, Y. Ozaki, Y. Kobayashi, N. Ohshima, K. Kinoshita, T. Suzuki, K. Nagahara, N. Ishiwata, K. Suemitsu, S. Fukami, H. Hada, T. Sugibayashi, and N. Kasai. A 90nm 12ns 32Mb 2T1MTJ MRAM. In *2009 IEEE International Solid-State Circuits Conference(ISSCC)*, pages 462–463,463a, Feb 2009.

[42] K. Tsuchida, T. Inaba, K. Fujita, Y. Ueda, T. Shimizu, Y. Asao, T. Kajiyama, M. Iwayama, K. Sugiura, S. Ikegawa, T. Kishi, T. Kai, M. Amano, N. Shimomura, H. Yoda, and Y. Watanabe. A 64Mb MRAM with Clamped-Reference and Adequate-Reference Schemes. In *2010 IEEE International Solid-State Circuits Conference(ISSCC)*, pages 258–259, Feb 2010.

[43] Peter J. Denning. The locality principle. *Communications ACM*, 48(7):19–24, July 2005.

[44] Tien-Fu Chen and Jean-Loup Baer. Effective hardware-based data prefetching for high-performance processors. *IEEE Transactions on Computers*, 44(5):609 –623, may 1995.

[45] Zhenlin Wang, Doug Burger, Kathryn S. McKinley, Steven K. Reinhardt, and Charles C. Weems. Guided region prefetching: a cooperative hardware/software approach. *SIGARCH Computer Architure News*, 31:388–398, May 2003.

[46] Kyle J. Nesbit and James E. Smith. Data cache prefetching using a global history buffer. In *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, HPCA '04, pages 96–, Washington, DC, USA, 2004. IEEE Computer Society.

[47] Hanyu Cui and S. Sair. Extending data prefetching to cope with context switch misses. In *IEEE International Conference on Computer Design (ICCD)*, pages 260–267, oct. 2009.

[48] D. Daly and H.W. Cain. Cache restoration for highly partitioned virtualized systems. In *IEEE 18th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1 –10, feb. 2012.

[49] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The Gem5 Simulator. *SIGARCH Computer Architure News*, 39(2):1–7, August 2011.

[50] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P Jouppi. CACTI 6.0: A Tool to Understand Large Caches, 2009.

[51] Bonnie++. http://www.coker.com.au/bonnie++/.

[52] Netperf. http://www.netperf.org/netperf/.

[53] S. Khan, A.R. Alameldeen, C. Wilkerson, O. Mutlu, and D.A. Jimenezz. Improving Cache Performance Using Read-Write Partitioning. In *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 452–463, Feb 2014.

[54] Suock Chung, K.-M. Rho, S.-D. Kim, H.-J. Suh, D.-J. Kim, H.J. Kim, S.H. Lee, J.-H. Park, H.-M. Hwang, S.-M. Hwang, J.-Y. Lee, Y.-B. An, J.-U. Yi, Y.-H. Seo, D.-H. Jung, M.-S. Lee, S.-H. Cho, J.-N. Kim, G.-J. Park, Gyuan Jin, A. Driskill-Smith, V. Nikitin, A. Ong, X. Tang, Yongki Kim, J.-S. Rho, S.-K. Park, S.-W. Chung, J.-G. Jeong, and S.J. Hong. Fully Integrated 54nm STT-RAM with the Smallest Bit Cell Dimension for High Density Memory Application. In *IEEE International Electron Devices Meeting (IEDM)*, pages 12.7.1–12.7.4, 2010.

[55] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu. Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 256–267. IEEE Computer Society, 2013.

[56] H. Noguchi, K. Kushida, K. Ikegami, K. Abe, E. Kitagawa, S. Kashiwada, C. Kamata, A. Kawasumi, H. Hara, and S. Fujita. A 250-MHz 256b-I/O 1-Mb STT-MRAM with Advanced Perpendicular MTJ Based Dual Cell for Nonvolatile Magnetic Caches to Reduce Active Power of Processors. In *2013 Symposium on VLSI Technology (VLSIT)*, pages 108–109, June 2013.

[57] Dongsoo Lee, Sumeet Kumar Gupta, and Kaushik Roy. High-Performance Low-Energy STT MRAM Based on Balanced Write Scheme. In *Proceedings of the 2012 ACM/IEEE*

International Symposium on Low Power Electronics and Design (ISLPED), pages 9–14, 2012.

[58] Tse-Yu Yeh and Yale N. Patt. Two-level Adaptive Training Branch Prediction. In *Proceedings of the 24th Annual International Symposium on Microarchitecture (MICRO)*, pages 51–61. ACM, 1991.

[59] M.K. Qureshi and G.H. Loh. Fundamental Latency Trade-off in Architecting DRAM Caches: Outperforming Impractical SRAM-Tags with a Simple and Practical Design. In *45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 235–246, Dec 2012.

[60] H. Homayoun and A. Veidenbaum. Reducing Leakage Power in Peripheral Circuits of L2 caches. In *25th International Conference on Computer Design (ICCD)*, pages 230–237, Oct 2007.

[61] T. Kishi, H. Yoda, T. Kai, T. Nagase, E. Kitagawa, M. Yoshikawa, K. Nishiyama, T. Daibou, M. Nagamine, M. Amano, S. Takahashi, M. Nakayama, N. Shimomura, H. Aikawa, S. Ikegawa, S. Yuasa, K. Yakushiji, H. Kubota, A. Fukushima, M. Oogane, T. Miyazaki, and K. Ando. Lower-current and fast switching of a perpendicular tmr for high speed and high density spin-transfer-torque mram. In *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pages 1 –4, dec. 2008.

[62] Shaobo Liu, Qing Wu, and Qinru Qiu. An Adaptive Scheduling and Voltage/Frequency Selection Algorithm for Real-Time Energy Harvesting Systems. In *46th ACM/IEEE Design Automation Conference (DAC '09)*, pages 782–787, July 2009.

[63] A. Ravinagarajan, D. Dondi, and T Simunic Rosing. DVFS Based Task Scheduling in a Harvesting WSN for Structural Health Monitoring. In *Proceedings of the Conference*

on Design, Automation and Test in Europe, DATE '10, pages 1518–1523, 3001 Leuven, Belgium, Belgium, 2010. European Design and Automation Association.

[64] Yongpan Liu, Huazhong Yang, R.P. Dick, Hui Wang, and Li Shang. Thermal vs Energy Optimization for DVFS-Enabled Processors in Embedded Systems. In *8th International Symposium on Quality Electronic Design (ISQED '07)*, pages 204–209, March 2007.

[65] Qingyuan Deng, D. Meisner, A. Bhattacharjee, T.F. Wenisch, and R. Bianchini. CoScale: Coordinating CPU and Memory System DVFS in Server Systems. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 143–154, Dec 2012.

[66] T. Komoda, S. Hayashi, T. Nakada, S. Miwa, and H. Nakamura. Power Capping of CPU-GPU Heterogeneous Systems through Coordinating DVFS and Task Mapping. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 349–356, Oct 2013.

[67] Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu. RAIDR: Retention-Aware Intelligent DRAM Refresh. In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society.

[68] Ishwar Bhati, Zeshan Chishti, Shih-Lien Lu, and Bruce Jacob. Flexible Auto-refresh: Enabling Scalable and Energy-efficient DRAM Refresh Reductions. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 235–246, New York, NY, USA, 2015. ACM.

[69] Wangyuan Zhang and Tao Li. Exploring Phase Change Memory and 3D Die-Stacking for Power/Thermal Friendly, Fast and Durable Memory Architectures. In *18th Interna-*

tional Conference on Parallel Architectures and Compilation Techniques (PACT), pages 101–112, Sept 2009.

[70] Sangyeun Cho and Hyunjin Lee. Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 347–357, New York, NY, USA, 2009. ACM.

[71] G. Dhiman, R. Ayoub, and T. Rosing. PDRAM: A Hybrid PRAM and DRAM Main Memory System. In *46th ACM/IEEE Design Automation Conference (DAC '09)*, pages 664–669, July 2009.

[72] HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael Harding, and Onur Mutlu. Row Buffer Locality Aware Caching Policies for Hybrid Memories. In *Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD 2012)*, ICCD '12, pages 337–344, Washington, DC, USA, 2012. IEEE Computer Society.

[73] Meng-Fan Chang, Pi-Feng Chiu, and Shyh-Shyuan Sheu. Circuit Design Challenges in Embedded Memory and Resistive RAM (RRAM) for Mobile SoC and 3D-IC. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, ASPDAC '11, pages 197–203, Piscataway, NJ, USA, 2011. IEEE Press.

[74] Karthik Sankaranarayanan and Kevin Skadron. Profile-based Adaptation for Cache Decay. *ACM Transactions on Architecture and Code Optimization (TACO)*, 1(3):305–322, 2004.

[75] Matteo Monchiero, Ramon Canal, and Antonio Gonzalez. Using coherence information and decay techniques to optimize l2 cache leakage in cmps. In *Proceedings of the 2009*

International Conference on Parallel Processing, ICPP '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[76] Krisztián Flautner, Nam Sung Kim, Steve Martin, David Blaauw, and Trevor Mudge. Drowsy Caches: Simple Techniques for Reducing Leakage Power. In *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA)*, pages 148–157, 2002.

[77] Salvador Petit, Julio Sahuquillo, Jose M. Such, and David Kaeli. Exploiting temporal locality in drowsy cache policies. In *Proceedings of the 2nd conference on Computing frontiers*, CF '05, pages 371–377, New York, NY, USA, 2005. ACM.

[78] H. Homayoun, A. Veidenbaum, and J.-L. Gaudiot. Adaptive Techniques for Leakage Power Management in L2 Cache Peripheral Circuits. In *26th International Conference on Computer Design (ICCD)*, pages 563–569, Oct 2008.

[79] Eishi Arima, Hiroki Noguchi, Takashi Nakada, Shinobu Miwa, Susumu Takeda, Shinobu Fujita, and Hiroshi Nakamura. Fine-Grain Power-Gating on STT-MRAM Peripheral Circuits with Locality-aware Access Control. In *The Memory Forum (in conjunction with the 41st International Symposium on Computer Architecture)*, June 2014.

[80] Hiroki Noguchi, Kazutaka Ikegami, Satoshi Takaya, Eishi Arima, Atsushi Kawasumi, Hiroyuki Hara, Keiko Abe, Naoharu Shimomura, Junichi Ito, Shinobu Fujita, Takashi Nakada, and Hiroshi Nakamura. 4Mb STT-MRAM-based Cache with Memory-Access-aware Power Optimization and Novel Write-Verified-Write / Read-Modified-Write Scheme. In *2016 IEEE International Conference of Solid-State Circuits (ISSCC)*, pages 132–133, Feb. 2016.

[81] Guangyu Sun, Xiangyu Dong, Yuan Xie, Jian Li, and Yiran Chen. A Novel Architecture of the 3D Stacked MRAM L2 Cache for CMPs. In *IEEE 15th International Symposium on High Performance Computer Architecture (HPCA)*, pages 239–249, 2009.

[82] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, PACT '08, pages 72–81, New York, NY, USA, 2008. ACM.

[83] Moinuddin K. Qureshi and Yale N. Patt. Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 423–432, Washington, DC, USA, 2006. IEEE Computer Society.

Publication list by the Author

Journal Publications

[J-1] Eishi Arima, Toshiya Komoda, Takashi Nakada, Shinobu Miwa, Hiroki Noguchi, Kumiko Nomura, Keiko Abe, Shinobu Fujita, Hiroshi Nakamura, Analyzing Requirements Specification of STT-MRAM Last Level Cache Considering Low CPU Loads (in Japanese), *IEICE Transactions*, Vol.J97-A, No.10, pp.629-647, 2014.

[J-2] Eishi Arima, Toshiya Komda, Takashi Nakada, Shinobu Miwa, Hiroshi Nakamura, Lost Data Prefetching to Reduce Performance Degradation Caused by Powering off Caches (in Japanese), *IPSJ Transaction of Advanced Computing Systems*, Vol.6, No.3, pp.118-130, 2013.

International Conference and Workshop Publications

[I-1] Hiroki Noguchi, Kazutaka Ikegami, Satoshi Takaya, Eishi Arima, Atsushi Kawasumi, Hiroyuki Hara, Keiko Abe, Naoharu Shimomura, Junichi Ito, Shinobu Fujita, Takashi Nakada, Hiroshi Nakamura, 4Mb STT-MRAM-based Cache with Memory-Access-aware Power Optimization and Novel Write-Verified-Write / Read-Modified-Write Scheme, In *Proceedings of 2016 IEEE International Conference of Solid-State Circuits (ISSCC)*, pp.132–133, Feb., 2016

[I-2] Susumu Takeda, Hiroki Noguchi, Kumiko Nomura, Shinobu Fujita, Shinobu Miwa, Eishi Arima, Takashi Nakada, Hiroshi Nakamura, Low-power cache memory with

state-of-the-art STT-MRAM for high-performance processors, In *Proceedings of the 12th International SoC Design Conference (ISOCC)*, pp.153–154, Nov., 2015

[I-3] Eishi Arima, Hiroki Noguchi, Takashi Nakada, Shinobu Miwa, Susumu Takeda, Shinobu Fujita, Hiroshi Nakamura, Immediate Sleep: Reducing Energy Impact of Peripheral Circuits in STT-MRAM Caches, In *Proceedings of the 33rd IEEE International Conference on Computer Design (ICCD)*, pp.149–156, Oct. 2015

[I-4] Eishi Arima, Hiroki Noguchi, Takashi Nakada, Shinobu Miwa, Susumu Takeda, Shinobu Fujita, Hiroshi Nakamura, Subarray Level Power-Gating in STT-MRAM Caches to Mitigate Energy Impact of Peripheral Circuits, *52nd Design Automation Conference (DAC)*, Work-in-Progress Session, June, 2015 (poster)

[I-5] Eishi Arima, Hiroki Noguchi, Takashi Nakada, Shinobu Miwa, Susumu Takeda, Shinobu Fujita, Hiroshi Nakamura, Fine-Grain Power-Gating on STT-MRAM Peripheral Circuits with Locality-aware Access Control, *The Memory Forum (in conjunction with the 41st International Symposium on Computer Architecture)*, June, 2014

[I-6] Hiroki Noguchi, Kumiko Nomura, Keiko Abe, Shinobu Fujita, Eishi Arima, Kyundong Kim, Takashi Nakada, Shinobu Miwa, Hiroshi Nakamura, D-MRAM Cache: Enhancing Energy Efficiency with 3T-1MTJ DRAM/MRAM Hybrid Memory, In *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp.1813–1818, Mar., 2013

Domestic Conference and Technical Society Meeting Publications

[D-1] Eishi Arima, Shinobu Miwa, Takashi Nakada, Hiroshi Nakamura, A Data Management Scheme for Large LLCs to Mitigate Performance Degradation Caused by TLB Misses (in Japanese), *IPSJ SIG Notes*, 2014-ARC-214, No.8, pp.1-6, 2015.

[D-2] Eishi Arima, Hiroki Noguchi, Takashi Nakada, Shinobu Miwa, Susumu Takeda, Shinobu Fujita, Hiroshi Nakamura, A Locality-Aware Power Management Scheme for STT-MRAM Peripheral Circuits (in Japanese), *IPSJ SIG Notes*, 2014-ARC-211, No.11, pp.1-6, 2014.

[D-3] Eishi Arima, Toshiya Komoda, Shinobu Miwa, Hiroshi Nakamura, An Implementation of Lost Data Prefetching (in Japanese), *IPSJ SIG Notes*, Vol. 2012-ARC-201, No. 15, pp. 1-7, jul 2012.

[D-4] Eishi Arima, Toshiya Komoda, Shinobu Miwa, Hiroki Noguchi, Kumiko Nomura, Keiko Abe, Shinobu Fujita, Hiroshi Nakamura, Comparison among Leakage Reduction Techniques for Last Level Caches under OS Power Management (in Japanese), In *Proceeding of the 25th Workshop on Circuits and Systems*, Vol. 25, pp. 402-407, 2012.

[D-5] Eishi Arima, Toshiya Komoda, Shinobu Miwa, Hiroshi Nakamura, A Technique to Mitigate the Performance Impact of Turning off Caches during Idle State (in Japanese), *IPSJ SIG Notes*, Vol. 2012-ARC-198, No. 2, pp. 1-6, 2012.