공학박사 학위논문

Architecting Main Memory Systems to Achieve Low Access Latency

짧은 접근 지연시간을 갖는 메인 메모리 시스템 설계

2016 년 8 월

서울대학교 융합과학기술대학원

융합과학부 지능형융합시스템전공

손 영 훈

Architecting Main Memory Systems to Achieve Low Access Latency

짧은 접근 지연시간을 갖는 메인 메모리 시스템 설계

지도 교수 안 정 호

이 논문을 공학박사 학위논문으로 제출함
2016 년 7 월

서울대학교 융합과학기술대학원
융합과학부 지능형융합시스템전공
손 영 훈

손영훈의 공학박사 학위논문을 인준함
2016 년 7 월

위 원 장 _____ 곽 노 준 _____ (인)

부위원장 _____ 안 정 호 _____ (인)

위 원 _____ 박 재 흥 _____ (인)

위 원 _____ 김 동 준 _____ (인)

위 원 _____ 이 재 욱 _____ (인)

Abstract

Architecting Main Memory Systems to Achieve Low Access Latency

Young Hoon Son

Intelligence Systems

Department of Transdisciplinary Studies

The Graduate School

Seoul National University

DRAM has been a de facto standard for main memory, and advances in process technology have led to a rapid increase in its capacity and bandwidth. In contrast, its random access latency has remained relatively stagnant, as it is still around 100 CPU clock cycles. Modern computer systems rely on caches or other latency tolerance techniques to lower the average access latency. However, not all applications have ample parallelism or locality that would help hide or reduce the latency. Moreover, applications' demands for memory space continue to grow, while the capacity gap between last-level caches and main memory is unlikely to shrink. Consequently, reducing the main-memory latency is important for application performance. Unfortunately, previous proposals have not

adequately addressed this problem, as they have focused only on improving the bandwidth and capacity or reduced the latency at the cost of significant area overhead.

Modern DRAM devices for the main memory are structured to have multiple banks to satisfy ever-increasing throughput, energy-efficiency, and capacity demands. Due to tight cost constraints, only one row can be buffered (opened) per bank and actively service requests at a time, while the row must be deactivated (closed) before a new row is stored into the row buffers. Hasty deactivation unnecessarily re-opens rows for otherwise row-buffer hits while hindsight accompanies the deactivation process on the critical path of accessing data for row-buffer misses. The time to (de)activate a row is comparable to the time to read an open row while applications are often sensitive to DRAM latency. Hence, it is critical to make the right decision on when to close a row. However, the increasing number of banks per DRAM device over generations reduces the number of requests per bank. This forces a memory controller to frequently predict when to close a row due to a lack of information on future requests, while the dynamic nature of memory access patterns limits the prediction accuracy.

In this thesis, we propose three novel DRAM bank organizations to reduce the average main-memory access latency. First, we introduce a novel DRAM bank organization with center high aspect-ratio (i.e., low-latency) mats called CHARM. Experiments on a simulated chip-multiprocessor system show that CHARM improves both the instructions per cycle and system-wide energy-delay

product up to 21% and 32%, respectively, with only a 3% increase in die area. Second, we propose SALAD, a new DRAM device architecture that provides symmetric access latency with asymmetric DRAM bank organizations. SALAD leverages the asymmetric bank structure of CHARM in an opposite way. SALAD applies high aspect-ratio mats only to remote banks to offset the difference in data transfer time, thus providing uniformly low access time (tAC) over the whole device. Our evaluation demonstrates that SALAD improves the IPC by 13% (10%) without any software modifications, while incurring only 6% (3%) area overhead. Finally, we propose a novel DRAM microarchitecture that can eliminate the need for any prediction. By decoupling the bitlines from the row buffers using isolation transistors, the bitlines can be precharged right after a row becomes activated. Therefore, only the sense amplifiers need to be precharged for a miss in most cases. Also, we show that this row-buffer decoupling enables internal DRAM μ-operations to be separated and recombined, which can be exploited by memory controllers to make the main memory system more energy efficient. Our experiments demonstrate that row-buffer decoupling improves the geometric mean of the instructions per cycle and MIPS2/W by 14% and 29%, respectively, for memory-intensive SPEC CPU2006 applications.

Keywords : memory system, DRAM, CHARM, SALAD, row-buffer decoupling, access latency, area overhead

Student Numbers : 2012-30700

Contents

List of Figures

List of Tables

Chapter 1

Introduction

DRAM has been a de facto standard for main memory for decades thanks to its high density and performance. DRAM has more than ten times higher storage density than SRAM and is orders of magnitude faster than NAND flash devices. With continued technology scaling, DRAM devices have evolved to exploit these smaller and faster transistors to increase mainly their capacity and bandwidth under tight cost constraints. To increase capacity, the DRAM cell size has been scaled down aggressively, and more cells share control and datapath wires. Meanwhile, DRAM arrays are divided into many subarrays, or mats, not to slow down those wires, and more bits are prefetched to improve bandwidth. However, the latency of DRAM devices, especially their random access time, has been reduced much more slowly. It is still around 50ns, which translates to approximately 100 CPU clock cycles (Figure 1.1).

This Section is based on [1, 2, 3]. – © [2013, 2014, 2016] ACM and IEEE Reprinted, with permissions, from ISCA’ 13, ISCA’ 14 and CAL' 16.

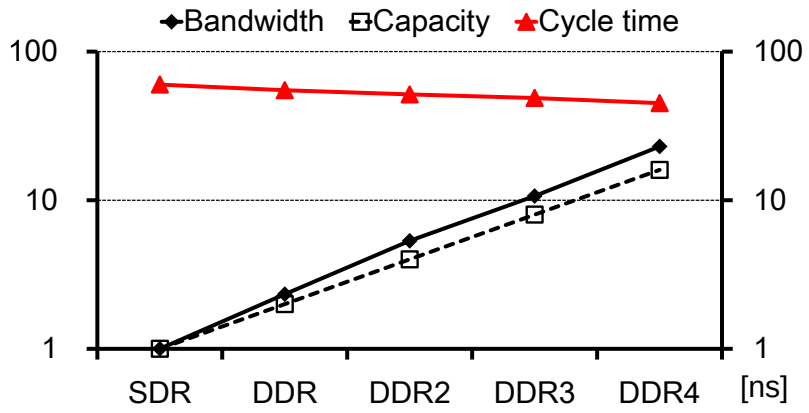


Figure 1.1: The capacity and bandwidth of DRAM devices have increased rapidly over time, but their latency has decreased much more slowly [41].

Modern computer systems try to address this memory wall problem [51] with either latency tolerance techniques, such as out-of-order speculative execution [37], vector [36], stream [18], and massive multithreading [6, 28], or multilevel caches that lower the average memory access time. However, not all applications can be made insensitive to the main-memory latency because they often have insufficient parallelism or locality. Also, the memory footprints of popular applications [15] keep growing, and emerging applications, such as in-memory databases [59] and genome assemblies [61], demand even higher capacity. Besides, the gap in size between last-level caches and main memory has not been narrowed. As a result, lowering the main-memory latency would benefit many such applications. Figure 1.2 shows the relative instructions per cycle (IPC) of SPEC CPU2006 applications [15]

when we lower the 28ns access time and the 48ns cycle time of DRAM by 25%, 50%, and 75%. The degree of IPC improvement varies across applications, but an IPC improvement of more than 50% is observed for memory-intensive applications when we cut the DRAM latency by half without changing the bandwidth.

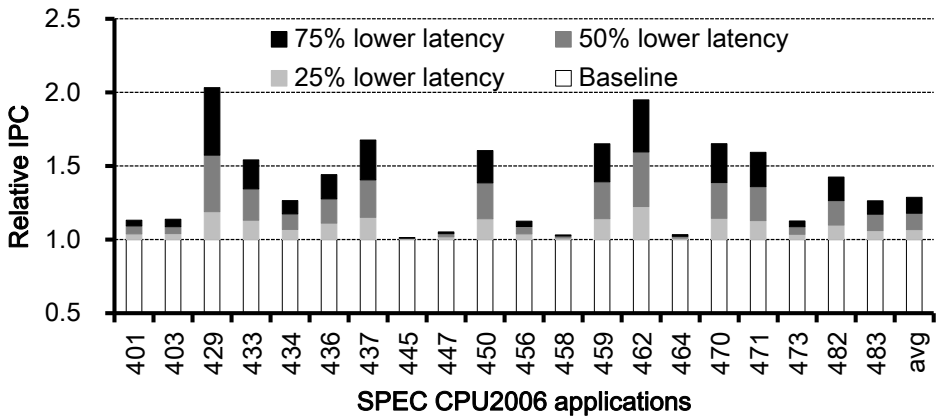


Figure 1.2: Many SPEC CPU2006 applications benefit from the reduction in memory access latency.

There have been several proposals for lowering the DRAM latency, including Reduced Latency DRAM (RLDRAM) [34], MoSys 1T-SRAM [13], and Fast Cycle DRAM (FCRAM) [42], but only with significantly increased die area. For example, an RLDRAM die is reported to be 40-80% larger than a comparable DDR2 DRAM die [20]. Alternatively, the ideas of embedding DRAM into processor dies [7], embedding SRAM into DRAM dies [56], or providing multiple row buffers per DRAM bank [14, 29] have been proposed, but they are more suitable for caches. Stacking DRAM

dies on top of the processor die [46] can reduce main-memory access latency and power, as the physical distances and impedance between the dies are greatly reduced. However, this technique is mostly applied to embedded systems due to high heat density and limited scalability in capacity.

This dissertation proposes asymmetric DRAM bank organizations to reduce the average main-memory access latency. Through detailed analysis of the access and cycle times of a contemporary DRAM device, we identify that it is critical to reduce both datapath capacitance within mats and data transfer distance from I/Os to the mats. We reduce the former by making fewer DRAM cells share a datapath wire, or equivalently, by increasing the aspect ratio of a mat. We add extra datapath and control wires to enable non-uniform bank accesses, which shorten the data transfer distance to banks located close to I/Os at the center of a device. By synergistically combining these two aforementioned techniques, we devise a novel asymmetric DRAM bank organization with center high-aspect-ratio mats (CHARM). Exploiting the observation that a relatively small subset of memory pages are performance-critical [9, 39], CHARM places the low-latency high-aspect-ratio mats only for a small subset of banks at the center of the device and hence maintains low area overhead. Our evaluation demonstrates that CHARM improves system performance and energy efficiency for a variety of workloads with minimal area overhead. When CHARM DRAM devices with 2× higher aspect ratio mats are placed on a quarter of the DRAM banks

at the center, our simulation results on SPEC CPU2006 benchmarks [15] show improvements in both the IPC and energy-delay product (EDP) by 4.4% and 7.6% on average, and up to 21% and 32%, respectively, with 3% area overhead. Multiprogrammed and multithreaded workloads also benefit from CHARM, depending on their bandwidth demands and latency sensitivities.

However, the new NUMA memory devices and systems often require intrusive modifications along the memory access path in hardware, software, or both. For example, software needs to be rewritten to identify hot memory regions and allocate them to the fast banks. Without such criticality-aware memory allocation, it is difficult to achieve robust performance with CHARM.

To address this problem, we propose SALAD, a new DRAM device architecture that provides symmetric access latency with asymmetric DRAM bank organizations. SALAD leverages the asymmetric bank structure of CHARM in an opposite way — it applies high aspect-ratio (i.e., low-latency) mats to remote banks, instead of local banks, thus reducing the unbuffered access latency (tAC) of a whole device. This balanced latency reduction is achieved by different banks reducing different components of tAC, which is a sum of activate time (tRCD) and column access latency (tCL). Remote banks have lower bitline capacitance, and hence lower tRCD; local banks have lower I/O transfer time due to shorter topological distance, and hence lower tCL. By providing uniformly low access time, SALAD not only obviates the need for expensive software changes but also improves bank-level parallelism by

better spreading memory accesses across all banks, instead of a handful of hot (fast) banks. Our evaluation shows that, even with data criticality-oblivious mapping, SALAD improves the IPC by 13% (10%), while incurring 6% (3%) area overhead. This compares favorably to the 9% (5%) IPC improvement with all banks resized to have smaller bitline capacitance and 12% (10%) with CHARM augmented with a hot page-aware mapping. When the hot page-aware mapping is employed, SALAD outperforms CHARM by 4% (2%).

Also, modern DRAM devices have multiple banks, each of which consists of thousands of rows but has only one row buffer. To access data in a row, the row should be activated (opened) first. If it is already opened (a row-buffer hit), the activation process is not needed, reducing latency and energy. If the row buffer is occupied by another row (a row-buffer miss), the buffer should first be deactivated. Hasty deactivation needlessly re-opens rows for otherwise row-buffer hits while hindsight accompanies the deactivation process on the critical path of accessing data for row-buffer misses. Because the time to (de)activate a row is comparable to the time to read data from an opened row, around a dozen nanoseconds for modern DRAM devices [72, 41], it is crucial to properly manage the row buffers by correctly deciding when to close rows. A memory controller has a request queue and it decides when to close the row in a bank based on the pending requests to the bank. However, the memory controller often does not have sufficient pending requests to a bank because the number of banks

per DRAM device keeps increasing, applications often have ample bank-level parallelism, and the size of the request queue is limited [66], all of which limit the number of requests per bank. Therefore, the memory controller has to rely on prediction in managing the row buffers.

Two popular row-buffer management policies are the open- and close-page policies [40], in which the former always keeps a row open and the latter closes it, both statically. Because memory access patterns vary significantly between applications and phases within each, there would be no clear winner between the two. More adaptive policies, which switch between open- and close-page policies based on a recent history of the row-buffer hit rates [16] and delay the precharge for a certain period [67, 19], have been proposed and exhibited a higher prediction accuracy than the static policies. However, their effectiveness is limited due to the dynamic nature of application behaviors, quantified in Section 5.3.

In this dissertation, we propose a novel DRAM microarchitecture that can obviate the need for any prediction instead of improving its accuracy. The open-page policy has a higher latency than the close-page policy on a row-buffer miss because the former needs to deactivate the currently active row. Even though deactivating the row buffer itself takes less than a nanosecond, precharging the bitlines (BLs) connected or coupled with the row buffer in conventional DRAM devices takes most of the deactivation time due to their dominance in total capacitance. By decoupling the row buffers from the BLs through adding isolation

transistors between them, we can start BL precharging right after activating a row while making the row buffer hold the active row. This early-precharge policy, enabled by row-buffer decoupling, allows for BL precharging off the critical path in most row-buffer misses, achieving the latency of the open-page policy for row-buffer hits and a latency nearly identical to the close-page policy for misses which is the better of both static policies. The area overhead is minimal (0.8% for the modeled devices) because the isolation transistors can be an order of magnitude smaller than the bitline sense amplifiers (BLSAs), which function as row buffers, without affecting the activation time. The early-precharge policy incurs more frequent BL precharges for writes and hence consumes more energy, but the system-level static energy reduction due to improved performance outweighs this overhead.

We also show that this row-buffer decoupling enables richer design options by separating internal DRAM μ-operations and recombining them to DRAM commands, giving more flexibility in controlling the timings of the DRAM μ-operations. For example, a restore μ-operation, which is required as a part of an activate process due to the destructive nature of reading DRAM cells, can be delayed until a row-buffer miss by utilizing the isolation transistors, hoping for reducing the restore energy. We also augment the early-precharge policy by exposing activate and write DRAM μ-operations to the memory controller such that redundant restores and BL precharges are minimized in the cases of bursty writes to an open row. Our experiments on a simulated chip-multiprocessor

system showed that the augmented early-precharge policy offered by row-buffer decoupling improves the geometric mean of the instructions per cycle (IPC) and MIPS2/W by 24% and 43% over the open-page policy and by 14% and 29% over the adaptive row-buffer management policy [16], respectively, for nine memory-intensive SPEC CPU2006 applications.

1.1 Research Contributions

In summary, this dissertation makes the following contributions:

- We present detailed breakdowns of the DRAM access and cycle times, through which we identify the key structures to reorganize within and outside of DRAM mats.
- We reduce the access and cycle times by increasing the aspect ratio of the mats and further improve the access time for local banks with better-than-worst-case delays.
- We propose CHARM, a practical solution to the latency problem of DRAM with minimal area overhead, which exploits the non-uniform criticality of memory accesses across the entire memory footprint.
- We quantify the system-wide benefits of CHARM in performance (IPC) and energy efficiency (EDP) using various workloads.
- To address the limitations of CHARM with non-uniform access time, we propose SALAD that provides symmetric access latency with asymmetric DRAM bank organizations.
- By providing uniformly low access time, SALAD not only obviates the need for expensive software changes but also improves bank-level parallelism by better spreading memory accesses across all banks, instead of a handful of hot (fast) banks.
- We identified that BLs dominate BLSAs, or row buffers, in

capacitance. As a result, BL precharging takes most of the row deactivation time, which is the source of the performance penalty of the open-page policy.

- We propose a novel DRAM microarchitecture that minimizes the row deactivation time by decoupling the BLSAs from the BLs, allowing the BLSAs to hold an active row and the BLs to be precharged right after row activation, which is mostly off the critical path for row-buffer misses.
- We explore the design space of scheduling DRAM μ-operations enabled by row-buffer decoupling, and examine the design options of mitigating the excessive power consumption of the early-precharge scheme.
- We quantify the impact of row-buffer decoupling on the performance and energy efficiency of a simulated multicore system and demonstrate its superior performance over various system configurations.

1.2 Outline

We introduce the organization of this dissertation as follows.

In Chapter 2, we explain the background on modern DRAM. Chapter 3 and 4 describe the detailed organizations and operations of the asymmetric DRAM architectures, called *CHARM* and *SALAD* respectively. We explore the *Row-Buffer Decoupling architecture* in Chapter 5, where we also evaluate the system-level impacts and benefits on the performance and energy efficiency when it is applied to the main memory system in common with Chapter3 and 4. In Chapter 6, we review the related works and finally, Chapter 7 presents the conclusion of this dissertation.

Chapter 2

Background

We first review the pertinent details of the organization and operations of contemporary DRAM devices to understand how various DRAM timing parameters influence the main-memory access latency. In so doing, we emphasize the importance of reducing both DRAM access time and cycle time to improve application performance. And then, we review the memory access scheduling and DRAM page management challenges.

2.1 Modern DRAM Device Organization

The continuing evolution of DRAM device organization has steadily increased its capacity and bandwidth even under tight cost constraints. A DRAM cell, which stores a bit of data, consists of a

---

This Section is based on [1, 2, 3]. - © [2013, 2014, 2016] ACM and IEEE Reprinted, with permissions, from ISCA' 13, ISCA' 14 and CAL' 16.

transistor and a capacitor. Multiple DRAM cells share control and datapath wires, called wordlines and bitlines, respectively, to improve area efficiency. In this 2D array structure, a row decoder specifies the wordline to drive, and a column decoder chooses one or more bitlines to transfer data to and from I/O pads. The wordlines and bitlines are made of metallic or poly-silicon stripes to minimize the area overhead due to wiring. As the capacity of a device increases, the resistance and capacitance of these wordlines and bitlines also increase rapidly, leading to high access and cycle times. To address this problem, hierarchical structures [49] have been applied to the control and datapath wires such that an array is divided into multiple mats, where each mat has sub-wordline drivers and local bitline sense amplifiers. Global datalines connect the local sense amplifiers to the data I/Os. The global datalines also have sense amplifiers to increase transfer speed.

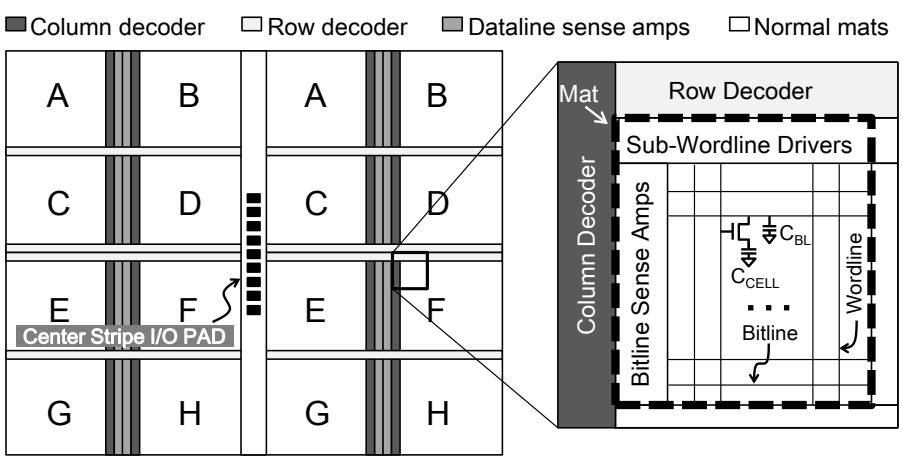


Figure 2.1: A typical modern main-memory DRAM organization with multiple banks and hierarchical structures.

Parameter	Symbol	Min (ns)
Activate to read delay	tRCD	13.75
Read to first data delay (= tCK×CL)	tAA	13.75
Access time (= tRCD + tAA)	tAC	27.5
Activate to precharge delay	tRAS	35
Precharge command period	tRP	13.75
Cycle time (= tRAS + tRP)	tRC	48.75
Read to read command delay	tCCD	5
Activate to activate command delay	tRRD	6
Four bank activate window	tFAW	30

Table 2.1: DDR3-1600 (1.25ns tCK) timing parameters [41].

DRAM devices have adopted prefetching and multi-bank architectures [24] to improve the sequential and random access bandwidth. Instead of increasing the internal operating frequency through deep pipelining (i.e., reducing tCCD), the DRAM mats transfer more bits in parallel through a wide datapath to keep up with ever-surging bandwidth demands. The transfer rate of a data I/O (2/tCK) is 8 times higher than the operating frequency of DDR3 [41] and DDR4 DRAM arrays. Hence, a DRAM array has 8 times more global datalines than the data I/O width (×N), which is called the prefetch size. This prefetching increases DRAM access granularity. Because a row in an array must be latched in the sense amplifiers before transferring data, it takes time to latch another row in the same array, which is defined as the cycle time (tRC). In

modern DRAM devices, the cycle time (~50ns in Table 2.1) is much longer than the reciprocal of the array's operating frequency (5ns in Table 2.1). For random accesses, the number of accesses to a specific row is at most a few; therefore, a mismatch in the tRC and tCCD leads to a poor performance. This problem is alleviated by having multiple banks where each bank is essentially an independent array, while sharing resources with others, such as I/Os, DLLs, and charge pumps. The multi-bank devices have inter-bank datalines to connect the global datalines of each bank to the data I/Os. Depending on target applications (e.g. graphics [8] and mobile [22, 33]), the designs of the I/O interfaces, the widths of the global datalines, or the transistor characteristics are modified, whereas the internal organization is mostly unchanged, as illustrated in Figure 2.1.

The capacity and bandwidth of DRAM devices have increased rapidly for years, but the latency, especially the cycle time has improved much more slowly. Figure 1.1 shows the relative capacity, bandwidth, and cycle time of multiple generations of DRAM devices. The DRAM cell size has been reduced from 8F2 to 6F2 and 4F2 [17], where F is the minimum wire pitch. The reduction in cell size, continuing evolution of process technology, and prefetching techniques all lead to an over 10× improvement in the capacity and bandwidth when we compare SDR and DDR3 DRAM devices. Meanwhile, the cycle time has improved much more slowly; tRC of DDR3 DRAM is still more than half that of SDR DRAM.

2.2 How DRAM Devices Operate

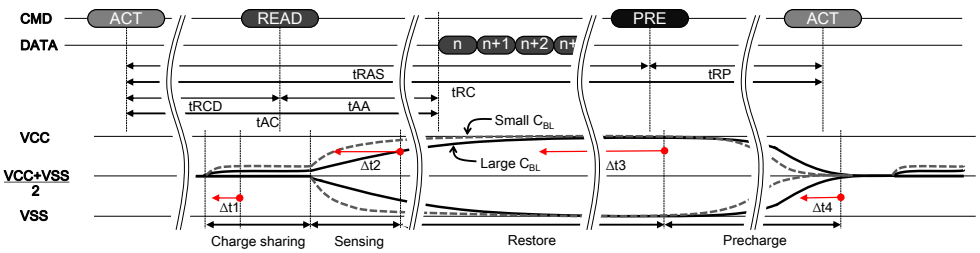


Figure 2.2: A sequence of DRAM commands and the corresponding changes in voltage of a bitline over a DRAM cycle.

A memory controller manages DRAM devices under various timing constraints imposed by resource sharing, limitations on the operation speed and power, and the volatile nature of DRAM cells. The memory controller receives requests from various components of a processor, stores them into a request queue, and generates a sequence of commands to the attached DRAM devices to service the requests. The devices are grouped into one or more ranks, where all of the devices in a rank receive the same commands and operate in tandem. The ranks that share control and datapath I/Os compose a memory channel. A memory controller controls one or more channels.

Figure 2.2 shows a sequence of commands injected into a DRAM bank and the corresponding changes in the voltage of a bitline over a DRAM cycle time. When a bank is idle, bitlines and other datalines remain precharged and no data is latched in the bitline sense amplifiers. First, an activate command (ACT) arrives at the I/O pads and proceeds to the row decoder of the specified

bank, where the decoder drives the specified wordline. On the mats that include the selected row, the access transistors controlled by the wordline are turned on and connect the DRAM cells to the bitlines in the mats. This charge sharing develops a voltage difference between the shared bitlines and the reference non-shared bitlines. Because the voltage difference is small due to the limited cell capacitance, sense amplifiers (often called row buffers) are used to speed up the voltage sensing for the cells whose voltage level is either VSS (zero) or VCC (one).

Once the values are latched, column-level commands, such as read (RD) and write (WR), are applied to the open (activated) row and the specified address regions are accessed. The access time of a device (tAC) is the sum of the minimum ACT to RD/WR time (tRCD) and the RD command to the first data in the I/O time (tAA 2). Once the sense amplifiers latch the values, the bitlines are fully charged or discharged over time to store the latched data back into the cells in the open row. This process is known as the restore process. Then, the bank can accept a precharge command (PRE), which changes the voltage of the datapath wires back to (VCC + VSS)/2, the original voltage level, to access data in another row. The cycle time of a device (tRC) is the sum of the ACT to PRE time (tRAS) and the precharge time (tRP). A bank with an open row can read or write a batch of data (8×N bits for DDR3) on every tCCD, but only one bank can occupy the data I/Os of the device at any given time, which determines the device bandwidth. The capability of internal power delivery networks limits the number of the ACT

commands that a device can process during the period of tFAW and tRRD. Table 2.1 summarizes the major timing constraints on a state-of-the-art DDR3 device.

2.3 The Impact of DRAM Timing Parameters on Memory Access Latency

All the DRAM timing parameters reviewed thus far affect how much time it takes for a memory controller to process an arriving request, while their relative impact depends on memory access patterns and access scheduling policies [40]. When a request arrives at the controller, if it has other pending requests and the scheduling policy determines to process them first, the request experiences a queuing delay. The major parameters that influence the amount of this delay are tCCD and tRC. If the pending requests are well distributed across the banks or concentrated into a certain row in a bank, each request can be serviced at an interval of tCCD. Otherwise, tRC determines the minimum number of cycles needed to service any two requests that access the same bank but different rows in the bank. Besides, tFAW and tRRD may impose additional constraints for a controller operating with fewer DRAM ranks.

Once a request is ready to be serviced after experiencing an optional queuing delay, the memory controller generates one or more commands to process the request. If the bank targeted by the request has a row open and the row is different from the target of the request, the controller initially needs to wait until the bitline

voltages are fully developed to either VCC or VSS (governed by tRAS), after which it precharges the corresponding bitlines (tRP), activates the target row (tRCD), and accesses the specified column (tAA). If the bitlines of the bank are already precharged or if the target row is already activated, the time to service the request can further be reduced. All of these facts show that both the access time and the cycle time of DRAM devices heavily influence main-memory access latency.

Each application has a different degree of memory-level parallelism (MLP), which determines its level of performance sensitivity on the DRAM timing parameters. The performance of the applications that have higher MLP, such as STREAM [32], typically depends less on main-memory access latency and more on the bandwidth (tCCD), while that of applications with lower MLP is often sensitive to the access latency, as evidenced in Figure 1.2, which shows the performance sensitivity of SPEC CPU2006 applications. Therefore, it is of great importance to devise DRAM microarchitectures that reduce both the access time and the cycle time of DRAM devices.

2.4 State-Dependent DRAM Latency

The modern main memory system typically has one or more channels. A channel connects a memory controller with one or more dual in-line memory modules (DIMMs), each of which is divided into one or more ranks. A rank consists of multiple banks, where

each bank can service a memory request independently. To provide the necessary bandwidth and capacity, a rank is often constructed with multiple DRAM devices connected in tandem; for a rank comprised of N DRAM devices, each DRAM device contributes to 1/N of the total bandwidth and capacity of a rank and a bank. Due to tight cost constraints, only one row (page) can be buffered (opened) in a row buffer per bank and actively service requests at a time.

When a memory controller receives a request, the latency for servicing the request significantly depends on the internal state of the bank servicing the request, apart from the queuing delay. First, consider a state in which the row buffer of the bank is invalidated after the datapath within the bank is precharged. For a given address, the corresponding row should be activated by an activate (ACT) command, which takes tRCD; the row is accessed and the data from the row are subsequently stored into the row buffer. Then, the data in the specified columns are accessed after tAA. The latency of such an access is tRCD+tAA. Second, consider a state in which the row stays open, maintaining the data from the previous request in the row buffer. If the current row address is the same as the previous one, a row activation (i.e., sending an ACT command) is not needed. Thus, the latency of such an access is simply tAA. However, if the current row address is not the same as the previous one, the previously opened row should be closed (deactivated) by a precharge (PRE) command, taking tRP, and then the row for the current address is activated. The latency of such a read is tRP+tRCD+tAA. Moreover, transitioning from one state to another

takes a varied amount of time depending on the various DRAM timing and resource constraints. Thus, processor-memory interfaces, such as DDR3/4 [72, 34], LPDDR2/3 [33], and GDDR5 [73], expose DRAM microarchitecture to a memory controller so that the memory controller can track internal bank states and make the best scheduling decision for the pending requests to minimize their average service latency.

2.5 Memory Access Scheduling and Page Management Challenges

A memory controller enqueues memory requests from processor cores and other sources in its request queue, tracks internal states of banks, and generates commands for the banks to service the requests, while considering various DRAM timing and resource constraints. The simplest memory controller (i.e., first-come-first-serve (FCFS)) checks the address of the oldest request in the queue and generates one or more commands to read or write data in the corresponding location depending on the internal state of the bank specified by a given address.

More advanced controllers check other recent requests in the queue and service a request by sending one or more commands if it satisfies DRAM resource and timing constraints. This effectively reorders the requests to improve channel utilization. Reordering may increase the access latency of some requests, but higher channel utilization mostly leads to lower latency for most requests

on average. Typically, an older request, which has stayed longer in the queue, is considered with a higher priority than a younger request (first-ready FCFS (FRFCFS) [40]). There can be other priority criteria, such as the number of requests from a certain CPU core [35], the access type (e.g., read over write), the access address (higher priority to the currently open rows), and the annotated criticality information. However, the degree of reordering can be limited to improve the quality of service, such as batching [35].

As discussed earlier, the latency and the number of commands to service a request heavily depend on the internal states of the banks. Hence, it is important for the memory controller to decide whether to close the currently open row that has just serviced a request. If any requests in the queue target the currently open row, it is better to keep the row open. In contrast, it is better to close the row by sending a PRE command to the bank if no request in the queue targets the currently open row. If a memory controller has no more enqueued request to the bank that has just serviced a request, it has to decide whether to close the currently open row after predicting the row address of future requests to the bank and act accordingly.

Misprediction penalty varies case by case, but its amount is significant because the precharge time (tRP) and the activate time (tRCD) are comparable to the address access time (tAA) taken from determination of the address to start of output of the read data. When it decides to keep a row open but the next request does not

hit the same row in the bank, the subsequent request experiences an additional latency up to tRP because the datapath within the bank must be precharged before the next row is activated. When it decides to close a row but the next request hits the same row, that request experiences an additional latency of tRCD (or up to tRC, minimum ACT to ACT time to a bank, in the worst case when the next request arrives right after the decision is made and read-with-autoprecharge is used [41]). Moreover, such mis-prediction wastes DRAM energy due to one extra pair of ACT and PRE.

Predicting whether to close a page is frequently inevitable in modern computer systems. If the request queue of a memory controller holds enough requests for at least two requests per bank, prediction is not needed at all. However, there are two reasons that this is unlikely to occur. First, applications often have ample bank-level parallelism (BLP). Thus, the number of enqueued requests to a specific bank can be small even though the applications have many pending requests in the queue. Second, the number of banks per DRAM device is increasing in contemporary DRAM architectures to narrow the performance gap between sequential and random memory accesses [16, 72, 38]. However, the request queue is an expensive resource and in the critical path [66]. Consequently, its size does not typically surpass few dozen slots. These two factors (i.e., increasing number of banks and limited request queue size) lead to even fewer enqueued requests per bank on average (i.e., higher probability of no or just one enqueued request to some banks). Therefore, unless an application has bursty memory

requests with high spatial locality, which is not typically the case, prediction is needed more frequently.

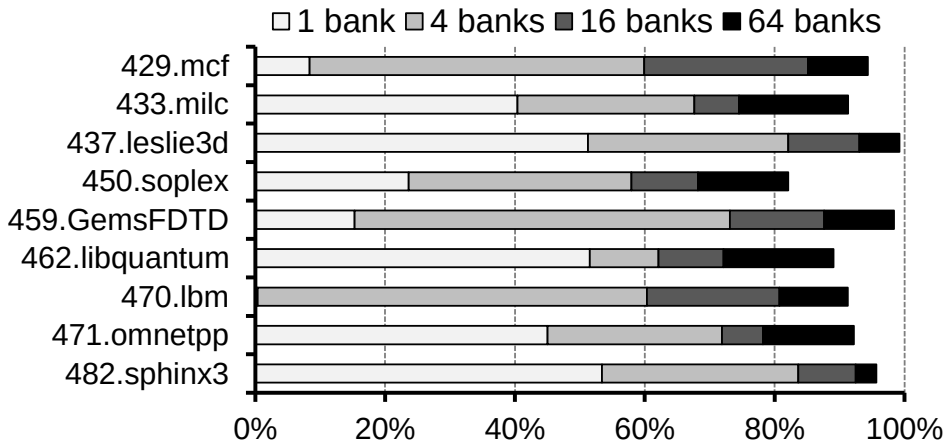


Figure 2.3: The rate that an access request needs predictions due to lack of information available for the controller on spec-high applications listed in Table 5.4.

Figure 2.3 shows the percentage of DRAM read and write requests that need predictions due to insufficient information (i.e., enqueued requests) available for the memory controller. The simulation methodology is detailed in Section 5.2. In many SPEC CPU2006 benchmark applications [15], predictions are needed for more than 80% of the requests even with 16 banks per memory controller. Furthermore, the percentage increases as the number of banks increases regardless of the applications.

The simplest prediction scheme is always to either close the row right after servicing a request or keep it open statically, which is similar to static branch prediction (i.e., always-taken or always-

non-taken). The former and the latter are the close- and open-page row-buffer management policies, respectively. Because the spatial locality of main memory accesses varies significantly depending on the applications, there is no clear winner between the two shown in Figure 2.2 (more in Section 5.3).

Considering the dynamic nature of the application behaviors, however, it is challenging to achieve a high prediction accuracy with either the close- or open-page policy. Consequently, various adaptive page-management policies, which ping-pong between the open- and close-page policies [70] and delay precharging [67, 19], have been proposed. Such adaptive policies can have a higher prediction accuracy than that of the static ones but with only limited success.

Chapter 3

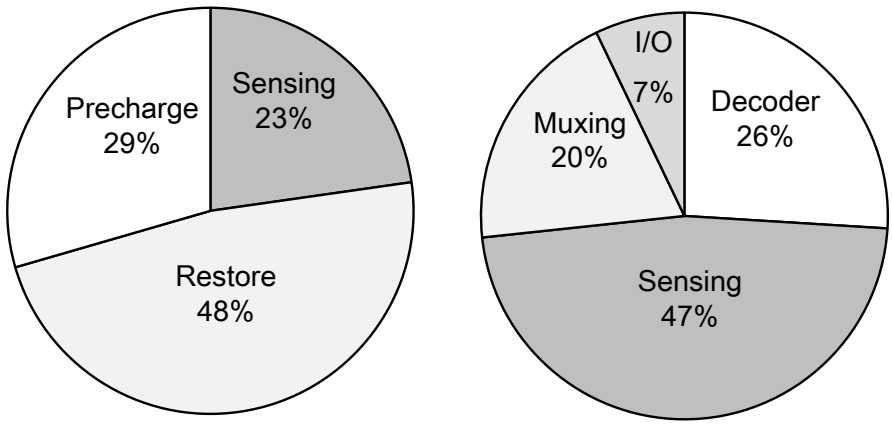
Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations

3.1 Asymmetric DRAM Bank Organizations

In this section, we first identify the key contributors of the cycle and access times of a modern DRAM device. Then, we reduce its cycle time by decreasing the number of wordlines per mat, which also lowers the access time and power dissipation at the cost of an area increase. Based on the observation that the main-memory accesses of many applications are not uniformly distributed over their memory footprints, we alleviate the area overhead by locating mats with different aspect ratios together and further reduce the access time on a portion of the device by placing mats with high aspect ratios close to the I/Os.

This chapter is based on [1]. - © [2013] ACM. Reprinted, with permission, from ISCA'13.

3.1.1 DRAM Cycle and Access Time Analysis



(a) tRC (cycle time) (b) tAC (access time)

Figure 3.1: The DRAM timing breakdown of (a) tRC and (b) tAC.

To devise techniques to lower memory access latency, we first analyze the cycle and access times of a contemporary DRAM device to understand their key delay components. We use a 4Gb Samsung DDR3 device in a 28nm process technology [41] as a reference chip with the following assumptions: (1) each bitline is connected to 512 cells; (2) each mat has 512 bitlines; (3) the cell array width of a mat is 9 times the width of a bitline sense amplifier; (4) the bitline capacitance is 6 times higher than the cell capacitance based on recently reported values [16, 52]. The simulated 8Gb device, which serves as baseline organization, has 8 banks and follows the DDR3 specification. We use the PTM low power model [57] for SPICE simulation.

The simulation results show that the critical path of the cycle time (Figure 3.1(a)) is composed of sensing, restore, and

precharge processes, all of which only depend on the structures within a mat. Moreover, all the processes involve manipulation of bitline voltages so that managing the capacitance and resistance of the bitline heavily influences the cycle time. The access time (Figure 3.1(b)) is affected not only by the bitline sensing time (tRCD) but also by the address decoding process, transfer and rate conversion times through datalines and multiplexers (muxing), and I/O driving time. Both the address decoding process and dataline transfer time depend on the physical distance of control and datapath wires between individual mats and I/Os. As a result, it is necessary to reduce the time taken both within and outside of mats.

3.1.2 Low-Latency Mats with High Aspect Ratios

We first focus on reducing the time to load and store data within mats. The analysis in Section 3.1.1 shows that the sensing, restore, and precharge processes are all sensitive to the local bitline capacitance. Hence, we propose to decrease the number of wordlines per mat, which makes fewer DRAM cells attached to a bitline and reduces the bitline capacitance. Figure 2.2 illustrates the effects of the reduced bitline capacitance. The amount of voltage developed by the charge sharing between a bitline and a cell increases as the bitline capacitance decreases, thus reducing the sensing time (Δt1 and Δt2). The bitline capacitance is a significant portion of the load capacitance of the sense amplifiers and the precharge drivers in a mat, so both the cell restore time (Δt3) and

the precharge time (⊿t4) decrease. Note that decreasing the number of bitlines per mat has little influence on the sensing, restore, and precharge time and is not further explored in the dissertation.

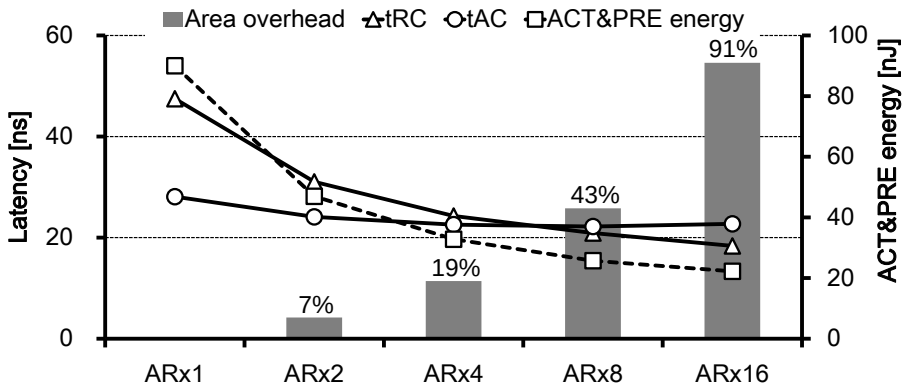


Figure 3.2: The area overhead, tRC, tAC, and activate and precharge energy of a DRAM device as the number of wordlines per mat increases.

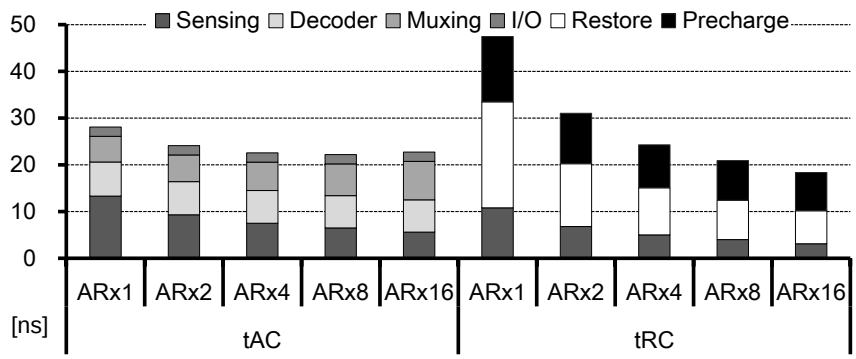


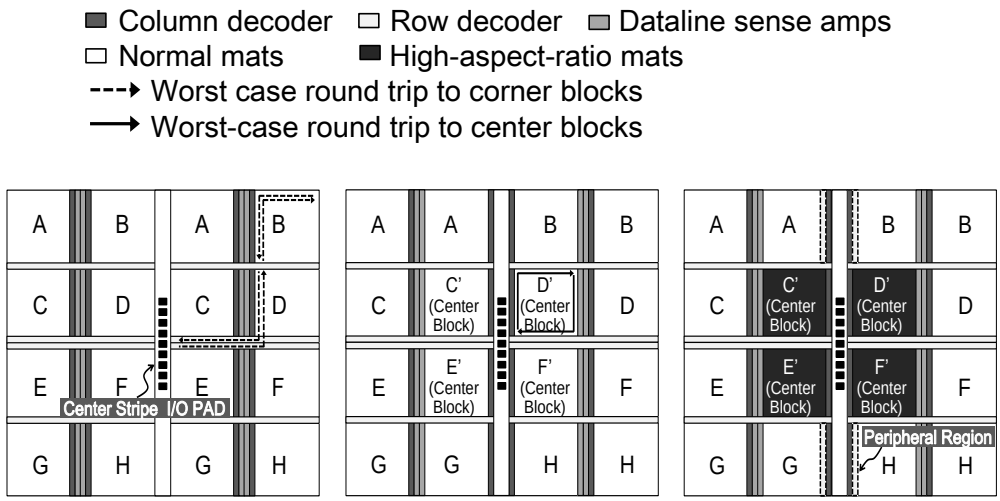
Figure 3.3: The breakdowns of tAC and tRC over various numbers of wordlines per mat. Fewer wordlines per mat lower sensing and precharge delays significantly first, but we then see diminishing returns.

There are two main sources of overhead when decreasing the number of wordlines per mat. First, a bank needs more mats to keep its capacity constant as each mat has fewer wordlines. Because a mat has the same number of sense amplifiers regardless of the number of wordlines, having more mats incurs area overhead. Second, a global dataline within a bank becomes longer and is connected to more mats. This increases the fan-in and junction capacitance of the global dataline. However, the impact of this increase in capacitance on the access latency is limited because global datalines, which are made of metal, have much lower capacitance and resistance than bitline wires.

Figures 3.2 and 3.3 present the SPICE simulation results of the area overhead, tRC, tAC, and the activate and precharge energy of the device with varying numbers of wordlines per mat. The reference mat has 512 bitlines and 512 wordlines. Therefore, decreasing the number of wordlines is equivalent to increasing the aspect ratio of the mat. We use the notation AR×N to denote a mat with aspect ratio N. A mat with 128 wordlines (AR×4) has half the cycle time of the reference mat, while further quadrupling the aspect ratio results in additional 10% reduction. Increasing the aspect ratio also reduces the activate and precharge energy of the mat. Figure 3.2 shows that a mat with 128 wordlines requires 33% less activate and precharge energy compared to the reference mat. In contrast, increasing the aspect ratio does not improve tAC much because the sensing process becomes faster but the decoding speed is mostly unchanged while the datapath delay between mats and

I/Os becomes worse, as shown in Figure 3.3. Also, the area increases rapidly with more mats per bank (Figure 3.2). Using four times more mats incurs an area overhead of 19%, and the bank becomes almost twice as large with 16 times more mats. Because the cost of DRAM devices is very sensitive to the area, we need to devise microarchitectures that limit the area overhead and further decrease the DRAM access time.

3.1.3 Banks with a Non-Uniform Access Time



(a) Reference organization (b) Non-uniformity in bank accesses (c) Center high-aspect-ratio mats

Figure 3.4: A DRAM device with (a) the reference structure is reorganized to (b) support non-uniformity in bank accesses and (c) replace the mats in the center blocks with high-aspect-ratio mats (CHARM).

Because decreasing the number of wordlines per mat has a limited impact on the DRAM access time, it is necessary to decrease the physical distance between the banks and I/Os to reduce the access time further. We leverage the idea of non-uniform cache architecture [21] and propose a multibank organization with a non-uniform access time in which the control and data transfer time between a bank and I/Os depends on the location of the bank within a device. Here we assume that the I/Os are located at the center of the device without a loss of generality.

To improve the random access performance, many modern main-memory DRAM devices have 8 or 16 banks. A bank is typically implemented with one or more blocks. Each block has a row decoder and a column decoder to process the assigned request. Existing DRAM specifications, such as DDR3 and DDR4, assume a single, uniform access time to all the banks. Therefore, devices complying with the specifications are designed to improve the area efficiency under the constraint that all of the banks have the same transfer time regardless of their locations. For example, a split-bank architecture is applied to the baseline DDR3 DRAM organization shown in Figure 2.1 and Figure 3.4(a). Either side of a device has 8 blocks, each of which corresponds to half of a bank. Each pair of blocks operates in tandem such that one of the blocks takes charge of half of the data being transferred. This can reduce the number of inter-bank datalines by half compared to the organization that places an entire bank on a single side.

We reduce the access time to the blocks located at the center

of a device by relocating the column decoders to the center stripe and by making a single block, not a pair of blocks, take charge of a data transfer instance, as shown in Figure 3.4(b). This enables the center blocks (e.g., Block D') to start decoding column addresses much earlier than the corner blocks (e.g., Block B). We also group the blocks of a bank together on the same side such that both blocks share the same dataline sense amplifiers, which keeps the number of dataline sense amplifiers needed in a device unchanged. Still, the number of inter-bank datalines is doubled compared to the split-bank architecture. These changes make the command path and inter-bank datalines of the 4 center blocks (the solid arrows in Figure 3.4(b)) become shorter than those of the 4 blocks at the corner (the dotted arrows in Figure 3.4(a)) due to the perimeter of a block in a round trip. The SPICE simulation results show that the access time of 4 center blocks is 6ns lower than that of the other blocks. The layout overhead of the relocated column decoders and the additional inter-bank dataline wires needed for data transfers increase the device area by 1%. In this dissertation, we assume that the corner blocks and the remaining blocks at the edges have the same access latency and leave the exploration of three-tier or more bank organizations for future work.

3.1.4 CHARM: Center High-Aspect-Ratio Mats

We synergistically combine the ideas of increasing the aspect ratio of the mats and introducing non-uniformity in bank accesses to

further decrease average access and cycle times of DRAM devices with low area overhead. Replacing all of the mats in a device with high-aspect-ratio (HAR) mats incurs high area overhead, as explained in Section 3.1.2. Instead, we only use HAR mats in the center blocks of the device as shown in Figure 3.4(c). We call this DRAM microarchitecture CHARM, Center High-Aspect-Ratio Mats.

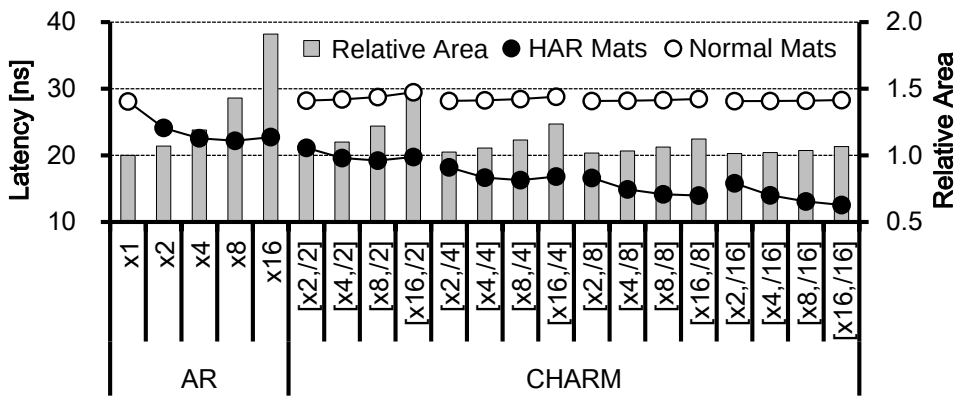


Figure 3.5: The access time to the center HAR mats and the remaining normal mats and the area overhead of various CHARM DRAM organizations. The reference organization has no HAR mats and accesses all the banks with uniform latency.

The CHARM microarchitecture can cut both the access time and the cycle time of the center HAR mats down to at least half of the values of the remaining mats with a normal aspect ratio while limiting the area increase rate to a single digit percentage. Because a DRAM block with HAR mats is larger than a block with only normal aspect ratio mats when both blocks have the same capacity, the blocks in the middle columns of the device are misaligned. This

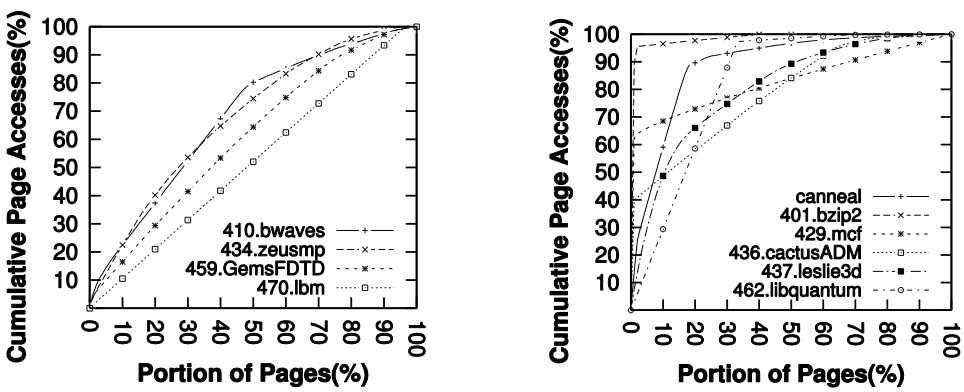
spare area can be easily filled with peripheral resources, such as charge-pump circuits, decoupling capacitors, and repeaters for inter-bank datalines, which further reduces the area overhead.

We compare the access time and the relative area of the CHARM and other organizations in Figure 3.5. The organizations that have the HAR mats in all of the blocks and uniformly access them (AR×N in the figure) has a single access time. The access time of an AR×N (N > 1) organization is lower than that of the reference organization (AR×1) mainly due to the decrease in the activate time (tRCD). The read-to-first-data delay (tAA) actually increases as the aspect ratio (N) increases because the device gets larger and more mats are connected to global datalines. We use the notation CHARM[×N,/M] for a CHARM DRAM that positions the HAR mats with the aspect ratio N only in the center blocks that encompass the one M-th of the device capacity. It alleviates the increase of tAA of the normal mats. As M increases, the tAA value of the normal mats becomes closer to that of the AR×1 and tAA of the HAR mats improves further. The access time of the HAR mats for CHARM[×4,/8] becomes 15ns, almost half the access time of the reference organization. Further increasing M improves the tAA of the HAR mats further, but with diminishing returns. The area overhead of CHARM[×N,/M] is lower than that of AR×N as well because the HAR mats exist only at the center blocks. For example, the area overhead of AR×4 is 19%, while the corresponding percentages of CHARM[×4,/4] and CHARM[×2,/4] are only 6% and 3%, respectively. Note that the cycle time of a mat only

depends on the aspect ratio of the mat. This is not shown in the figure.

The analysis thus far shows that CHARM DRAM organizations with center high-aspect-ratio mats can significantly improve the average cycle and access time with minimal area overhead. We quantitatively compare the system-level performance and energy efficiency of CHARM and the other DRAM organizations using popular single- and multithreaded benchmark suites in Section 3.3.

3.1.5 OS Page Allocation



(a) Accesses more uniformly distributed over pages

(b) Accesses less uniformly distributed over pages

Figure 3.6: Cumulative page accesses over the portion of touched pages on SPEC CPU2006 applications with high L2 cache MPKI and canneal from PARSEC where pages are sorted by the access frequency.

The non-uniform bank organization becomes more effective if we can assign performance-critical data to lower-latency blocks. Previous work suggests that a relatively small subset of pages is performance-critical [9, 39]. The performance criticality of an OS page can be estimated by (a combination of) various metrics, such as the access frequency and recency [25], the cache miss rate [9], and the TLB miss rate [48]. Note that this problem is nothing new; similar problems appear in other non-uniform memory systems such as distributed non-uniform memory access (NUMA) machines [48], hybrid memory systems [39], adaptive granularity memory systems [55], as well as conventional demand paging systems.

For the rest of this dissertation, we use access frequency as page criticality predictor due to its ease of implementation. Exploiting the non-uniformity of access frequencies across the entire memory footprint, we assign frequently-accessed pages to low-latency blocks in CHARM DRAM devices. The page access frequency can be easily measured via profiling. Note that this work primarily focuses on novel DRAM bank organizations and their performance potentials and that we leave more sophisticated page-allocation mechanisms for future work.

Figure 3.6 offers an evidence of the existence of the aforementioned access non-uniformity in SPEC CPU2006 applications [15]. The figure shows the cumulative memory access frequency over the portion of the accessed pages sorted according to the access frequency. Some applications access pages relatively evenly such that their cumulative curves closely resemble a

diagonal line (Figure 3.6(a)). They often repeat sweeping large blocks of memory rapidly (470.lbm and 459.GemsFDTD, for example) or access random locations (RADIX in SPLASH-2 [50]). In contrast, for other applications such as 429.mcf, 437.leslie3d, and 462.libquantum, relatively small portions of pages account for most of the page accesses (Figure 3.6(b)). These applications can benefit greatly from the proposed non-uniform bank organization approach.

The page placement decision can be made statically (e.g., compiler profiling, programmer annotations) or dynamically (e.g., OS tracking per-page memory access frequencies), possibly assisted by hardware support. A new memory allocation system call that maintains a separate free list for low latency page frames is necessary, which may also migrate data between low- and high-latency blocks as needed. In this dissertation, we take a static approach via off-line profiling to identify the most-frequently accessed data regions. We envision this process to be automated using compiler passes targeting the new system call. Automating profile-guided page allocation and evaluating OS-managed dynamic allocation are outside the scope of this dissertation. We evaluate the performance impact of allocating only a portion of frequently accessed pages to low-latency blocks in Section 3.3.

3.2 Experimental Setup

Parameter	Reference	CHARM[x2,/4]	
	(AR x1)	HAR mats	Normal mats
tRCD	14ns	10ns	14ns
tAA	14ns	8ns	14ns
tRAS	35ns	20ns	35ns
tRP	14ns	11ns	14ns
ACT+PRE energy	90nJ	47nJ	90nJ
RD energy	15.5nJ	13.8nJ	15.7nJ
WR energy	16.5nJ	14.7nJ	16.7nJ

Table 3.1: Power and timing parameters of the representative DRAM organizations.

We modeled a chip-multiprocessor system with multiple memory channels to evaluate the system-level impact of the DRAM devices with HAR mats and asymmetry in bank accesses on performance and energy efficiency. The system has 16 out-of-order cores. Each core operates at 3 GHz, issues and commits up to 4 instructions per cycle, has 64 reorder buffer entries, and has separate L1 instruction and data caches and a combined L2 cache. The size and associativity of each L1 cache and L2 cache are 16 KB and 4, and 512 KB and 16, respectively. Each cache has 4 banks with a line size of 64 B. A MESI protocol is used for cache coherency and a reverse directory is associated with each memory controller. The system has 8 memory controllers unless mentioned

otherwise, while each controller has one memory channel. There are two dual-inline memory modules (DIMMs) per channel and each DIMM consists of 2 ranks. Each rank has 9 8Gb ×8 DDR3-2000 DRAM devices, 8 for data and 1 for ECC, making the peak memory bandwidth of the system with 8 memory controllers 128 GB/s. Each memory controller uses the PAR-BS [35] and the open-row [19] policy for access scheduling and has 64 request queue entries.

Group	SPEC CPU2006 applications
spec-high	429.mcf, 433.milc, 437.leslie3d, 450.soplex, 459.GemsFDTD, 462.libquantum, 470.lbm, 471.omnetpp, 482.sphinx3
spec-med	401.bzip2, 403.gcc, 410.bwaves, 434.zeusmp, 435.gromacs, 436.cactusADM, 464.h264ref, 473.astar, 481.wrf, 483.xalancbmk
spec-low	400.perlbench, 416.gamess, 444.namd, 445.gobmk, 447.dealII, 453.povray, 454.calculix, 456.hmmer, 458.sjeng, 465.tonto

Table 3.2: We categorized the SPEC CPU2006 applications into 3 groups depending on their L2 cache MPKI values.

We slightly modified the memory access scheduler to give equal priority to the requests targeting both the center HAR mats and the

other normal mats. Row-level commands, such as ACT and PRE, change the status within the mats and only share the command path, which is sufficient to make the scheduler apply proper timing constraints depending on the mat type. Column-level commands share the command and datapath I/Os. The scheduler first assumes that even the center HAR mats have the tAA of the normal mats, after which it finds the request that has the highest priority and meets all the timing and resource constraints. When a RD or WR command to a center HAR mat is generated by the scheduler, it also checks whether the data can be transferred into the slot specified by the original tAA of the center HAR mat and uses the slot if available.

McSimA+ [5] was used for simulation. We obtained the power, area, and timing models of out-of-order cores and caches using McPAT [27]. The power and timing values of the physical interfaces between the processor and DRAM devices and their internal components are based on the DDR3 specifications [41] and the SPICE modeling results described in Section 3.1.1. The key timing and power parameters we obtained from the modeling in Section 3.1 on the representative DRAM organizations are summarized in Table 3.1. Note that if the increase in tAA on normal mats is smaller than tCK, we increase the size of drivers in inter-bank datalines to negate the increase in tAA. This increases the energy for read and write operations slightly, which is properly modeled and shown in the table.

We used the SPEC CPU2006 [15], SPLASH-2 [50], and

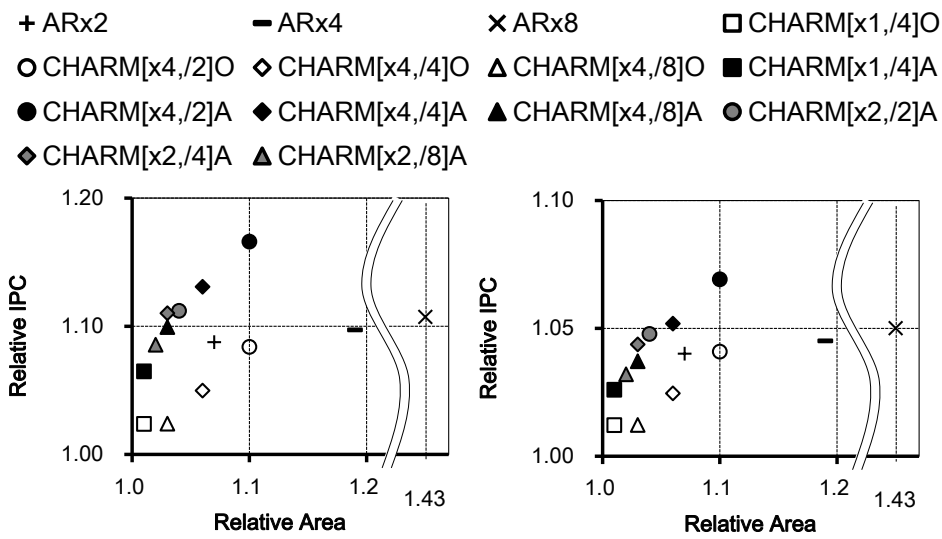
PARSEC [10] benchmark suites to evaluate the efficiency of the CHARM DRAM-based main memory system. We ran Simpoint [43] to find the representative phases of the SPEC CPU2006 applications, and selected the 4 slices with the highest weights per application. Each slice consists of 100 million instructions. We simulated regions of interest for SPLASH-2 and PARSEC. We used reference datasets for SPEC CPU2006, simlarge datasets for PARSEC, and the datasets listed in [27] for SPLASH-2. We classified SPEC CPU2006 applications into three groups based on L2 cache misses per kilo-instructions (MPKI) [15]. These are called spec-high, spec-med, and spec-low, as shown in Table 3.2. We created four mixtures of multiprogrammed workloads, one from each group and one from all three groups. The latter is called spec-blend and consists of five applications from spec-high, six from spec-med, and five from spec-low. For each multiprogrammed mixture, a simulation point is assigned to each core, and one or two highest weight points are used per application. We use a weighted speedup approach [44] to compare the performance of multiprogrammed workloads. We compute the weighted speedup by initially determining the relative performance of an application, which is the ratio of its IPC in a multiprogrammed environment to IPC in a standalone execution environment, and then aggregating the relative performance of all applications in the mixture.

3.3 Evaluation

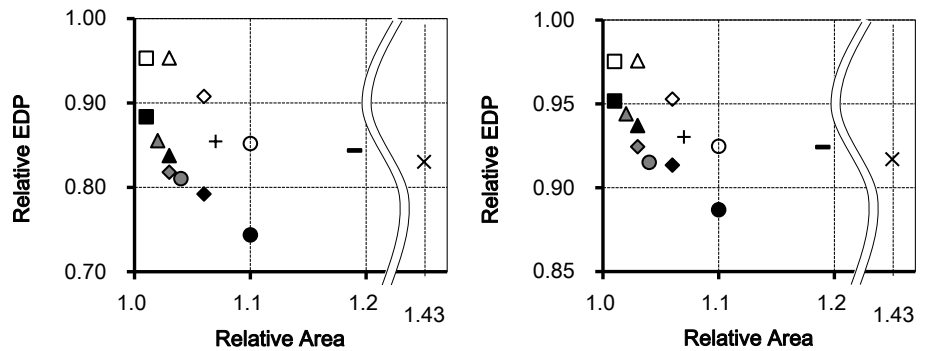
We evaluate the system-level impacts of CHARM on the performance and energy efficiency of a contemporary multicore system with a variety of workloads. We first execute SPEC CPU2006 programs to show how much improvement in instructions per cycle (IPC) and energy-delay product (EDP) is achievable with CHARM on single-threaded applications. Then, we run multiprogrammed and multithreaded workloads to evaluate the efficacy of CHARM DRAMs in scenarios in which main-memory devices serve accesses from multiple requesters with and without correlation.

3.3.1 Performance Impact on Single-Threaded Applications

Figure 3.7 shows scatter plots of the relative area (x-axis) versus the relative IPC or EDP (y-axis) for various DRAM organizations on single-threaded SPEC CPU2006 benchmark programs. The organization with normal mats and uniform accesses for all of the banks (AR×1) is the reference. Due to limited space, we only present the average values over all the programs (Figure 3.7(b) and Figure 3.7(d)) and the average values over those with high L2 cache MPKI, categorized as spec-high in Table 3.2 (Figure 3.7(a) and Figure 3.7(c)). For this experiment, we use only one memory controller to stress memory system bandwidth.



(a) IPC over area for spec-high (b) IPC over area for spec-all



(c) EDP over area for spec-high (d) EDP over area for spec-all

Figure 3.7: Scatter plots of the relative area and the relative IPC of the various DRAM organizations on (a) the average of SPEC CPU2006 applications with high main-memory bandwidth (spec-high in Table 3.2) and (b) the average of all SPEC CPU2006 applications (spec-all), and the relative area and the relative EDP on (c) the average of spec-high and (d) the average of spec-all.

We make the following observations from the experimental results. First, increasing the aspect ratio of all the mats in a device

improves the IPC (higher is better), but increasing it by more than twice the original value yields only marginal improvements in IPC. We define the return on investment (ROI) of IPC as the gain in IPC over the area overhead such that the gradients of the virtual lines between the points and the origin in the figures become the ROIs. Figure 3.7(b) shows that AR×2 improves the average IPC of all the SPEC CPU2006 applications by 4.0%, while AR×4 and AR×8 improve it by 4.5% and 5.0%, respectively. For applications with low intra-page spatial locality (i.e., with high ratio of row-level commands to column-level commands), such as 429.mcf and 436.cactusADM, increasing the aspect ratio improves performance because they benefit from lower tRC. However, for applications with high spatial locality, such as 437.leslie3d and 482.sphinx3, tAA influences more on performance than tRC. Increasing the aspect ratio in fact increases tAA, making those applications perform worse. Considering that AR×4 (19%) and AR×8 (43%) have much higher area overhead than AR×2 (7%), AR×2 has a higher ROI and can be regarded as a more effective configuration.

Second, CHARM DRAM devices have lower area overhead but its performance gain is affected by how effectively an application utilizes the center HAR mats. CHARM organizations with main-memory access frequency oblivious mapping (configurations with suffix O in Figure 3.7) result in smaller performance improvement than AR×N. For the CHARM[×N,/M]O, only one M-th of memory accesses head to the center HAR mats on average when the oblivious mapping is used. When M is small, a large portion of data

are allocated to the HAR mats, but the access time to the center HAR mats also increases and becomes closer to that of AR×N. As M increases, the access time to the center HAR mats becomes much lower than that of AR×N, but there is a fixed area overhead of implementing asymmetric bank accesses and the fewer memory accesses utilize the HAR mats. These trends make the ROI of CHARM[×N,/M]O comparable to that of AR×N. The organizations with access frequency aware mapping (ones with suffix A in Figure 3.7) yield substantially higher performance gain on average without any further area overhead. The IPCs of CHARM[×2,/2]A, CHARM[×2,/4]A, CHARM[×4,/2]A, and CHARM[×4,/4]A are all higher than the IPC of AR×8, which provides the highest performance with uniform bank accesses, on spec-high applications. The observation made above also holds here such that CHARM[×2,/M]A has a higher ROI than CHARM[×4,/M]A. Among those, CHARM[×2,/4]A gives the highest ROI with an 11% IPC improvement and a 3% area increase. Hereafter, we use it as default configuration.

Third, more performance gains are generally observed for applications with higher main-memory bandwidth demands. 429.mcf and 450.soplex, two of the top three bandwidth-demanding applications, provide the highest performance gains with CHARM[×2,/4]A, showing 21% and 19% increases in IPC, respectively. If we take the average IPC of the 9 SPEC CPU2006 applications with high L2 MPKI values, the IPC gain for CHARM[×2,/4]A is 11% higher than the average IPC gain for the

same configuration over the entire CPU2006 applications. Comparison of Figure 3.7(a) and Figure 3.7(b) shows that the relative order of the ROI values among the DRAM organizations is mostly preserved even though we only take the average of spec-high applications. Figure 3.8 shows the absolute and relative IPCs of SPEC CPU2006 applications for CHARM[×2,/4]A.

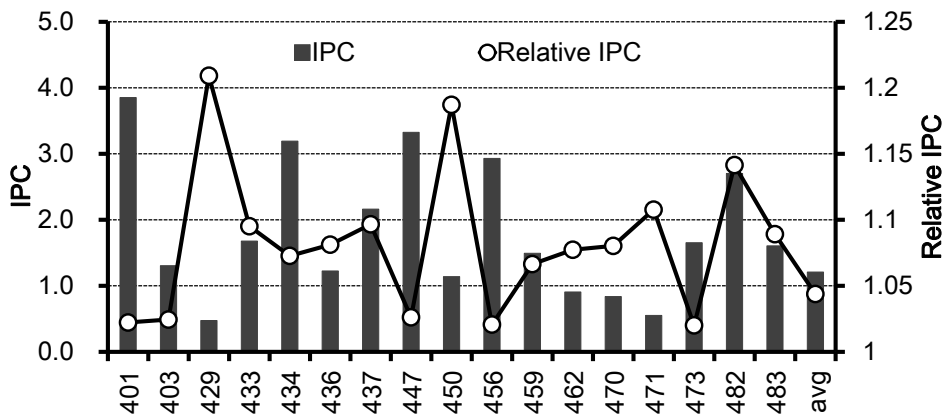
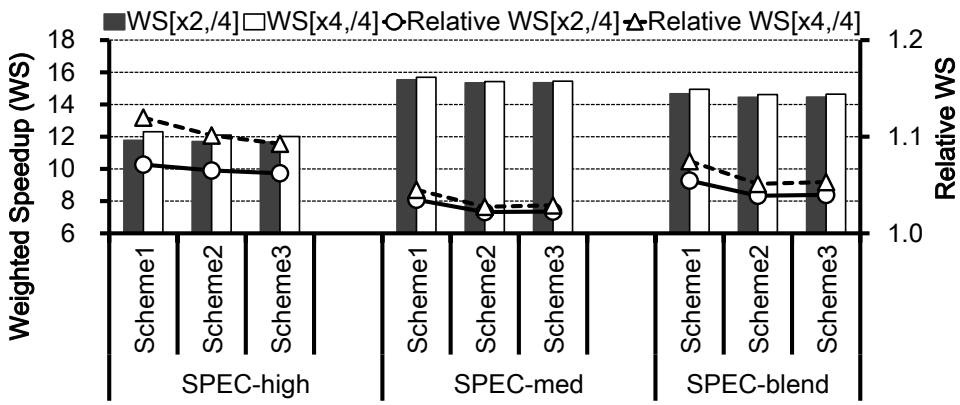


Figure 3.8: The absolute and relative IPCs of SPEC CPU2006 applications for CHARM[×2,/4]A. The applications with low memory bandwidth demand are omitted in the graph, but included when the average values are computed.

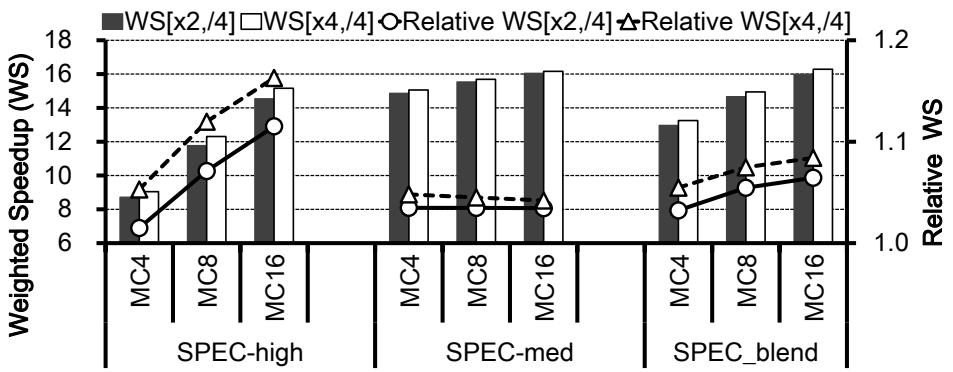
Fourth, the EDP (lower is better) and IPC values of the various DRAM organizations show similar trends. The relative order of the ROI values is preserved. Because the CHARM DRAM devices improve the performance of the applications and consume less activate and precharge energy, the absolute ROI value in EDP is higher than that in IPC on the same configuration. On CHARM[×2,/4]A, the EDP is improved by an average of 7.6% and

18% on all the SPEC applications and the spec-high applications, respectively.

3.3.2 Performance Impact on Multiprogrammed Workloads



(a) weighted speedup with 8 memory controllers



(b) weighted speedup of Scheme 1

Figure 3.9: The absolute and relative weighted speedups of spec-high, spec-med, and spec-blend on the systems using CHARM[×2,/4] and CHARM[×4,/4] with (a) 8 memory controllers. We vary the number of controllers and present the weight speedups of Scheme 1 in (b).

The CHARM organization also improves the performance of systems running multiprogrammed workloads. Figure 3.9 shows the weighted speedups [44] of the three mixtures described in Section 3.2 on systems using CHARM[×2,/4] and CHARM[×4,/4]. We do not present the spec-low results because the aggregate main-memory bandwidth from the applications is too low to make a noticeable difference in the weighted speedups. We test three interesting memory-provisioning schemes for each mixture. These are characterized as follows: 1) each application is provisioned with the same amount of memory space, which equals the maximum footprint of the application in the mixture, with a quarter of that allocated to CHARM (Scheme 1); 2) each application is provisioned with the same footprint, which equals the average footprint of all applications in the mixture, with a quarter of that allocated to CHARM (Scheme 2); and 3) a quarter of the memory footprint of each application is allocated to CHARM (Scheme 3). We use the weighted speedups of the AR×1 configuration as references and apply the access-frequency-aware memory allocation scheme discussed in Section 3.1.5.

We make the following observations from the experiment. First, spec-high benefits more (i.e., better relative weighted speedups) from the CHARM DRAM devices compared to the other mixtures because spec-high applications are more sensitive to the main-memory access latency than spec-med and spec-blend applications. However, spec-high yields the smallest absolute weighted speedup values, as memory requests from the applications

often contend with each other in the memory controllers and because each application experiences a higher average memory access time than when executed in isolation. The weighted speedup values are improved by 7.1% and 12% for spec-high on CHARM[×2,/4] and CHARM[×4,/4], respectively.

Second, among the three memory-provisioning schemes, the one that assigns the maximum memory footprint among all of the applications in the mixture to each application (Scheme 1) provides the greatest weighted speedup improvement, as a larger memory footprint is allocated to the center HAR mats in Scheme 1 compared to Schemes 2 and 3. Nonetheless, all the scenarios provide noticeable improvements in the weighted speedup and show similar trends.

Third, the relative weighted speedup improves as more main-memory bandwidth is provided in the system for mixtures with bandwidth-demanding applications. We change the number of memory controllers and present the weighted speedups of Scheme 1 in Figure 3.9(b). Because memory requests are almost always piled up in the request queues of the spec-high memory controllers, the bandwidth of the main-memory system is the most prominent factor affecting system performance. As a result, both the absolute and relative weighted speedups of spec-high are more sensitive to the main-memory bandwidth than the others. These findings demonstrate the effectiveness of CHARM for multiprogrammed workloads.

3.3.3 Performance Impact on Multithreaded Workloads

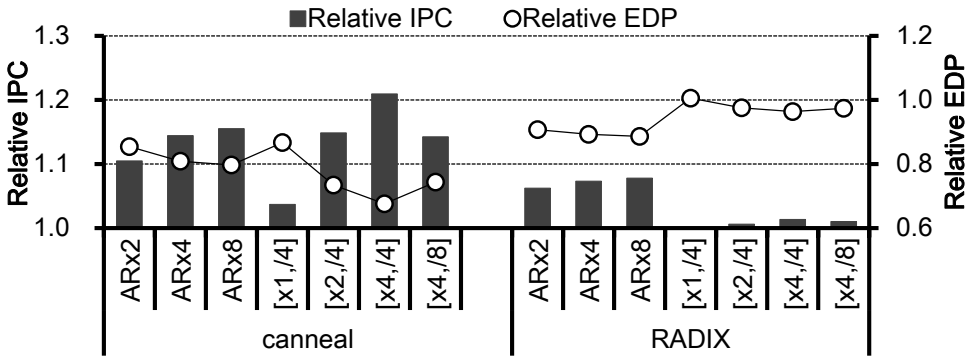


Figure 3.10: The relative IPC and EDP of the various DRAM organizations on canneal and RADIX applications. AR×1 is the reference organization.

The CHARM DRAM devices also improve the performance and energy efficiency of the simulated system on multithreaded applications, while the degree of improvement depends on the characteristics of the applications. Similar to the single-threaded applications, multithreaded applications with low L2 cache MPKI are mostly insensitive to the DRAM organizations. Among the ones with higher L2 cache MPKI, we present the relative IPC and EDP of canneal from PARSEC and RADIX from SPLASH-2 in Figure 3.10. Again, AR×1 is the reference DRAM organization. RADIX is an integer radix sort application that randomly accesses main memory, while canneal has memory regions that are accessed more frequently. Therefore, we use the access frequency aware memory mapping only for canneal.

The performance of canneal is substantially improved both by increasing the aspect ratio of the mats and by decreasing the read to data delay (tAA) on the center blocks, but the latter has higher influence. AR×2, AR×4, and AR×8 improve the IPC of canneal by 10%, 14%, and 15%, respectively. Exploiting the low tAA of the center blocks enables more of a speedup; CHARM[×2,/4] and CHARM[×4,/4] increase the IPC by 15% and 21%, respectively, as the center blocks, which correspond to only 25% of the DRAM capacity, service 89% of main-memory accesses (Figure 3.6(b)) with the access frequency aware memory allocation scheme. Note that CHARM[×4,/8] increases the IPC by 14% compared to the reference organization, which is slightly worse than CHARM[×2,/4]. This occurs because a significant portion of the frequently accessed memory footprints are not allocated to the center HAR mats due to their limited capacity, which negates the benefits of lower tAA on the HAR mats.

The reduction in the DRAM cycle time is the most influential factor for RADIX. As opposed to the case of canneal, CHARM[×4,/4] increases the IPC only by 1.6%. AR×2, AR×4, and AR×8 improve the IPC by 6.2%, 7.3%, and 7.8%, respectively. Because the capacity of the center HAR mats is only one eighth and one fourth of the total capacity for CHARM[×4,/8] and CHARM[×2,/4], the performance gains are 0.6% and 1.5%, respectively.

Chapter 4

SALAD: Achieving Symmetric Access Latency with Asymmetric DRAM Architecture

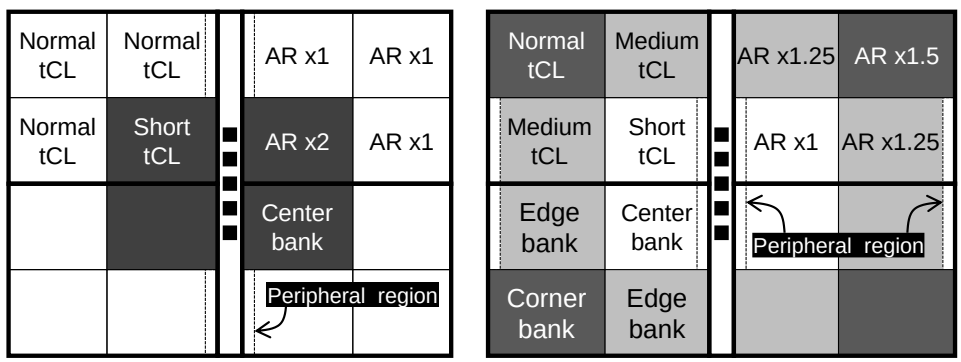
4.1 Symmetric Access Latency with Asymmetric DRAM Architecture

To address the aforementioned limitations of the DRAM device with non-uniform access time, we propose a novel DRAM architecture. Similar to CHARM, we leverage the ideas of 1) non-uniform bank accesses [21] to reduce the read/write latency of center banks and 2) increasing the aspect ratio (AR) of a portion of DRAM arrays to reduce their activate/precharge time. However, unlike CHARM, we apply HAR mats to edge banks, not center banks which have lower transmission latency (tCL) due to their shorter topological distance to the I/O pads. CHARM reduces the unbuffered access latency

---

This chapter is based on [2]. - © [2016] IEEE. Reprinted, with permision, from Computer Architecture Letters.

(tAC) significantly, but only for a few center banks per device. In contrast, our proposal lowers tAC of all the banks as higher tCL at the edge and corner banks is compensated by the lower tRCD of HAR mats. We call this new DRAM architecture SALAD (Symmetric Access Latency with Asymmetric DRAM).



(a) CHARM (b) SALAD

Figure 4.1: Organizations of (a) CHARM and (b) SALAD

We design SALAD to have three bank groups based on the distance from the I/O pads, as shown in Figure 4.1(b). We apply SALAD to the baseline DDR4-2400 device, which has 16 banks, throughout this dissertation. The four center banks have mats with the default AR (512→512 cells per mat), the eight edge banks have mats with a higher AR to compensate longer transmission time of inter-bank command/address/data signals. The AR of the mats in the four corner banks is even higher. As the ARs of the mats in the edge and corner banks increase, the activate/ precharge times of these banks decrease further at the cost of higher DRAM area overhead. Table 4.1 tabulates the energy and timing parameters of

the proposed SALAD architecture with area overheads of 3% and 6% (modeling methodology detailed in Section 4.2). Due to asymmetry in the ARs of mats across banks, the size of a bank depends on which group it belongs to. The resulting spare area is filled with peripheral resources, such as I/O pads, decoupling capacitors, and charge-pump circuitry, similar to CHARM.

SALAD effectively improves the performance of applications even if hot pages do not exist or are not identified. When memory accesses are evenly spread across all the banks in the system and have low spatial locality, a memory access commonly results in a row buffer miss, which is only exacerbated by frequent DRAM refreshes. In this case, the average unbuffered latency of SALAD with 3% and 6% area overheads is 22ns and 20ns, respectively, whereas that of CHARM is 25ns and 24ns. For a given area overhead SALAD outperforms CHARM in terms of average memory latency. When hot pages exist in an application and the annotation/profiling techniques successfully map those pages to center DRAM banks, CHARM provides lower latency than SALAD for the pages. However, this latency gap narrows if the application has ample spatial locality in memory accesses because both CHARM and SALAD have the same latency (tCL) for any row-buffer hits. SALAD provides lower latency to the remaining, less frequently accessed pages than CHARM, providing more robust performance without counting on the quality of hot page-aware allocation.

Minimal changes are needed to a memory controller to support

SALAD. Similar to [65], [1], SALAD has multiple types of read commands with different tCL values (at least three types as we assume three bank groups1). A bank group can accept read commands with tCL values equal to or higher than its intrinsic column access latency value, which is determined by the distance between the bank and the I/O pads of a device. When it receives a command with a latency higher than its intrinsic value, the command/address signals are latched at buffers close to the bank to delay the read process accordingly. Storing command and address values are more cost-effective than latching data because the latter requires storing more bits. This DRAM-side latching alleviates the complexity of the DRAM access scheduler to support SALAD. When all the banks have the single uniform tCL value (e.g., conventional DRAM devices), no conflict in data bus exists due to read commands. With multiple tCL values, a read command scheduled at a certain cycle may occupy the data bus at the same time as a previously issued read command when the latter has higher tCL values. Using the 'delayed' read command, the memory controller can avoid this conflict without rescheduling the read operation at the following cycles.

4.2 Experimental Setup and SPICE Modeling

We model a chip-multiprocessor system to evaluate the performance and energy efficiency of SALAD. The system consists of 16 out-of-order cores and 4 memory channels, where a core

operates at 3.6GHz, issues/commits up to 4 instructions per cycle, and has 64 reorder buffers. Each core is connected to separate L1-I and L1-D caches ($s) and a unified L2 $, all with 64B $ lines and 4 banks. The capacity and set associativity of an L1 and L2 $ are (16KB, 4) and (1MB, 16), respectively. A MOESI protocol and a linear hardware prefetcher [63] that detects/prefetches streams of consecutive memory accesses in address are used. Each channel consists of 4 ranks of DDR4-2400 modules, where each rank consists of 18 8GB ÷4 DRAM devices (2 for ECC). The memory controller (MC) of each channel has 64 request queue entries and adopts the parallelism-aware batch scheduling [35] and minimalist open-page management policy [19]. To support the multi-tier latency values of SALAD/CHARM, the MC initiates different counter values depending on the bank group that an ACT/PRE command falls on. It schedules a RD/WR command with the minimal available latency that satisfies the minimal tCL of the corresponding bank group and avoids conflicts in the data I/O by the 'delayed' read commands. We modify McPAT [27] for CPU modeling at the 14nm technology node and McSimA+ [1] for performance simulation.

SPEC CPU2006 [15] benchmark suite is used for single- and multiprogrammed workloads. We identify and use the most representative simulation point of each application using Simpoint [43], which consists of 100M instructions. We measure the memory accesses per kilo-instructions for the SPEC applications to classify the eight most memory-intensive applications as spec-high: mcf, milc, leslie3d, soplex, GemsFDTD, libquantum, lbm, and omnetpp.

	Refer ence	All-HAR		CHARM			
Area overhead	0%	3%	6%	3%	6%	3%	6%
Aspect Ratio (AR)	1.0	1.3	1.7	2.0	4.0	1.0	1.0
tRCD [ns]	14	12	10	9	8	14	14
tCL [ns]	14	14	14	8	8	14	14
tRAS [ns]	35	29	24	20	15	35	35
tRP [ns]	14	13	12	11	9	14	14
ACT+PRE energy [nJ]	63.5	54.4	46.7	40.2	31.1	63.5	63.5
RD/WR energy [nJ]	20.8	21.0	21.2	15.4	15.5	21.0	21.2
	Refer ence	SALAD					
Area overhead	0%	3%	6%	3%	6%	3%	6%
Aspect Ratio (AR)	1.0	1.5	2.0	1.25	1.5	1.0	1.0
tRCD [ns]	14	10	9	12	10	14	14
tCL [ns]	14	14	14	11	11	8	8
tRAS [ns]	35	26	20	30	26	35	35
tRP [ns]	14	12	11	13	12	14	14
ACT+PRE energy [nJ]	63.5	49.2	40.2	55.7	49.2	63.5	63.5
RD/WR energy [nJ]	20.8	21.0	21.2	18.2	18.3	15.1	15.2

Table 4.1: Energy and timing parameters of the baseline DRAM, All-HAR, CHARM, and SALAD organizations.

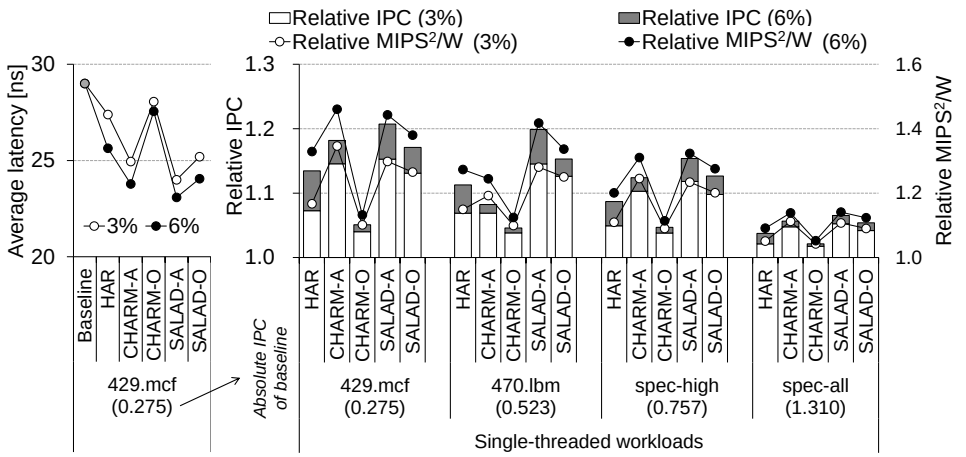
We use two multiprogrammed workloads called mix-high and mix-blend, where the former is composed of two instances of the spec-high applications and the latter of an instance of perlbench, bzip2, gobmk, dealII, bwaves, zeusmp, sjeng, h264ref, astar, xalancbmk, mcf, milc, GemsFDTD, lbm, omnetpp, and sphinx3. MICA [64] (a key-value store), PARSEC [10] and SPLASH-2 [50] are used for multithreaded workloads.

Table 4.1 summarizes the latency and energy values of the baseline DDR4, All-HAR, CHARM, and SALAD devices obtained from SPICE simulation. A modified PTM low-power model [57] assuming a 22nm, 3-metal layer process is used. We size transistors to satisfy the baseline DDR4-2400 timing specifications [62].We set target area overheads over the baseline as 3% and 6%, the representative design points with high performance gains on CHARM. All-HAR increases the aspect ratio (AR) of all the mats in a DRAM device, similar to RLDRAM [34], with ARs of →1.3 and →1.7, which correspond to overall area overheads of 3% and 6%, respectively. Increasing the AR of the mats at the 4 center banks out of 16 banks per device by 2 and 4 times leads to 3% and 6% area overheads, which is consistent with the results in [1]. For SALAD, banks at the (center, edge, corner) groups have the mats with ARs of (→1.0, →1.25, →1.5) and (→1.0, →1.5, →2.0) to have 3% and 6% area overheads, respectively. Similar to the baseline and All-HAR, SALAD has the unbuffered access latency (tAC) mostly the same over an entire device, whereas CHARM has huge difference in tAC between the center and edge/corner banks. For a

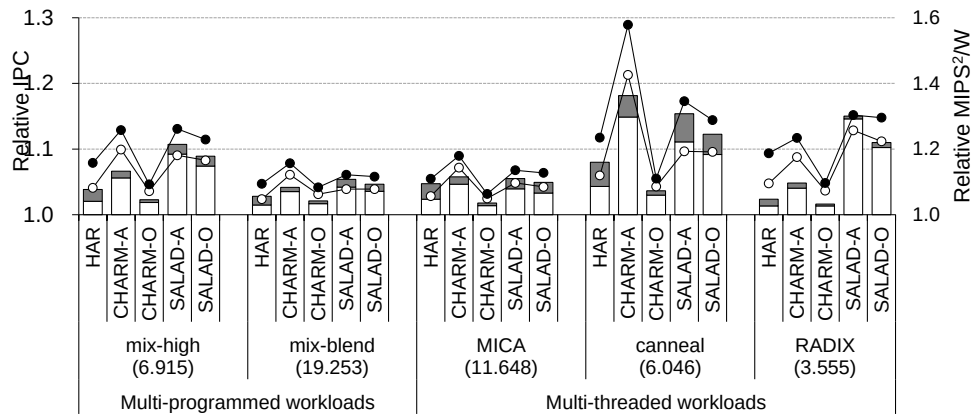
given area overhead, the access latency of SALAD is lower than that of All-HAR, demonstrating the benefits of exploiting non-uniform accesses across banks and HAR mats. Due to decreased BL capacitance, banks with higher-AR mats consume lower energy per ACT/PRE. To keep the delay of inter-bank communication constant even for bigger devices, faster transistors are used, increasing RD/WR energy for all banks in All-HAR and for corner banks in SALAD and CHARM.

4.3 Evaluation

We quantify the system-level performance and energy efficiency benefits of SALAD over previous low-latency DRAM architectures using various workloads. Figure 4.2 shows the performance (IPC) and system-level energy efficiency (MIPS2/W) of the three DRAM organizations-HAR on all mats (All-HAR), CHARM, and SALAD-with 3% and 6% area overheads over the baseline DDR4-2400 device, using single-threaded and multiprogrammed SPEC CPU2006 [15] workloads, and multithreaded MICA [64], SPLASH2 [50] and PARSEC [10] benchmark applications. We apply two page mapping schemes for SALAD and CHARM. To obtain the results of SALAD-A and CHARM-A, frequently accessed (hot) pages are identified via profiling and mapped to fast banks. In contrast, SALAD-O and CHARM-O adopt a hot page-oblivious mapping. We present the aggregate IPC values for multiprogrammed workloads.



(a) Single−threaded workloads



(b) Multiprogrammed and multithreaded workloads

Figure 4.2: Average read latency, performance (IPC) and MIPS2/W of DRAM organizations with HAR on all mats, CHARM, and SALAD with 3% and 6% area penalty compared to the baseline DDR4−2400 device on single−threaded and multiprogrammed SPEC CPU2006, and multithreaded MICA, SPLASH2, and PARSEC benchmark applications.

A single memory channel is populated for single-threaded workloads to stress the main memory system.

We make the following key observations. First, SALAD with hot page-oblivious mapping performs similar to CHARM with hot page-aware mapping. With 3% area overhead, SALADO and CHARM-A both improve the IPC by 10% on average for spec-high applications. Moreover, the energy efficiency (MIPS2/W) is improved over the baseline DDR4 configuration by 20% and 25%, respectively. CHARM-A is effective for applications with uneven memory access frequency over pages (e.g., 429.mcf), but not for applications with evenly distributed access patterns (e.g., 470.lbm). In contrast, SALAD is appealing as it provides robust performance even without careful, criticality-aware page allocation, while CHARM-O improves the IPC of spec-high applications only by 4% on average.

Second, hot-page aware mapping further improves the performance of SALAD so that SALAD-A outperformed CHARMA by 2% and 4% with 3% and 6% area overheads, respectively, for spec-high applications. Because SALAD has three bank groups, whereas CHARM has only two, the edge banks in SALAD have a lower tCL than CHARM. When an application has many hot pages (e.g. 429.mcf) whose aggregate size exceeds the capacity of the center banks, SALAD can achieve further latency reduction (4% with 3% area overheads) over CHARM as SALAD has uniformly low access latency over the whole device. Finally, increasing the aspect ratio of all the mats of a device (All-HAR) is not as

effective as SALAD. With 3% area overhead, All-HAR improves the IPC of spec-high and spec-all (i.e., the average over all SPEC CPU2006 applications) only by 5% and 2%, respectively. This is because All-HAR has a single tCL value, which is determined by the topological distance between I/O pads and the farthest bank of the device.

SALAD is also effective for multiprogrammed and multithreaded workloads. SALAD-O improves the IPC of mix-high and mix-blend workloads by 7% and 4% with 3% area overhead and by 9% and 5% with 6% area overhead. It also achieves 22% and 19% better MIPS2/W over the baseline DDR4 for RADIX of SPLASH-2 and canneal of PARSEC.

Chapter 5

Row-Buffer Decoupling: A Case for Low-Latency DRAM Microarchitecture

5.1 Row-Buffer Decoupling

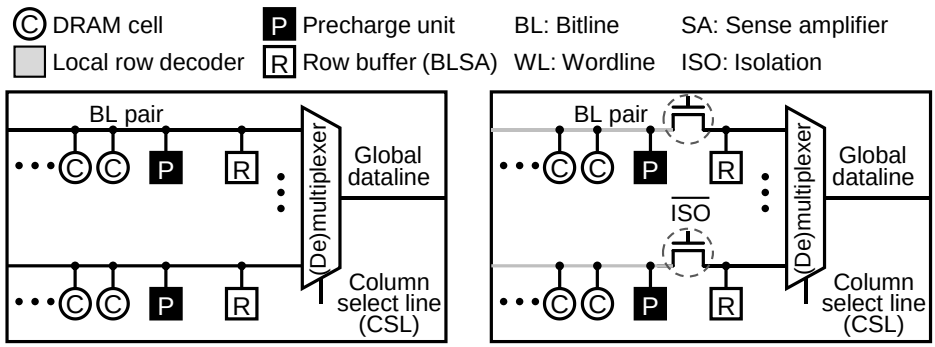
Instead of lowering the frequency or improving the accuracy of prediction in managing row buffers, we propose a new DRAM architecture that eliminates the need for prediction. We identify that the performance difference between the open- and close-page policies originates from the fact that the row buffers lose data when a precharge process starts because the sense amplifiers that work as row buffers, DRAM cells that store data, and the bitlines (BLs) in between are strongly coupled. We propose modifying the internal DRAM structure to break the dependency between most of the precharge process and data retention by decoupling the row buffers

---

This chapter is based on [3]. - © [2014] IEEE. Reprinted, with permission, from ISCA' 14.

from the BLs and the cells. In this section, we first review the pertinent details of the DRAM architecture, propose row-buffer decoupling, and explore more internal DRAM μ-operation scheduling policies enabled by decoupling, which enhances the performance and energy efficiency of main memory systems more than state-of-the-art row-buffer management policies.

5.1.1 Pertinent Details of DRAM Architecture to Explain Why Row Buffers are Coupled



(a) Datapath details (conventional)

(b) Datapath details (decoupled)

Figure 5.1: (a) the datapath details of a mat for conventional DRAM devices, and (b) the datapath details of a mat with bitline sense amplifiers (row buffers) and (de)multiplexers decoupled from the bitlines and the DRAM cells by isolation transistors.

A contemporary DRAM device employs a hierarchical structure to balance latency, bandwidth, energy consumption, and capacity under tight cost constraints. A DRAM device consists of multiple banks (e.g., 8 for DDR3), each of which is constituted with thousands of

DRAM mats arranged in a two-dimensional fashion. A DRAM mat, a unit of DRAM devices, consists of hundreds of rows and columns of cells (e.g., 512x512 bits), in which the rows and columns are connected to local wordlines (WLs) and BLs, respectively. A DRAM cell consists of an access transistor and a capacitor, which stores a bit of data by the voltage developed at the capacitor (i.e. Vdd means one and ground means zero, typically). Because the capacitance of a DRAM cell is much smaller than that of the connected BL, few BLs share a bitline sense amplifier (BLSA) to detect the value stored in the cells. Compared to the number of activated cells per mat, much fewer cells are read or written per request. Therefore, each mat has local datalines, which work as a datapath of the multiplexers/demultiplexers between BLSAs and global datalines. A global dataline is shared by all the mats in the same column of the bank.

With an ACT command from a memory controller, the global row decoder of a bank decodes the row address coincident with the ACT command and drives signals to the local row decoders of the corresponding mats through the global WLs. Subsequently, the local row decoders of the mats drive the local WLs coincident with the row of the bank to activate the row. Because very small cells (6F2 for modern devices [79]) are employed to achieve a high integration density (i.e., low cost per bit), the developed BL voltage after the local WLs are driven is also very small, and cell states are destroyed as a consequence. Hence, a BLSA is used to detect, restore, and latch a value stored in an accessed cell, i.e., opening

the corresponding DRAM row (page); a group of BLSAs that are involved in an ACT command is called a row buffer.

After an ACT command is sent, it takes tRCD for the column address coincident with a read (RD) or write (WR) command to be interpreted by the column decoder and delivered through the column select lines to control the multiplexers and demultiplexers. When the column address arrives at the specific mats through the column select lines, the BLSAs have not fully developed the BL voltage. Nonetheless, their values are large enough to be delivered through the local and global datalines for reads; after a RD command is sent, it takes tAA for the first chunk of the data (e.g., 8 bits for a 8 device) to be available for transmission through a memory channel. BLSAs keep developing a voltage difference and restores the values back to the connected DRAM cells that have lost values; it takes tRAS after an ACT command to complete the activate and restore processes.

Precharge units precharge BLs and BLSAs to half the Vdd, the average of Vdd (one) or ground (zero) stored in a DRAM cell, to prepare for the next activate. This process is initiated by a PRE command or following a read or write process in the form of an auto precharge (for the close-page policy). In the current DRAM microarchitecture, the precharge units are all connected (or strongly coupled) with BLs, connectors to (de)multiplexers, and BLSAs. Therefore, the following critical DRAM operations to read data from or write those to a row, which has not been activated yet, must be serialized: (i) the data of the currently active row are

restored to the corresponding cells, (ii) all the BLs and BLSAs are precharged, and (iii) the new row is activated. Note that the precharge process takes considerable time (tRP is 11ns to 15ns for DDR3/4 [72, 41] devices) while destroying the data stored in the BLSAs (i.e., the row buffer). Consequently, when to precharge significantly affects the latency of servicing subsequent memory requests in conventional DRAM devices.

5.1.2 DRAM Architecture with Decoupled Row Buffers

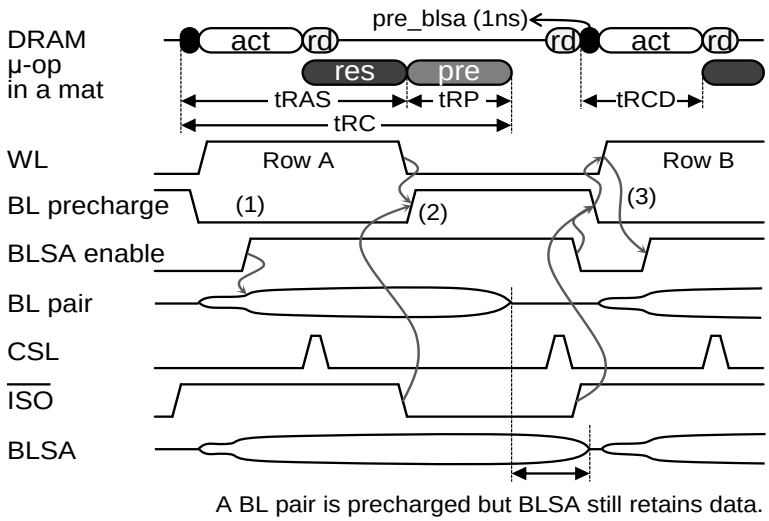


Figure 5.2: The timing diagram illustrating how the control and datapath signals within a mat supporting row-buffer decoupling are changed to process DRAM commands that activate a row, read it twice, and activate another row.

We modify the DRAM mat microarchitecture to take the advantages of both the open-page and close-page management policies by breaking the dependency or coupling between data retention in the

BLSAs and the BL precharge. We achieve this decoupling by adding isolation transistors to the BL pair connected to a BLSA as shown in Figure 5.1(a). When the isolation transistors are on, a DRAM mat can operate in the same way as a conventional one for reading and restoring the cell value. When they are off, the data in the BLSAs can be read or updated by the global datalines through the (de)multiplexers while the BLs can be precharged in preparation for a following activate at the same time.

The isolation transistors are controlled in such a way that the precharge time observed during a row-buffer miss is minimized as shown in Figure 5.2. Both the access transistor specified by a local WL driver and the isolation transistors are turned on such that the specified DRAM cell, the BL pair, and the BLSA are all connected during the activate and restore processes ((1) in Figure 5.2). After the data is restored to the cell, both the access transistor and the isolation transistors are turned off. This allows both the cell and the BLSA to retain the data of an open row while the BLs are precharged (2). Because the capacitance of a BL is an order of magnitude higher than that of a BLSA, the time to precharge the BLs dominates the tRP of a conventional DRAM mat. Therefore, only a small portion of the tRP of a conventional DRAM mat is needed to precharge the BLSA when a new row is activated by disabling the BLSA and turning on the isolation transistor. Then, the precharge unit is disabled and the access transistor corresponding to the new row is turned on initiating a new activate process, which is followed by enabling the BLSA after charge sharing between the

BL and the new cell develops a voltage difference which is enough for sensing (3).

This early-precharge policy enabled by row-buffer decoupling does not save the read latency if the DRAM cycle time (tRC) becomes the limiting factor because the activate, restore, and precharge processes are still needed per active row switch. However, a memory controller often has multiple ranks, each of which consists of several banks in modern main memory systems, and there are often multiple accesses per open DRAM row. Therefore, the average interval between two activates on a bank would be much longer than the tRC on many applications, where row-buffer decoupling does reduce access latency for row-buffer misses. Because the time to precharge the BLSAs is so small, we do not dedicate a DRAM command for it and instead, we include it as part of an activate process.

Decoupling the row buffers using the isolation transistors is feasible and their area overhead is minimal. We leverage the concept of using isolation transistors, which reduce the delay and power consumption, from SRAM designs [80]. In SRAM devices, the isolation transistors disconnect a BLSA from a pair of complementary BLs right after a specified SRAM cell develops just enough voltage for the BLSA to sense a stored value. This in turn prevents the pair of BL from swinging rail-to-rail, reducing unnecessary power consumption for the next precharge cycle. Furthermore, because the BLSA is disconnected from the BL exhibiting a large capacitance, it can swing to a desired voltage

level faster than the BLSA that is still connected to the BLs even after a sufficient voltage difference is developed. Although isolation transistors have been used for DRAM devices before, their purposes was simply to make multiple BLs share a BLSA to lower the area overhead of bulky BLSAs [69] and they have not been exploited for page management like in our proposal. Tiered-latency DRAM [26] also used isolation transistors, but those were used to provide asymmetry in accessing DRAM rows, unlike our row-buffer decoupling. We set the size of an isolation transistor pair as one tenth of the size of a BLSA, as suggested by Gealow [69] and in tiered-latency DRAM [26], which is large enough to be turned on and off within few tenths of a nanosecond. Therefore, the area overhead from adding the isolation transistors and their supporting circuitry is low at 0.8% for the DRAM process we specify in Section 3.4.

The energy overhead of the isolation transistors is also insignificant. An isolation transistor is turned on and off once per active row switch, like the access transistor of a cell, a precharge unit, and a BLSA. The driver for the isolation transistors is much smaller than the driver for the BLSAs. Because the DRAM device gets larger, the lengths of the global and inter-bank datalines increase, and their driver sizes are adjusted to maintain the data transfer latency, which is properly considered during the energy modeling in Section 5.1.4.

So far, we have focused on DRAM reads, where data restored in DRAM cells need not be updated. In contrast, a DRAM write

updates the data latched in the BLSAs using write drivers. Because the DRAM cells are disconnected from the BLSAs during the write in the case of the early-precharge policy, the cells must be reconnected to the BLSAs, updated (restored), and disconnected again, after which the BLs can be precharged to reduce the active row switch time. Even though the energy to update the cells during a write is lower than the energy consumed during a restore process, the update and precharge energy is not negligible and clearly an overhead. One way to alleviate this overhead is to keep the cells and the BLSAs connected once a WR command is applied to an open row. Then, the active row switch time would be the same as the case of the open-page policy, but because writes are less frequent than reads for many applications [68], their impact on average memory access latency could be limited. Another way to alleviate the energy overhead is to make the memory controller detect consecutive writes on an open row. Then, all but the last write can skip the update and precharge processes. We elaborate on these extensions in the following section.

5.1.3 Scheduling Internal DRAM μ-operations

Row-buffer decoupling not only enables early precharges but also provides more freedom in controlling the timings of internal DRAM μ-operations. A DRAM command prompts one or more μ-operations within the specified DRAM devices, which is similar to μ-ops in CPUs. For conventional DRAM devices, an ACT command

corresponds to latching data of the specified row at the BLSAs (a μ-operation called act) followed by restoring the data back to the original cells (res). A WR command overwrites a portion of data at the BLSAs (wr) then updates the overwritten values to the corresponding DRAM cells (res). A PRE command precharges both BLs (pre_bl) and BLSAs (pre_blsa). A RD command is not further divided and corresponds to a rd μ-operation. We use capital letters for the explicit DRAM commands and lower-case letters for the μ-operations. With the isolation transistors, pre_bl can be a part of ACT and WR to save a row-buffer miss latency discussed in Section 5.1.2 for the early-precharge policy. In contrast, if we enforce an ACT not to do res and delay it until act fully develops BLSAs, which is enabled by the isolation transistors, we can reduce the res energy. This is because BLSAs are more power efficient when used as drivers instead of performing sensing and driving at the same time. In addition, if an entire row gets overwritten after being activated or the row is read but not accessed any more, res can be omitted.

The design space of scheduling the DRAM μ-operations is huge, and the performance and energy efficiency of the system can vary significantly depending on the interaction between scheduling policies and memory access patterns. However, the early-precharge policy performs best among the possible scheduling options because delaying res or pre might save energy but does not reduce the latency for row-buffer misses. Therefore, if an application is less sensitive to main memory latency, a delayed-

restore policy could improve the system-level energy efficiency with negligible degradation in performance. Even though it could be interesting to devise adaptive schemes for scheduling the μ-operations, such as adopting reinforcement learning [71, 78], it is beyond the scope of this dissertation and will be considered in possible future work. Table 5.1 enumerates the mapping between DRAM commands and the corresponding μ-operations for several representative policies. For both ACT and WR, we have three options: 1) neither res nor pre_bl (Dres), 2) only res after act or wr (Eres), and 3) res then pre_bl (Epre). We use the notation [policy for ACT, policy for WR] or just policy in case ACT and WR have the same policy. If either ACT or WR requires pre_bl, ACT should start with pre_blsa to disable the BLSAs before the normal activation process begins. The PRE command can be omitted for [Epre, *] if there is no WR after the row gets activated. Epre does not need PRE at all. Eres operates the same as the open-page policy of conventional DRAM devices. On top of these policies, we also examine the option of populating two kinds of ACT and WR, one with res and the other without it, and letting the memory controller use them properly to precharge the BLs only once when there are multiple writes to an open bank in the request queue for Epre. We call this new policy an augmented early-precharge scheme or a-Epre. In Section 5.3, we compare the performance of Dres, Epre, and a-Epre.

CMD	μ-op	Dres	Eres (Ref)	[Epre, Dres]	[Epre, Dres]	[Epre, Dres]
ACT	pre_blsa			√	√	√
	act	√	√	√	√	√
	res		√	√	√	√
	pre_bl			√	√	√
WR	wr	√	√	√	√	√
	res		√		√	√
	pre_bl					√
PRE (no WR after ACT)	res	√				
	pre_bl	√	√	N/A	N/A	N/A
	pre_blsa	√	√			
PRE (1+WR after ACT)	res	√		√	√	
	pre_bl	√	√	√	√	N/A
	pre_blsa	√	√			

Table 5.1: Mapping between DRAM commands and the matching μ-operations for several scheduling policies enabled by row-buffer decoupling. √ indicates that the μ-op is the part of the DRAM command in the same row for a certain policy.

5.1.4 Quantifying the Row-Buffer Decoupling Overhead through SPICE Modeling

To quantify the overheads and savings of row-buffer decoupling, we modeled the power and timing of internal DRAM components and operations using SPICE simulation, and we obtained the inter-die I/O power values of the processor-memory interfaces from the

latest DDR3 specifications [41]. We assumed the following: 28nm DRAM process, 3 metal layers, 8Gb and 8 banks per device, 8KB per row, 16GB/s per channel, and 6F2 DRAM cells for the SPICE modeling with a modified version of the PTM low-power model [57]. Row-buffer decoupling slightly increases the DRAM die area (by 0.8%), which in turn increases the on-chip datapath length. We increased the driver strength to keep the read and write latency same as the reference device and applied the corresponding energy increase during DRAM reads and writes.

Parameter	Reference	Delayed restore	Early precharge
tRCD	14ns	14ns	15ns
tAA	14ns	14ns	14ns
tRC	49ns	54ns	50ns
tWR	15ns	10ns	29ns
tRP	14ns	19ns	N/A
E_{ACT}	42.5nJ	14.1nJ	50.1nJ
E_{RD}	7.0nJ	7.1nJ	7.1nJ
E_{WR}	7.0nJ	7.0nJ	32.2nJ
E_{PRE}	7.5nJ	12.5nJ	N/A

Table 5.2: Latency and energy parameters of the primary DRAM operations for the rank without decoupling (reference) and the rank with decoupled row buffers employing the delayed-restore (Dres) and early-precharge (Epre) policies.

The acquired latency and the energy consumption values of major DRAM operations for the reference (without decoupling), Dres, and Epre ranks are summarized in Table 5.2. The read time (tAA) is the same for all the DRAM types. Dres has a higher cycle time (tRC) than that of the others because res is the part of a PRE command. tRCD is 15ns for Epre and 14ns for others due to the overhead from disabling and precharging the BLSAs. The write recovery time (tWR) is different because Dres dispenses with res while Epre entails both res and pre for a write. The precharge time (tRP), is 14ns for the reference rank and 19ns for Dres because res is needed. Similar trends are observed for energy consumption. Epre consumes more energy for WR than other types because it does both res and pre. The sum of the ACT and PRE energy is the lowest for Dres because BLSAs have lower load capacitance during sensing and work as drivers during restore as explained in Section 5.1.2.

5.2 Experimental Setup

We simulated a chip-multiprocessor system with multiple memory controllers to evaluate the system-level impact of row-buffer decoupling with the default parameters summarized in Table 5.3. Memory addresses were interleaved in the following way: DRAM row, bank, rank, memory channel, and column from the most significant address bit to the least significant bit (so called page interleaving) by default. Other interleaving schemes are compared

in Section 5.3.2. We modified McSimA+ [5] to support row-buffer decoupling for performance simulation. McPAT [27] was used for modeling the power and timing values of the cores, caches, and memory controllers assuming a 22nm logic process.

Group	Value
Number of cores	16
Number of memory controllers	4
Coherence policy	MESI
Coherence implementation	Reverse directory
Frequency	3GHz
Issue policy	Out-of-Order
Issue/commit width	4/4
L1 I/D cache size/associativity	16KB/4
L2 cache size/associativity	1MB/16
L1, L2 cache line size	64B
Number of channels per controller	1
Bandwidth per channel	16GB/s
Number of ranks per channel	2
Capacity per rank	8GB
ECC	SECDED
Scheduling policy	PAR-BS [35]
Request queue size	32
DRAM refresh interval (tREFI)	7.8 us

Table 5.3: Default parameters of the simulated multicore system.

The SPEC CPU2006 [15], SPLASH-2 [50], and PARSEC [10] benchmark suites were used for evaluation. We identified the representative phases for SPEC CPU2006 applications using Simpoint [43]. Per application, we chose the top-4 slices in weight, each of which includes 100 million instructions. As for SPLASH-2 and PARSEC, we simulated regions of interest and used the datasets listed in [27]. We classified the SPEC CPU2006 applications into three groups (spec-low, spec-med, and spec-high) based on the main memory accesses per kilo-instructions (MAPKI) [15]. Table 5.4 shows four mixtures we created as multiprogrammed workloads. mix-high is composed of spec-high applications; mix-close consists of the applications with relatively high row-buffer conflict rates favoring the close-page policy; mix-open consists of the ones with low row-buffer conflict rates, and mix-blend is composed of four applications from spec-low, six from spec-med, and six from spec-high. For each mixture, each core runs a simulation point, and the number of simulation points used per application is specified in Table 5.4. A weighted speedup approach [44] was used to evaluate the performance of the multiprogrammed workloads. To compute the weighted speedup, we first determined the relative performance of each application and the ratio of its IPC in a configuration for a multiprogrammed workload to the IPC in a standalone execution configuration, and accumulated the relative performance of all the applications belonging to the mixture.

	400.perlbench	401.bzip2	416.gamess	435.gromacs	444.namd	445.gobmk	447.dealII	453.povray	454.calculix	456.hmmer	403.gcc	410.bwaves	434.zeusmp	436.cactusADM	458.sjeng
Mixes	spec-low											spec-med			
mix-high															
mix-open				1	1			1	1		1				1
mix-close			2							1			2	2	1
mix-blend	1	1				1	1					1	1		1

	464.h264ref	465.tonto	473.astar	481.wrf	483.xalancbmk	429.mcf	433.milc	437.leslie3d	450.soplex	459.GemsFDT	462.libquantum	470.lbm	471.omnetpp	482.sphinx3
Mixes	spec-med					spec-high								
mix-high						2	2	2	2	2	2	2	1	1
mix-open		1		1			2	1	2		1			2
mix-close			2			2				2			2	
mix-blend	1		1		1	1	1			1		1	1	

Table 5.4: We categorize the SPEC CPU2006 applications into 3 groups based on the main memory accesses per kilo instructions (MAPKIs). The numbers in the table mean the number of cores that run the specific applications for a certain mixture of multiprogrammed workloads.

5.3 Evaluation

We evaluated the system-level impacts of row-buffer decoupling on the simulated multicore system. We first used single-threaded workloads to quantify the benefits of employing decoupled row buffers with an average main memory read latency, IPC (instructions per cycle), and MIPS2/W (square of million instructions per second per Watt). Then, we populated multiple memory channels and cores to quantify the effectiveness of row-buffer decoupling on multiprogrammed and multithreaded workloads and to demonstrate its robustness over various system configurations.

5.3.1 The Impact of Row-Buffer Decoupling on Single-Threaded Workloads

Figure 5.3 shows the average memory read latency, relative IPC, and relative MIPS2/W values of the SPEC CPU2006 applications for systems with and without decoupled row buffers. The average values for all the SPEC CPU2006 benchmark applications (spec-all) and their memory-bandwidth-demanding subset (spec-high specified in Table 5.4) are also presented. Six main memory system policies were used for the evaluation. Among the ones without row-buffer decoupling, open, close, and adapt employ the open-page, close-page, and adaptive row-buffer management policies, respectively. As for the system with decoupled row buffers, the delayed-restore (Dres) policy was evaluated as well

as the early-precharge (Epre) policy and its augmentation that utilizes ACT and WR without restore in the case of consecutive writes (a-Epre). For close, we make a row stay open until the tRAS timing constraint is met, as suggested by the minimalist-open policy [19], instead of closing it with an auto-precharge command. At the end of an epoch, adapt ping-pongs between the open- and close-page policies based on the row-buffer hit rate of the current epoch [16]. We set 100 DRAM accesses as the size of an epoch. MIPS2/W is inversely proportional to the energy-delay product. We populated one core and one memory controller for the SPEC CPU2006 single-threaded workloads to stress the main memory bandwidth. open is the baseline policy for each application. The number within the round brackets in the label per application is the absolute IPC on open.

We made the following observations from the experiments on the single-threaded workloads. **(1) The early-precharge policy (Epre) enabled by row-buffer decoupling substantially improves both the performance and energy efficiency of the simulated system compared to the policies without decoupled row buffers.** Among the main memory bandwidth demanding SPEC CPU2006 applications, 429.mcf, 459.GemsFDTD, and 471.omnetpp had a lower memory read latency and higher IPC for close rather than open because they have relatively higher row-buffer conflict rates on the simulated system. The other six applications grouped under spec-high favor open. adapt successfully traces the transition of the row-buffer conflict rates depending on applications and their phases so that it

gives a higher IPC and MIPS2/W than open and close even though the values are often similar to the better of the two. More importantly, Epre consistently outperforms adapt in the average read time, IPC, and MIPS2/W, which shows the effectiveness of overlapping the bitline precharge process and row-buffer retention through decoupling. On average, for main memory bandwidth demanding applications (spec-high), Epre gives 24%, 27%, and 14% higher IPC values than that of open, close, and adapt, respectively. Compared to adapt, Epre reduces the average main memory read latency by 20% and improves the MIPS2/W by 16%. **(2) The energy efficiency of the early-precharge policy can further be improved by allowing the memory controller to exploit the ACT and PRE commands without restore.** Even though Epre is consistently better than adapt in IPC and MIPS2/W, the gap in MIPS2/W between the two is small on 437.leslie3d and 470.lbm. This is because Epre consumes restore and precharge energy for every write. The augmented early-precharge policy, a-Epre, saves the restore and precharge energy by letting the memory controller detect the cases when the first access after row activation is write and there are multiple writes piled up in the request queue to an open row as described in Section 5.1.3. Thus, a-Epre performs equally to Epre but also improves MIPS2/W by as high as 23% on 470.lbm, and on average, 12% of the applications categorized as spec-high, respectively, compared to Epre. Note that the delayed-restore policy, Dres, is not as effective as Epre or adapt because it has higher latency penalties for row-buffer misses even though it

conserves DRAM dynamic energy. Because the additional static energy dissipation from the CPUs and the DRAM devices due to a longer execution time outweigh the energy savings from delaying restore operations, Dres does not match Epre in MIPS2/W either.

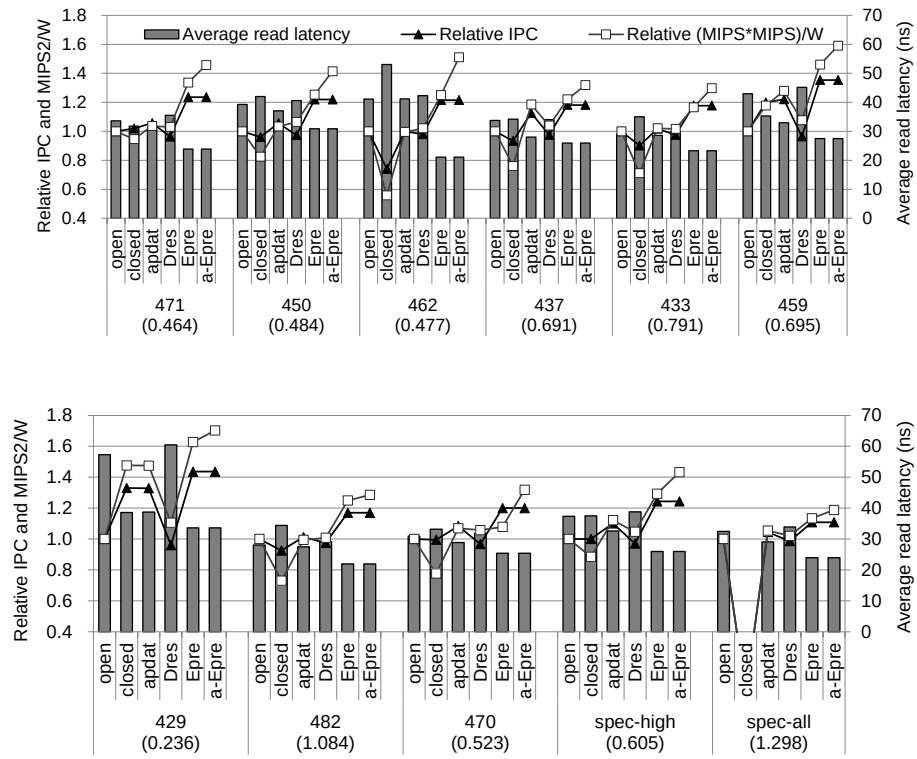


Figure 5.3: The average memory read latency, relative IPC, and MIPS2/W values of SPEC CPU2006 applications and their averages for the open-page, close-page, and adaptive management policies without decoupled row buffers and the delayed-restore (Dres), early-precharge (Epre), and augmented-early-precharge (a-Epre) policies with row-buffer decoupling. The number within the round brackets in the label per application is the absolute IPC on open.

(3) The impact of row-buffer decoupling on system-level performance and energy efficiency is diminished for less memory bandwidth demanding applications, but they also experience a noticeable reduction in the average read latency. Compared to adapt, Epre improves IPC by 14% on average for spec-high applications, but only 6% on average for all the SPEC CPU2006 applications (spec-all) because a reduction in memory accesses is less critical to the performance of applications with lower MAPKI values. However, the average main memory read latency of Epre is 20% and 19% lower than that of adapt for spec-high and spec-all, respectively, which is not very different. All these analyses show that neither the open- nor close-page policy clearly outperforms the other, and the adaptive policy is better than both, but it is only as good as the early-precharge policies in reducing the memory access time while its impact on system-level performance and energy efficiency depends on the main memory bandwidth demands of applications.

5.3.2 The Impact of Row-Buffer Decoupling on Multithreaded and Multiprogrammed Workloads

Row-buffer decoupling is also effective on multiprogrammed and multithreaded workloads. Figure 5.4 shows the average memory read latency, relative IPC, and relative MIPS2/W values of multiprogrammed and multithreaded workloads for the main memory system employing the open, close, and adapt policies

without row-buffer decoupling and for the system employing Dres, Epre, and a-Epre policies with decoupling. Compared to the system simulated for the single-threaded workloads, the system for the multi-programmed and multithreaded workloads has 16 times more cores and 4 times more memory controllers. Therefore, the number of banks per application decreases for the multiprogrammed workloads, which increases the row-buffer miss rates. Among the multiprogrammed workloads specified in Table 5.4, mix-high, mix-close, and mix-blend favor close while mix-open favors open. adapt performs better than the worse of the open and close, but it is less effective compared to the case of single-threaded workloads because main memory requests from different cores, which often have different access patterns, interfere with each other and make the adaptive decision made by the controllers less accurate. Because row-buffer decoupling does not rely on prediction, a-Epre is consistently superior to the policies for the system without the decoupled row buffers, which is better than adapt by 12% and 19% in IPC and MIPS2/W, respectively, for mix-blend.

Even though the sensitivity for memory access latency varies, the early-precharge policies improve the IPC and MIPS2/W of SPLASH-2 and PARSEC applications. For example, on LU, a SPLASH-2 application, a-Epre is 14% and 23% better than adapt in IPC and MIPS2/W, respectively. The STREAM benchmark [32] is an interesting application because its performance is much more sensitive to main memory bandwidth than the latency, and it has high spatial locality in memory accesses.

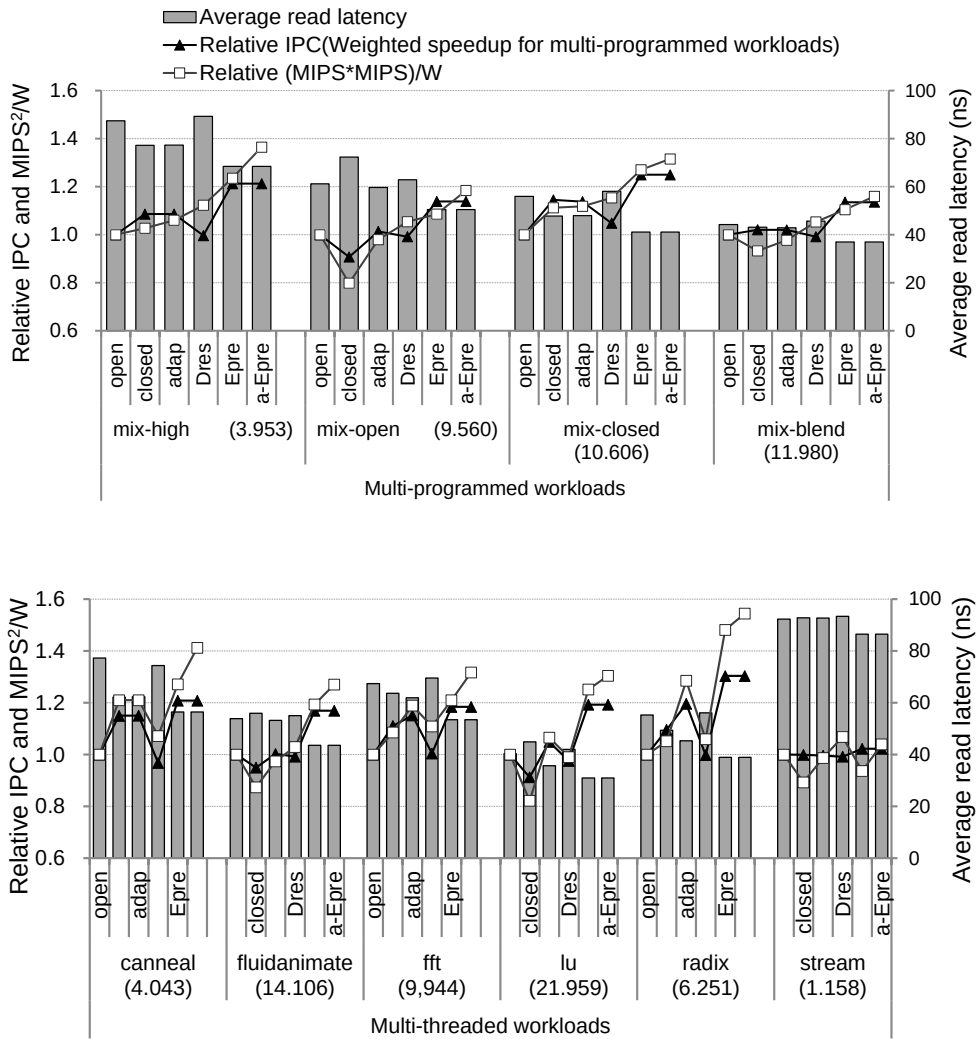


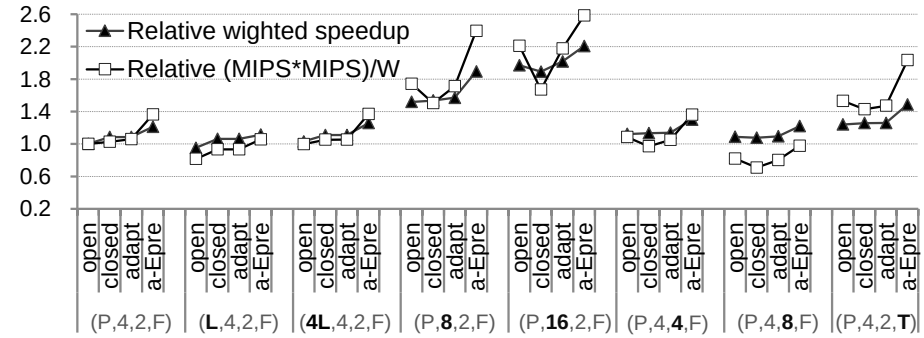
Figure 5.4: The average memory read latency, relative IPC, and MIPS2/W values of multiprogrammed (specified in Table 5.4) and multithreaded workloads for the open-page, close-page, and adaptive policies without row-buffer decoupling and the delayed-restore (Dres), early-precharge (Epre), and augmented-early-precharge (a-Epre) policies with decoupling.

Therefore, all the tested policies perform similarly and close, and Apre has the worse energy efficiency than that of the others

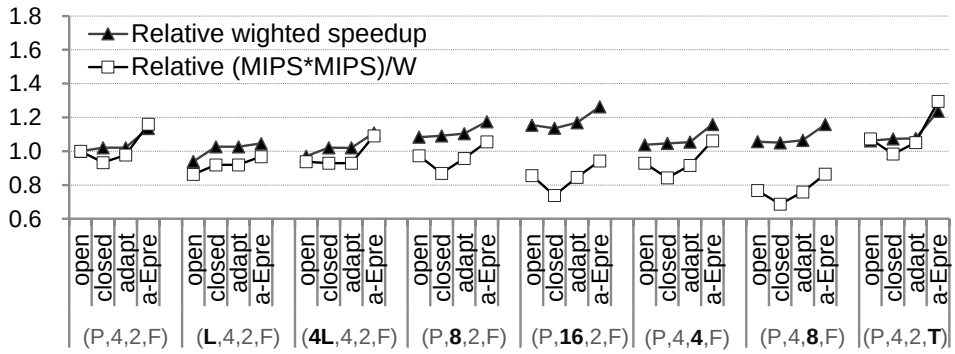
due to frequent restore and precharge μ-operations for WRs. Dres is more energy efficient than the others because it consumes less restore energy than the other policies as explained in Section 5.1.3. The energy efficiency of a-Epre is close to that of Dres and better than Epre because the memory controller of a-Epre successfully detects the stream of memory writes and minimizes the restore and precharge μ-operations, correspondingly.

5.3.3 Sensitivity Studies

The superiority of the DRAM devices with decoupled row buffers remains unchanged even if we apply data prefetching and vary the address interleaving scheme, number of memory channels, and number of ranks per channel. Figure 5.5 shows the relative IPC and MIPS2/W of the open, close, adapt, and a-Epre over various memory-system configurations for mix-high (Figure 5.5(a)) and mix-blend (Figure 5.5(b)). We use the notation (interleave, NMC, Nrank, T/F), for which interleave is either page interleaving (P), cache line interleaving (L), or interleaving at the granularity of 4 cache lines (4L); the number of memory channels is either 4, 8, or 16 (NMC = 4, 8, 16); the number of ranks per channel is either 2, 4, or 8 (Nrank = 2, 4, 8), and T/F means whether a hardware prefetcher [74] is used (T) or not (F). Bold fonts are used to highlight the parameters deviated from the default values. The open-page policy with (P, 4, 2, F) is the baseline per workload.



(a) mix−high



(b) mix−blend

Figure 5.5: The relative IPC and MIPS2/W of open, close, adapt, and a−Epre on mix−high (a) and mix−blend (b). We use the notation (interleave, NMC, Nrank, T/F), where interleave is page (P), line (L), and 4−line (4L) interleaving, 4, 8, and 16 memory channels (NMC), 2, 4, and 8 ranks per channel (Nrank), and T/F means if hardware prefetching is used (T) or not (F).

The cache−line−interleaving scheme relatively favors the close−page policy while the page−interleaving scheme favors the open−page policy in IPC and MIPS2/W. The 4−cache−lineinterleaving scheme, which is suggested by [19, 38], performs in−between. a−Epre outperforms the policies without decoupling

regardless of the interleaving schemes. With more memory controllers ((P, *, 2, F)), both the IPC and MIPS2/W values are improved for all the page-management policies, but the policy that provides the highest performance and energy efficiency is always a-Epre. When we increase the number of ranks per channel from 2 to 4 and 8, the IPC values first increase then decrease because more banks per channel lowers row-buffer conflicts, but the overhead of switching the ownership of memory channels (tRTRS [16]) negates the impact of reduced row-buffer conflicts when N_{rank} is large. Because populating more ranks implies that more DRAM devices dissipate static power and more I/O energy is needed per bit of data transfer due to inferior signal integrity, system power consumption increases resulting in a worse MIPS2/W. Prefetching increases both the IPC and DRAM power consumption because it decreases the average memory access time with decent prefetch hit rates, but at the same time wastes main memory bandwidth on prefetch misses. For the tested workloads, the advantages of prefetching outweigh the disadvantages and hence, both IPC and MIPS2/W are improved. More importantly, prefetching is beneficial to the page-management policies with row-buffer decoupling as well so that a-Epre is still the one with the highest performance and energy efficiency.

Chapter 6

Related Work

There is a large body of research that aims to improve the performance and energy efficiency of main-memory systems. Besides core-side multi-level caches and fast-page modes in DRAMs, there have been many proposals sharing this goal.

6.1 High-Performance DRAM Bank Structures

There have been several proposals to lower the average access time by reducing the number of cells attached to a bitline, including Reduced Latency DRAM (RLDRAM) [34], MoSys 1TSRAM [13], and Fast Cycle DRAM (FCRAM) [42]. This can be achieved by either fragmenting a cell array into smaller subarrays [34, 42] or by increasing the bank count [13]. However, these strategies mainly involve low-cost SRAM replacements with much higher area overhead than CHARM [16]. For example, an RLDRAM memory

This Section is based on [1, 2, 3]. - © [2013, 2014, 2016]ACM and IEEE Reprinted, with permissions, from ISCA' 13, ISCA' 14 and CAL' 16.

array and associated circuits are reported to be 40–80% larger than those of a comparable DDR2 device [20]. Single Subarray Access (SSA) [47] and Fine-Grained Activation [11] microarchitectures read or write data to an entirely activated row in a DRAM mat. This activates fewer mats per access and reduces energy consumption. However, it requires as many data lines as the row size of a mat, thus requiring much higher area and static power overhead than CHARM. Kim et al. [23] propose the subarray-level parallelism system, which overlaps the latencies of different requests that head to the same bank. Sudan et al. [45] propose micro-pages which allow chunks from different pages to be collocated in a row buffer to improve both the access latency and energy efficiency by better exploiting locality. CHARM is complementary to both proposals, as we focus on modifying the microarchitecture of DRAM banks to lower the access and cycle times.

6.2 DRAM-Side Caching

Both Enhanced DRAM [56] and Virtual Channel DRAM [60] add an SRAM buffer to the DRAM device to cache frequently accessed DRAM data. Any hit in the cache obviates the need for a row access and hence improves the access time. Because an SRAM cell is much larger than a DRAM cell, the cache incurs significant area overhead. In contrast, CHARM adds only minimal area overhead (a 3% increase) to conventional DRAM devices and is suitable for multi-

gigabyte main memory. Tiered-latency DRAM [26] also advocates an idea to make a portion of a mat to have lower access time, but it is targeted to be used as a cache. ChargeCache [81] is based on the key observation that DRAM latency of following access to the same row can be shorter, but only reducing the sensing delay is not enough to improve system performance.

6.3 3D Die Stacking

Madan et al. [31] and Loh and Hill [30] propose the idea of stacking a DRAM die on top of a processor die, connecting them using through-silicon vias (TSVs), and using the DRAM die as last-level caches, while wide I/O DRAMs [22] are used for main memory. Hybrid Memory Cube [38] (HMC) packages a logic die below multiple DRAM dies to improve the capacity per package and bandwidth for a processor package. Udipi et al. [46] also propose to offload part of the memory controller's function to the logic die of an HMC-style structure, and Loh [29] puts row-buffer caches onto the logic die. It is feasible to stack multiple CHARM DRAM dies to further reduce the access latency while increasing access bandwidth.

6.4 DRAM Module-Level Solutions

To enhance the energy efficiency of accessing main memory, multiple groups [4, 58] have proposed rank subsetting, the idea of

utilizing not all but a subset of DRAM chips in a DRAM rank to activate fewer DRAM cells per access. These methods save energy at the cost of additional latency, while CHARM lowers both access latency and energy consumption. Rank subsetting has been applied to improve the reliability and energy efficiency of main-memory systems as well [47, 53, 54]. The Fully-Buffered DIMM (FBDIMM) architecture [12] is characterized by non-uniform access time to DRAM banks, which is similar to CHARM from the viewpoint of a memory controller. However, an increase in latency and power consumption incurred by Advanced Memory Buffers in FBDIMM limits its applicability compared to CHARM.

6.5 Memory access schedulers

Many studies have focused on memory scheduling to improve either bandwidth utilization (i.e., higher DRAM throughput), fairness, or both. To achieve higher bandwidth utilization, the FR-FCFS scheduling policy prioritizes row-buffer hits [40]. FR-FCFS has been readily accepted for single-core processors, but it does not guarantee any fairness when used in a chip-multiprocessor (CMP) environment because it tends to favor applications with high row locality. Recently proposed policies attempt to make a memory controller aware of the characteristics of diverse applications being executed, and provide fairness to all applications. The parallelism-aware batch scheduling (PAR-BS) scheme issues requests in batches to provide fairness and tries to maintain the original bank-

level parallelism of an application [35]. The adaptive per-thread least-attained-service scheduler (ATLAS) prioritizes requests from applications that have received the least service from memory controllers [75]. Thread cluster memory scheduling (TCM) improves both bandwidth utilization and fairness by dividing applications into two groups, i.e. latency-sensitive and bandwidth-sensitive cluster, based on bank-level parallelism, row locality, and the memory intensity of the applications being executed [76]. As an adaptive memory scheduling approach, multi-objective reconfigurable self-optimizing memory scheduler (MORSE) adjusts the parameters of a self-optimizing memory scheduler [71] for various objectives such as power or energy instead of just bandwidth utilization through various techniques [78]. In this dissertation, we leveraged the PAR-BS scheme, but all these scheduling policies are orthogonal to row-buffer decoupling because the schedulers make decisions based on the pending requests in the queue while our row-buffer decoupling targets the case in which the information provided by the queue is insufficient.

6.6 DRAM page-management policies

Minimalist open-page policy [19] interleaves memory accesses at a sub-row granularity across channels and banks to minimize memory access interference between applications contending for access to the main memory system with an open-page policy, increasing bandwidth utilization and improving the fairness for

individual applications in a CMP environment. An adaptive open-and close-page policy, which was implemented in commercial Intel processors, attempts to improve both bandwidth utilization and access latency by adaptively choosing a page policy depending on the memory access characteristics of applications being concurrently executed [15]. We have already compared the adaptive open- and close-page policy with row-buffer decoupling and studied the impact of the address interleaving schemes.

Chapter 7

Conclusion

Recent research and development of DRAM microarchitectures for main-memory systems have mostly focused on increasing the capacity, bandwidth, and energy efficiency of the DRAM devices while retaining or even sacrificing the access latency. However, application performance is often sensitive not only to bandwidth but also to latency and applications access small portions of the memory footprints more frequently than the remainder. This observation has motivated us to propose asymmetric DRAM bank organizations that reduce the average access and cycle times and analyze their system-level impacts on performance and energy efficiency.

The cycle time analysis of contemporary DRAM devices shows that the sensing, restore, and precharge processes are slow due to high bitline capacitance within DRAM mats. Increasing the aspect

---

This Section is based on [1, 2, 3]. - © [2013, 2014, 2016] ACM and IEEE Reprinted, with permissions, from ISCA' 13, ISCA' 14 and CAL' 16.

ratio of the mats cuts the cycle time more than half and also reduces the access time and activate energy at the cost of increased area. To minimize the area overhead, we synergistically combine the high-aspect-ratio mats with support for non-uniform bank accesses to devise a novel DRAM organization called CHARM. A CHARM DRAM device places the high-aspect-ratio mats only at the center blocks, which are physically closer to I/Os, and reorganizes the column decoders and inter-bank datalines so that the access time to the center high-aspect-ratio mats is about half the access time to the other normal mats. Simulation results on a chip-multiprocessor system demonstrate that applications with higher bandwidth demands on main memory typically benefit more from CHARM. For example, CHARM DRAM devices with 2× higher aspect-ratio mats placed on a quarter of the DRAM banks increase the area only by 3%, but improve the IPC and the EDP of the system up to 21% and 32%, respectively, on single-threaded applications compared to the reference uniform organization. Similar degrees of performance and energy efficiency gains are also realized on the multithreaded and multiprogrammed workloads that frequently access main memory.

We have also proposed a low-latency main memory DRAM architecture called SALAD. SALAD reduces the access latency of all the banks in a DRAM device by 1) allowing each bank to have different tCL based on its distance to the I/O pads, and 2) reducing the activate and precharge time of the remote banks by decreasing the bitline capacitance of the mats for those banks. This is in

contrast to CHARM, which applies the aforementioned techniques only to the center banks, thus yielding non-uniform memory access latency (tAC) among banks. SALAD achieves lower access latency than a DRAM architecture that apply HAR mats to the entire DRAM device (All-HAR). Our evaluation demonstrates that SALAD outperforms the baseline DDR4 and CHARM by 10% and 6%, respectively, for memory-intensive SPEC applications at a 3% area overhead without requiring sophisticated memory allocation.

Finally, We have proposed a new DRAM microarchitecture for the main memory that decouples a bitline sense amplifier (BLSA), which works as a row buffer, from the corresponding bitline pair using isolation transistors. The proposal is based on the key observations that 1) the number of banks per memory channel keeps increasing so that the DRAM cycle time (minimal interval between row activates at a bank) has little influence on access latency; 2) the capacitance of a bitline is significantly larger than that of a BLSA so that bitline precharging takes most of the precharge time on modern DRAM devices, and 3) the precharge process is on the critical path in the case of a row-buffer miss.

The decoupling allows BLSAs to retain an open row and the bitlines to be precharged right after the row becomes activated so that the bitline precharging is off the critical path of the following row-buffer miss in most cases. On a row-buffer conflict, only the BLSAs need to be precharged, taking significantly less time than that of the conventional precharge process. Therefore, the early-precharge policy with decoupled row buffers perform as good as the

open-row policy on a row-buffer hit, almost the same as the close-page policy on a miss, and better than both policies when row-buffer hits and misses are mixed. Row-buffer decoupling incurs less than 1% area overhead, but the early-precharge policy causes excessive bitline precharge operations for writes. We mitigate this energy overhead by making the bitline precharge optional for the activate and write commands, enabled by row-buffer decoupling that separate and recombine internal DRAM μ-operations, and letting the memory controller choose either type based on the requests stacked in its queue. The simulation results show that row-buffer decoupling improves the instruction per cycle and MIPS2/W by 24% and 43%, respectively, compared to the open-page policy on average for nine main memory bandwidth demanding SPEC CPU2006 applications.

Bibliography

[1] Y. H. Son et al., “Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations,” in 40th ACM International Symposium on Computer Architecture (ISCA-40), Tel-Aviv, Israel, June 2013.

[2] Y. H. Son et al., " SALAD: Achieving Symmetric Access Latency with Asymmetric DRAM Architecture," IEEE Computer Architecture Letters, 2016.

[3] S. O et al., “Row-Buffer Decoupling: A Case for Low-Latency DRAM Microarchitecture,” in 41st IEEE International Symposium on Computer Architecture (ISCA-41), Minneapolis, Minnesota, USA, June 2014.

[4] J. Ahn et al., “Improving System Energy Efficiency with Memory Rank Subsetting,” ACM Transactions on Architecture and Code Optimization, vol. 9, no. 1, 2012.

[5] J. Ahn et al., “McSimA+: A Manycore Simulator with Application-level+ Simulation and Detailed Microarchitecture Modeling,” in IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, USA,

April 2013.

[6] R. Alverson et al., “The Tera Computer System,” in 4th ACM International Conference on Supercomputing (ICS-4), Amsterdam, Netherlands, June 1990.

[7] D. L. Anand et al., “Embedded DRAM in 45-nm Technology and Beyond,” IEEE Design Test of Computers, vol. 28, no. 1, 2011.

[8] S. J. Bae et al., “A 40nm 2Gb 7Gb/s/pin GDDR5 SDRAM with a Programmable DQ Ordering Crosstalk Equalizer and Adjustable Clock-tracking BW,” in IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, February 2011.

[9] A. Bhattacharjee et al., “Thread Criticality Predictors for Dynamic Performance, Power, and Resource Management in Chip Multiprocessors,” in 36th ACM International Symposium on Computer Architecture (ISCA-36), New York, NY, USA, June 2009.

[10] C. Bienia et al., “The PARSEC Benchmark Suite: Characterization and Architectural Implications,” in 17th ACM International Conference on Parallel Architectures and Compilation Techniques (PACT-17), New York, NY, USA, October 2008.

[11] E. Cooper-Balis et al., “Fine-Grained Activation for Power Reduction in DRAM,” IEEE Micro, vol. 30, no. 3, 2010.

[12] B. Ganesh et al., “Fully-Buffered DIMM Memory Architectures: Understanding Mechanisms, Overheads and Scaling,” in 13th IEEE International Symposium on High Performance Computer Architecture (HPCA-13), Scottsdale, AZ, USA, February 2007.

[13] P. N. Glaskowsky, “MoSys Explains 1T-SRAM Technology,”

Microprocessor Report, September. 1999.

[14] M. Hashimoto et al., “An Embedded DRAM Module using a Dual Sense Amplifier Architecture in a Logic Process,” in 43rd IEEE International Solid-State Circuits Conference (ISSCC-43), San Francisco, CA, USA, February 1997.

[15] J. L. Henning, “SPEC CPU2006 Memory Footprint,” Computer Architecture News, vol. 35, no. 1, 2007.

[16] B. Jacob et al., "Memory Systems: Cache, DRAM, Disk," Morgan Kaufmann Publishers Inc., 2007.

[17] D. James, “Recent Innovations in DRAM Manufacturing,” in IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC), San Francisco, CA, USA, July 2010.

[18] U. J. Kapasi et al., “Programmable Stream Processors,” IEEE Computer, vol. 36, no. 8, 2003.

[19] D. Kaseridis et al., “Minimalist Open-page: a DRAM Page-mode Scheduling Policy for the Many-core Era,” in 44th IEEE/ACM International Symposium on Microarchitecture (MICRO-44), Porto Alegre, Brazil, December 2011.

[20] B. Keeth et al., "DRAM Circuit Design, 2nd ed.," IEEE, 2008.

[21] C. Kim et al., “An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches,” in 10th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-10), San Jose, CA, USA, October 2002.

[22] J. S. Kim et al., “A 1.2V 12.8GB/s 2Gb mobile Wide-I/O DRAM with 4x 128 I/Os using TSV-based stacking,” in IEEE International Solid-State Circuits Conference (ISSCC), San

Francisco, CA, USA, February 2011.

[23] Y. Kim et al., “A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM,” in 39th IEEE International Symposium on Computer Architecture (ISCA-39), Portland, Oregon, USA, June 2012.

[24] C. Kozyrakis, “Scalable Vector Media-processors for Embedded Systems,” Ph.D. dissertation, University of California at Berkeley, 2002.

[25] D. Lee et al., “LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies,” IEEE Transactions on Computers, vol. 50, no. 12, 2001.

[26] D. Lee et al., “Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture,” in 19th IEEE International Symposium on High Performance Computer Architecture (HPCA-19), Shenzhen, China, February 2013.

[27] S. Li et al., “The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing,” ACM Transactions on Architecture and Code Optimization, vol. 10, no. 1, 2013.

[28] E. Lindholm et al., “NVIDIA Tesla: A Unified Graphics and Computing Architecture,” IEEE Micro, vol. 28, no. 2, 2008.

[29] G. H. Loh, “A Register-file Approach for Row Buffer Caches in Die-stacked DRAMs,” in 44th IEEE/ACM International Symposium on Microarchitecture (MICRO-44), Porto Alegre, Brazil, December 2011.

[30] G. H. Loh et al., “Efficiently Enabling Conventional Block Sizes for Very Large Die-stacked DRAM Caches,” in 44th IEEE/ACM

International Symposium on Microarchitecture (MICRO-44), Porto Alegre, Brazil, December 2011.

[31] N. Madan et al., "Optimizing Communication and Capacity in a 3D Stacked Reconfigurable Cache Hierarchy," in 15th IEEE International Symposium on High Performance Computer Architecture (HPCA-15), Raleigh, North Carolina, USA, February 2009.

[32] J. D. McCalpin, "STREAM: Sustainable Memory Bandwidth in High Performance Computers," University of Virginia, Tech. Rep., 1991.

[33] Micron Technology Inc., LPDDR2 SDRAM Datasheet, 2010.

[34] Micron Technology Inc., RLDRAM3 Datasheet, 2011.

[35] O. Mutlu et al., "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems," in 35th IEEE International Symposium on Computer Architecture (ISCA-35), Beijing, China, June 2008.

[36] D. Patterson et al., "A Case for Intelligent RAM," IEEE Micro, vol. 17, no. 2, 1997.

[37] D. A. Patterson et al., "Computer Architecture: A Quantitative Approach, 5th ed.," Morgan Kaufmann Publishers Inc., 2012.

[38] J. T. Pawlowski, "Hybrid Memory Cube," in Hot Chips, August 2011.

[39] L. E. Ramos et al., "Page Placement in Hybrid Memory Systems," in 25th ACM International Conference on Supercomputing (ICS-25), Tucson, Arizona, USA, June 2011.

[40] S. Rixner et al., "Memory Access Scheduling," in 27th ACM International Symposium on Computer Architecture (ISCA-27),

Vancouver, British Columbia, Canada, June 2000.

[41] Samsung Electronics, DDR3 SDRAM Datasheet, 2012

[42] Y. Sato et al., "Fast Cycle RAM (FCRAM); a 20-ns Random Row Access, Pipelined Operating DRAM," in IEEE symposium on VLSI Circuits (VLSI), Honolulu, Hawaii, USA, June 1998.

[43] T. Sherwood et al., "Automatically Characterizing Large Scale Program Behavior," in 10th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-10), San Jose, CA, USA, October 2002.

[44] A. Snavely et al., "Symbiotic Job Scheduling for a Simultaneous Mutlithreading Processor," in 9th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-9), Cambridge, MA, USA, November 2000.

[45] K. Sudan et al., "Micro-pages: Increasing DRAM Efficiency with Locality-aware Data Placement," in 15th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-15), Pittsburgh, Pennsylvania, USA, October 2010.

[46] A. N. Udipi et al., "Combining Memory and a Controller with Photonics through 3D-stacking to Enable Scalable and Energy-efficient Systems," in 38th ACM International Symposium on Computer Architecture (ISCA-38), San Jose, California, USA, June 2011.

[47] A. N. Udipi et al., "Rethinking DRAM Design and Organization for Energy-constrained Multi-cores," in 37th ACM International

Symposium on Computer Architecture (ISCA-37), Saint-Malo, France, June 2010.

[48] B. Verghese et al., "Operating System Support for Improving Data Locality on cc-NUMA Compute Servers," in 7th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-7), Cambridge, MA, USA, October 1996.

[49] T. Vogelsang, "Understanding the Energy Consumption of Dynamic Random Access Memories," in 43rd IEEE/ACM International Symposium on Microarchitecture (MICRO-43), Atlanta, Georgia, USA, December 2010.

[50] S. C. Woo et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations," in 22nd ACM International Symposium on Computer Architecture (ISCA-22), S. Margherita Ligure, Italy, June 1995.

[51] W. A. Wulf et al., "Hitting the Memory Wall: Implications of the Obvious," Computer Architecture News, vol. 23, no. 1, 1995.

[52] Y. Yanagawa et al., "In-substrate-bitline Sense Amplifier with Array-noise-gating Scheme for Low-noise 4F2 DRAM Array Operable at 10-fF Cell Capacitance," in IEEE symposium on VLSI Circuits (VLSI), Kyoto, Japan, June 2011.

[53] D. H. Yoon et al., "BOOM: Enabling Mobile Memory Based Low-Power Server DIMMs," in 39th IEEE International Symposium on Computer Architecture (ISCA-39), Portland, Oregon, USA, June 2012.

[54] D. H. Yoon et al., "Virtualized ECC: Flexible Reliability in Main Memory," IEEE Micro, vol. 31, no. 1, 2011.

[55] D. H. Yoon et al., “Adaptive Granularity Memory Systems: a Tradeoff Between Storage Efficiency and Throughput,” in 38th ACM International Symposium on Computer Architecture (ISCA-38), San Jose, California, USA, June 2011.

[56] Z. Zhang et al., “Cached DRAM for ILP Processor Memory Access Latency Reduction,” IEEE Micro, vol. 21, no. 4, 2001.

[57] W. Zhao et al., “New Generation of Predictive Technology Model for Sub-45nm Design Exploration,” in 7th IEEE International Symposium on Quality Electronic Design (ISQED-7), San Jose, CA, USA, March 2006.

[58] H. Zheng et al., “Mini-Rank: Adaptive DRAM Architecture for Improving Memory Power Efficiency,” in 41st IEEE/ACM International Symposium on Microarchitecture (MICRO-41), Lake Como, Italy, November 2008.

[59] “The SAP HANA Database,” http://www.sap.com.

[60] “Virtual Channel DRAM. Elpida Memory, Inc.” http://www.elpida.com/en/products/eol/vcdram.html.

[61] J. Ahn, “ccTSA: A Coverage-Centric Threaded Sequence Assembler,” PLoS ONE, vol. 7, no. 6, 2012.

[62] JEDEC, DDR4 SDRAM Specification, 2012.

[63] N. P. Jouppi, “Improving Direct-mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers,” in 17th ACM International Symposium on Computer Architecture (ISCA-17), Seattle, WA, USA, June 1990.

[64] S. Li et al., “Architecting to Achieve a Billion Requests Per Second Throughput on a Single Key-Value Store Server Platform,” in 42nd ACM International Symposium on Computer

Architecture (ISCA-42), Portland, Oregon, USA, June 2015.

[65] W. Shin et al., "NUAT: A Non-uniform Access Time Memory Controller," in 20th IEEE International Symposium on High Performance Computer Architecture (HPCA-20), Orlando, Florida, USA, February 2014.

[66] R. Ausavarungnirun et al., "Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems," in 39th IEEE International Symposium on Computer Architecture (ISCA-39), Portland, Oregon, USA, June 2012.

[67] L. A. Barroso et al., "Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing," in 27th ACM International Symposium on Computer Architecture (ISCA-27), Vancouver, British Columbia, Canada, June 2000.

[68] H. Choi et al., "Memory Access Pattern-Aware DRAM Performance Model for Multi-Core Systems," in IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, USA, April 2011.

[69] J. C. Gealow, "Impact of Processing Technology on DRAM Sense Amplifier Design," Ph.D. dissertation, Massachusetts Institute of Technology, 1990.

[70] Intel, Intel Xeon Processor 7500 Series Datasheet, March 2010.

[71] E. Ipek et al., "Self-Optimizing Memory Controllers: A Reinforcement Learning Approach," in 35th IEEE International Symposium on Computer Architecture (ISCA-35), Beijing, China, June 2008.

[72] JEDEC, DDR4 SDRAM Specification, September 2012.

[73] JEDEC, Graphics Double Data Rate (GDDR5) SGRAM Standard,

December 2013.

[74] N. P. Jouppi, “Improving Direct-mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers,” in 17th ACM International Symposium on Computer Architecture (ISCA-17), Seattle, WA, USA, June 1990.

[75] Y. Kim et al., “ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers.” in 16th IEEE International Symposium on High Performance Computer Architecture (HPCA-16), Bangalore, India, January 2010.

[76] Y. Kim et al., “Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior,” in 43rd IEEE/ACM International Symposium on Microarchitecture (MICRO-43), Atlanta, Georgia, USA, December 2010.

[77] Y. Kim et al., “A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM,” in 39th IEEE International Symposium on Computer Architecture (ISCA-39), Portland, Oregon, USA, June 2012.

[78] J. Mukundan et al., “MORSE: Multi-Objective Reconfigurable Self-Optimizing Memory Scheduler,” in 18th IEEE International Symposium on High Performance Computer Architecture (HPCA-18), New Orleans, Louisiana, USA, February 2012.

[79] T. Takahashi et al., “A Multi-gigabit DRAM Technology with 6F2 Open-Bitline Cell, Distributed Overdriven Sensing, and Stacked-Flash Fuse,” IEEE Journal of Solid-State Circuits, vol. 36, no. 11, 2001.

[80] B. Wicht et al., “Analysis and Compensation of the Bitline Multiplexer in SRAM Current Sense Amplifiers,” IEEE Journal of

Solid-State Circuits, vol. 36, no. 11, 2001.

[81] H. Hassan et al. "ChargeCache: Reducing DRAM latency by exploiting row access locality." in 22nd IEEE International Symposium on High Performance Computer Architecture (HPCA-22), Barcelona, Spain, March 2016.

국문초록

DRAM 은 주 기억장치의 사실상의 표준이 되어 왔고, 공정 기술의 진보는 DRAM 용량과 대역폭을 빠르게 증가시켰다. 이에 반해 랜덤 접근 지연시간은 약 100 CPU 클럭 사이클 수준으로 상대적으로 정체되어 있다. 현대 컴퓨터 시스템은 이러한 긴 평균 접근 지연시간을 줄이기 위해 캐시를 이용하거나 이를 극복할 수 있는 다른 기술들을 사용한다. 그러나 모든 어플리케이션들이 접근 지연시간을 감출 수 있을 정도로 충분한 병렬성을 가지고 있지 못하며 캐시 이용에 적합한 지역성 또한 충분하지 못하다. 게다가 많은 어플리케이션들이 큰 메모리 용량을 필요로 하며 최종 캐시와 메모리 사이의 용량 차이는 쉽게 줄어들지 않을 것이다. 결과적으로 메모리 접근 지연시간을 줄이는 것은 어플리케이션 성능에 있어서 매우 중요하다. 하지만 불행하게도 이전 연구들은 이 문제를 적절히 해결하지 않았는데 이는 대역폭과 용량 증가에 집중하거나 상당한 면적 오버헤드를 가지고 지연시간을 줄여 왔기 때문이다.

현대 DRAM 은 처리량과 에너지 효율 그리고 용량 수요를 증가시키기 위해 다중 뱅크를 채용하고 있다. 또한 비용적인 제약으로 인해 뱅크당 오직 한 개의 행만 저장될 수 (열릴 수) 있고 새로운 행을 행 버퍼에 저장하기 전에는 반드시 기존에 열려있는 행을 비활성화 시켜야 한다. 이 때 미리 비활성화를 시켰는데 행 버퍼 히트가 된 경우 이미 비활성화된 행을 불필요하게 다시 열어야 하는 문제가 발생하고 행 버퍼 미스인데 비활성화를 미리 시키지 않았으면 비활성화 동작을 위한 시간이 데이터 접근을 위한 시간에 추가되어야 한다. (비)활성화를 위한 시간은 열린 행에서 데이터를 읽는 시간만큼 상당하기 때문에 행을

비활성화 시키는 시기를 결정하는 것은 매우 중요하다. 하지만 DRAM 당 뱅크 개수가 증가하면 뱅크당 리퀘스트 수는 감소하게 되고 미래의 리퀘스트 정보가 부족하게 되어 결국 메모리 컨트롤러는 행을 비활성화 시키는 시기에 대한 결정을 예측에 의존할 수 밖에 없게 된다. 하지만 메모리 접근 형태가 매우 동적이기 때문에 예측에 대한 정확도 문제가 발생한다.

이 논문에서는 DRAM 의 평균 접근 지연시간을 줄일 수 있는 획기적인 세가지 DRAM 뱅크 구조를 제안한다. 첫번째 제안하는 구조는 CHARM 이라 불리는 종횡비가 큰 (지연시간이 작은) 매트를 DRAM 중앙에 위치시키는 것이다. 실험 결과 CHARM (Center High-Aspect Ratio Mats)을 사용하게 되면 IPC 는 21%, 에너지 효율은 32%까지 증가하는 것을 확인 하였다. 두번째 구조는 CHARM 과 달리 비대칭 DRAM 뱅크 구조에 대칭 접근 지연시간을 갖는 SALAD (Symmetric Access Latency with Asymmetric DRAM bank organizations) 이다. SALAD 는 종횡비가 큰 매트를 I/O 패드와 멀리 떨어진 뱅크에 적용하여 가까운 뱅크와 멀리 떨어진 뱅크 간의 데이터 전송 시간에 대한 차이를 줄여 전체 뱅크간 고르게 접근 지연 시간을 줄인다. 실험 결과 SALAD 구조를 이용하면 3% 면적 오버헤드로 IPC 가 10% 개선되는 것을 확인하였다. 마지막으로 행을 닫는 시기에 대한 어떠한 예측도 필요 없는 획기적인 DRAM 구조를 제안한다. Bitline 과 행 버퍼를 트랜지스터를 이용하여 분리함으로써 행의 활성화 직후 해당 행을 바로 비활성화 하는 것이 가능하게 된다. 만약 행 버퍼 미스일 경우 sense amplifier 만 비활성화 하면 된다. 또한 이렇게 분리된 구조는 기존 DRAM 동작을 내부의 마이크로 동작으로 분리, 재결합을 가능하게 하고 메모리 컨트롤러는 메모리 시스템을 더욱 에너지 효율적으로 관리할 수 있게 한다. 실험 결과 메모리 접근빈도가 높은

SPEC CPU2006 어플리케이션들에 대해 IPC 는 14%, 에너지 효율은 29%까지 증가하는 것을 확인하였다.

주요어 : 메모리 시스템, DRAM, CHARM, SALAD, row-buffer decoupling, 접근 지연시간, 면적 오버헤드
학 번: 2012-30700