A Performance Comparison of Contemporary DRAM Architectures

Vinodh Cuppu, Bruce Jacob
Dept. of Electrical & Computer Engineering
University of Maryland, College Park
{ramvinod,blj}@eng.umd.edu

Brian Davis, Trevor Mudge
Dept. of Electrical Engineering & Computer Science
University of Michigan, Ann Arbor
{btdavis,tnm}@eecs.umich.edu

ABSTRACT

In response to the growing gap between memory access time and processor speed, DRAM manufacturers have created several new DRAM architectures. This paper presents a simulation-based performance study of a representative group, each evaluated in a small system organization. These small-system organizations correspond to workstation-class computers and use on the order of 10 DRAM chips. The study covers Fast Page Mode, Extended Data Out, Synchronous, Enhanced Synchronous, Synchronous Link, Rambus, and Direct Rambus designs. Our simulations reveal several things: (a) current advanced DRAM technologies are attacking the memory bandwidth problem but not the latency problem; (b) bus transmission speed will soon become a primary factor limiting memory-system performance; (c) the post-L2 address stream still contains significant locality, though it varies from application to application; and (d) as we move to wider buses, row access time becomes more prominent, making it important to investigate techniques to exploit the available locality to decrease access time.

1 INTRODUCTION

In response to the growing gap between memory access time and processor speed, DRAM manufacturers have created several new DRAM architectures. This paper presents a simulation-based performance study of a representative group, evaluating each in terms of its effect on total execution time. We simulate the performance of seven DRAM architectures: Fast Page Mode [35], Extended Data Out [16], Synchronous [17], Enhanced Synchronous [10], Synchronous Link [38], Rambus [31], and Direct Rambus [32]. While there are a number of academic proposals for new DRAM designs, space limits us to covering only existent commercial parts. To obtain accurate memory-request timing for an aggressive out-of-order processor, we integrate our code into the SimpleScalar tool set [4].

This paper presents a baseline study of a *small-system DRAM organization*: these are systems with only a handful of DRAM chips (0.1–1GB). We do not consider large-system DRAM organizations with many gigabytes of storage that are highly interleaved. The study asks and answers the following questions:

- What is the effect of improvements in DRAM technology on the memory latency and bandwidth problems?

  Contemporary techniques for improving processor performance and tolerating memory latency are exacerbating the memory bandwidth problem [5]. Our results show that current DRAM architectures are attacking exactly this problem: the most recent technologies (SDRAM, ESDRAM, and Rambus) have reduced the stall time due to limited bandwidth by a factor of three compared to earlier DRAM architectures. However, the memory-latency component of overhead has not improved.

- Where is time spent in the primary memory system (the memory system beyond the cache hierarchy, but not including secondary [disk] or tertiary [backup] storage)? What is the performance benefit of exploiting the page mode of contemporary DRAMs?

  For the newer DRAM designs, the time to extract the required data from the sense amps/row caches for transmission on the memory bus is the largest component in the average access time, though page mode allows this to be overlapped with column access and the time to transmit the data over the memory bus.

- How much locality is there in the address stream that reaches the primary memory system?

  The stream of addresses that miss the L2 cache contains a significant amount of locality, as measured by the hit-rates in the DRAM row buffers. The hit rates for the applications studied range 8–95%, with a mean hit rate of 40% for a 1MB L2 cache. (This does not include hits to the row buffers when making multiple DRAM requests to read one cache-line.)

We also make several observations. First, there is a one-time trade-off between cost, bandwidth, and latency: to a point, latency can be decreased by ganging together multiple DRAMs into a wide structure. This trades dollars for bandwidth that reduces latency because a request size is typically much larger than the DRAM transfer width. Page mode and interleaving are similar optimizations that work because a request size is typically larger than the bus width. However, the latency benefits are limited by bus and DRAM speeds: to get further improvements, one must run the DRAM core and bus at faster speeds. Current memory busses are adequate for small systems but are likely inadequate for large ones. Embedded DRAM [5, 19, 37] is not a near-term solution, as its performance is poor on high-end workloads [3]. Faster buses are more likely solutions—witness the elimination of the slow intermediate memory bus in future systems [12]. Another solution is to internally bank the memory array into many small arrays so that each can be accessed very quickly, as in the MoSys Multibank DRAM architecture [39].

Second, widening buses will present new optimization opportunities. Each application exhibits a different degree of locality and therefore benefits from page mode to a different degree. As buses widen, this effect becomes more pronounced, to the extent that different applications can have average access times that differ by 50%. This is a minor issue considering current bus technology. However, future bus technologies will expose the row access as the primary performance bottleneck, justifying the exploration of mechanisms to exploit locality to guarantee hits in the DRAM row buffers: e.g. row-buffer victim caches, prediction mechanisms, etc.

Third, while buses as wide as the L2 cache yield the best memory latency, they cannot halve the latency of a bus half as wide. Page mode overlaps the components of DRAM access when making multiple requests to the same row. If the bus is as wide as a request, one

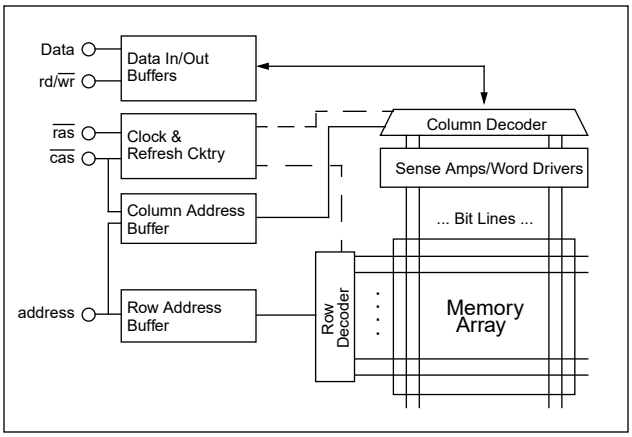

Figure 1: Conventional DRAM block diagram. The conventional DRAM uses a split addressing mechanism still found in most DRAMs today.

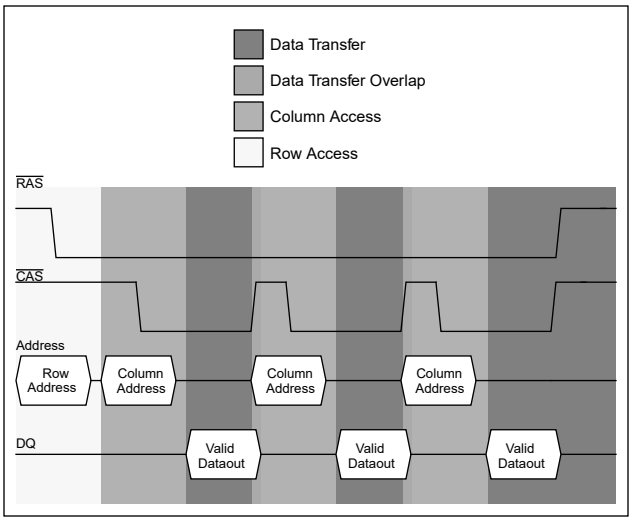

Figure 2: FPM Read Timing. Fast page mode allows the DRAM controller to hold a row constant and receive multiple columns in rapid succession.

cannot exploit this overlap. For cost considerations, having at most an N/2-bit bus, N being the L2 cache width, might be a good choice.

Fourth, critical-word-first does not mix well with burst mode. Critical-word-first is a strategy that requests a block of data potentially out of address-order; burst mode delivers data in a fixed but redefinable order. A burst-mode DRAM can thus can have longer latencies in real systems, even if its end-to-end latency is low.

Finally, the choice of refresh mechanism can significantly alter the average memory access time. For some benchmarks and some refresh organizations, the amount of time spent waiting for a DRAM in refresh mode accounted for 50% of the total latency.

As one might expect, our results and conclusions are dependent on our system specifications, which we chose to be representative of mid- to high-end workstations: a 100MHz 128-bit memory bus, an eight-way superscalar out-of-order CPU, lockup-free caches, and a small-system DRAM organization with ~10 DRAM chips.

2 RELATED WORK

Burger, Goodman, and Kagi quantified the effect on memory behavior of high-performance latency-reducing or latency-tolerating techniques such as lockup-free caches, out-of-order execution, prefetching, speculative loads, etc. [5]. They concluded that to hide memory latency, these techniques often increase demands on memory bandwidth. They classify memory stall cycles into two types: those due to lack of available memory bandwidth, and those due purely to latency. This is a useful classification, and we use it in our study. This study differs from theirs in that we focus on the access time of only the primary memory system, while their study combines all memory access time, including the L1 and L2 caches. Their study focuses on the behavior of latency-hiding techniques, while this study focuses on the behavior of different DRAM architectures.

Several marketing studies compare the memory latency and bandwidth available from different DRAM architectures [7, 29, 30]. This paper builds on these studies by looking at a larger assortment of DRAM architectures, measuring DRAM impact on total application performance, decomposing the memory access time into different components, and measuring the hit rates in the row buffers.

Finally, there are many studies that measure system-wide performance, including that of the primary memory system [1, 2, 9, 18, 23, 24, 33, 34]. Our results resemble theirs, in that we obtain similar figures for the fraction of time spent in the primary memory system. However, these studies have different goals from ours, in that they are concerned with measuring the effects on total execution time of varying several CPU-level parameters such as issue width, cache size & organization, number of processors, etc. This study focuses on the performance behavior of different DRAM architectures.

3 BACKGROUND

A Random Access Memory (RAM) that uses a single transistor-capacitor pair for each binary value (bit) is referred to as a Dynamic Random Access Memory or DRAM. This circuit is dynamic because leakage requires that the capacitor be periodically refreshed for information retention. Initially, DRAMs had minimal I/O pin counts because the manufacturing cost was dominated by the number of I/O pins in the package. Due largely to a desire to use standardized parts, the initial constraints limiting the I/O pins have had a long-term effect on DRAM architecture: the address pins for most DRAMs are still multiplexed, potentially limiting performance. As the standard DRAM interface has become a performance bottleneck, a number of "revolutionary" proposals [26] have been made. In most cases, the revolutionary portion is the interface or access mechanism, while the DRAM core remains essentially unchanged.

3.1 The Conventional DRAM

The addressing mechanism of early DRAM architectures is still utilized, with minor changes, in many of the DRAMs produced today. In this interface, shown in Figure 1, the address bus is multiplexed between row and column components. The multiplexed address bus uses two control signals—the row and column address strobe signals, $\overline{RAS}$ and $\overline{CAS}$ respectively—which cause the DRAM to latch the address components. The row address causes a complete row in the memory array to propagate down the bit lines to the sense amps. The column address selects the appropriate data subset from the sense amps and causes it to be driven to the output pins.

3.2 Fast Page Mode DRAM (FPM DRAM)

Fast-Page Mode DRAM implements *page mode*, an improvement on conventional DRAM in which the row-address is held constant and data from multiple columns is read from the sense amplifiers. The data held in the sense amps form an "open page" that can be accessed relatively quickly. This speeds up successive accesses to

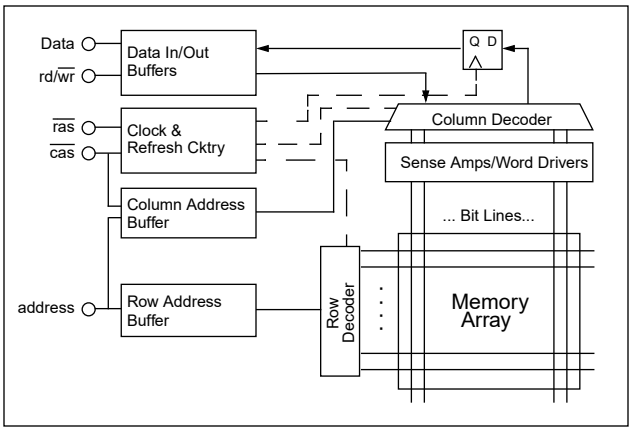

Figure 3: Extended Data Out (EDO) DRAM block diagram. EDO adds a latch on the output that allows CAS to cycle more quickly than in FPM.

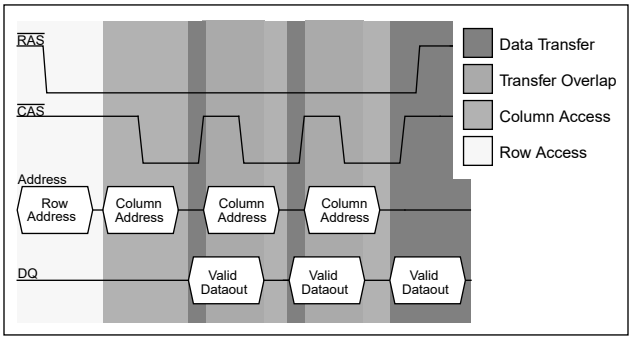

Figure 4: EDO Read Timing. The output latch in EDO DRAM allows more overlap between column access and data transfer than in FPM.

the same row of the DRAM core. Figure 2 gives the timing for FPM reads. The labels show the categories to which the portions of time are assigned in our simulations. Note that page mode is supported in all the DRAM architectures in this study.

3.3 Extended Data Out DRAM (EDO DRAM)

Extended Data Out DRAM, sometimes referred to as hyper-page mode DRAM, adds a latch between the sense-amps and the output pins of the DRAM, shown in Figure 3. This latch holds output pin state and permits the CAS to rapidly de-assert, allowing the memory array to begin precharging sooner. In addition, the latch in the output path also implies that the data on the outputs of the DRAM circuit remain valid longer into the next clock phase. Figure 4 gives the timing for an EDO read.

3.4 Synchronous DRAM (SDRAM)

Conventional, FPM, and EDO DRAM are controlled asynchronously by the processor or the memory controller; the memory latency is thus some fractional number of CPU clock cycles. An alternative is to make the DRAM interface synchronous such that the DRAM latches information to and from the controller based on a clock signal. A timing diagram is shown in Figure 5. SDRAM devices typically have a programmable register that holds a bytes-per-request value. SDRAM may therefore return many bytes over several cycles per request. The advantages include the elimination of the timing strobes and the availability of data from the DRAM each clock cycle. The underlying architecture of the SDRAM core is the same as in a conventional DRAM.

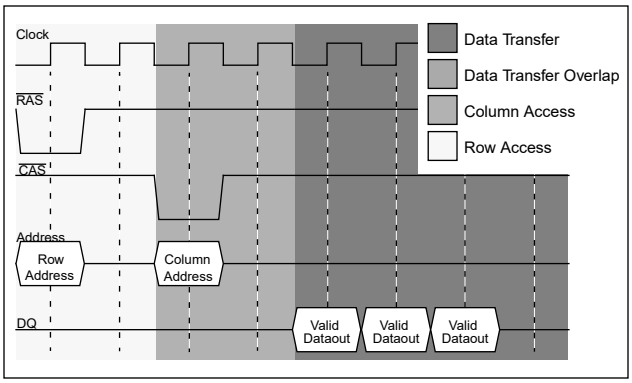

Figure 5: SDRAM Read Operation Clock Diagram. SDRAM contains a writable register for the request length, allowing high-speed column access.

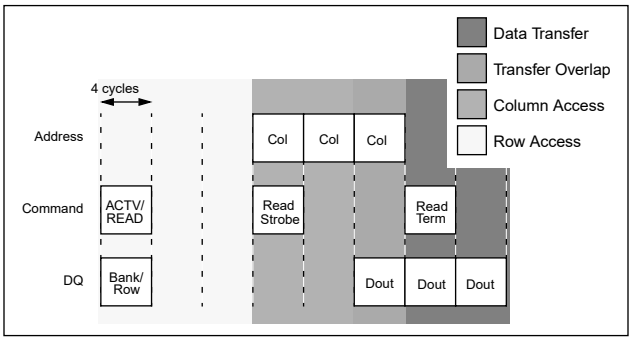

Figure 6: Rambus DRAM Read Operation. Rambus DRAMs transfer on both edges of a fast clock and can handle multiple simultaneous requests.

3.5 Enhanced Synchronous DRAM (ESDRAM)

Enhanced Synchronous DRAM is an incremental modification to Synchronous DRAM that parallels the differences between FPM and EDO DRAM. First, the internal timing parameters of the ESDRAM core are faster than SDRAM. Second, SRAM row-caches have been added at the sense-amps of each bank. These caches provide the kind of improved intra-row performance observed with EDO DRAM, allowing requests to the last accessed row to be satisfied even when subsequent refreshes, precharges, or activates are taking place.

3.6 Synchronous Link DRAM (SLDRAM)

RamLink is the IEEE standard (P1596.4) for a bus architecture for devices. Synchronous Link (SLDRAM) is an adaptation of RamLink for DRAM, and is another IEEE standard (P1596.7). Both are adaptations of the Scalable Coherent Interface (SCI). The SLDRAM specification is therefore an open standard allowing for use by vendors without licensing fees. SLDRAM uses a packet-based split request/response protocol. Its bus interface is designed to run at clock speeds of 200-600 MHz and has a two-byte-wide datapath. SLDRAM supports multiple concurrent transactions, provided all transactions reference unique internal banks. The 64Mbit SLDRAM devices contain 8 banks per device.

3.7 Rambus DRAMs (RDRAM)

Rambus DRAMs use a one-byte-wide multiplexed address/data bus to connect the memory controller to the RDRAM devices. The bus runs at 300 Mhz and transfers on both clock edges to achieve a theoretical peak of 600 Mbytes/s. Physically, each 64-Mbit RDRAM is

Table 1: DRAM Specifications used in simulations

DRAM type	Size	Rows	Columns	Transfer Width	Row Buffer	Internal Banks	Speed	Pre-charge	Row Access	Column Access	Data Transfer
FPMDRAM	64Mbit	4096	1024	16 bits	16K bits	1	–	40ns	15ns	30ns	15ns
EDODRAM	64Mbit	4096	1024	16 bits	16K bits	1	–	40ns	12ns	30ns	15ns
SDRAM	64Mbit	4096	256	16 bits	4K bits	4	100MHz	20ns	30ns	30ns	10ns
ESDRAM	64Mbit	4096	256	16 bits	4K bits	4	100MHz	20ns	20ns	20ns	10ns
SLDRAM	64Mbit	1024	128	64 bits	8K bits	8	200MHz	30ns	40ns	40ns	10ns
RDRAM	64Mbit	1024	256	64 bits	16K bits	4	300MHz	26.66ns	40ns	23.33ns	13.33ns
DRDRAM	64Mbit	512	64	128 bits	4K bits	16	400MHz	20/40ns	17.5ns	30ns	10ns

Table 2: Time components in primary memory system

Component	Description
Row Access Time	The time to (possibly) precharge the row buffers, present the row address, latch the row address, and read the data from the memory array into the sense amps
Column Access Time	The time to present the column address at the address pins and latch the value
Data Transfer Time	The time to transfer the data from the sense amps through the column muxes to the data-out pins
Data Transfer Time Overlap	The amount of time spent performing both column access and data transfer simultaneously (when using page mode, a column access can overlap with the previous data transfer for the same row) Note that, since determining the amount of overlap between column address and data transfer can be tricky in the interleaved examples, for those cases we simply call all time between the start of the first data transfer and the termination of the last column access *Data Transfer Time Overlap* (see Figure 8).
Refresh Time	Amount of time spent waiting for a refresh cycle to finish
Bus Wait Time	Amount of time spent waiting to synchronize with the 100MHz memory bus
Bus Transmission Time	The portion of time to transmit a request over the memory bus to & from the DRAM system that is not overlapped with *Column Access Time* or *Data Transfer Time*

divided into 4 banks, each with its own row buffer, and hence up to 4 rows remain active or open[1]. Transactions occur on the bus using a split request/response protocol. Because the bus is multiplexed between address and data, only one transaction may use the bus during any 4 clock cycle period, referred to as an octcycle. The protocol uses packet transactions; first an address packet is driven, then the data. Different transactions can require different numbers of octcycles, depending on the transaction type, location of the data within the device, number of devices on the channel, etc. Figure 6 gives a timing diagram for a read transaction.

3.8 Direct Rambus (DRDRAM)

Direct Rambus DRAMs use a 400 Mhz 3-byte-wide channel (2 for data, 1 for addresses/commands). Like the Rambus parts, Direct Rambus parts transfer at both clock edges, implying a maximum bandwidth of 1.6 Gbytes/s. DRDRAMs are divided into 16 banks with 17 half-row buffers[2]. Each half-row buffer is shared between adjacent banks, which implies that adjacent banks cannot be active simultaneously. This organization has the result of increasing the row-buffer miss rate as compared to having one open row per bank, but it reduces the cost by reducing the die area occupied by the row buffers, compared to 16 full row buffers. A critical difference between RDRAM and DRDRAM is that because DRDRAM partitions the bus into different components, three transactions can simultaneously utilize the different portions of the DRDRAM interface.

1. In this study, we model 64-Mbit Rambus parts, which have 4 banks and 4 open rows. Earlier 16-Mbit Rambus organizations had 2 banks and 2 open pages, and future 256-Mbit organizations may have even more.
2. As with the previous part, we model 64-Mbit Direct Rambus, which has this organization. Future (256-Mbit) organizations may look different.

4 EXPERIMENTAL METHODOLOGY

For accurate timing of memory requests in a dynamically reordered instruction stream, we integrated our code into SimpleScalar, an execution-driven simulator of an aggressive out-of-order processor [4]. We calculate the DRAM access time, much of which is overlapped with instruction execution. To determine the degree of overlap, and to separate out memory stalls due to bandwidth limitations vs. latency limitations, we run two other simulations—one with perfect primary memory (zero access time) and one with a perfect bus (as wide as an L2 cache line). Following the methodology in [5], we partition the total application execution time into three components: T_P T_L and T_B which correspond to time spent processing, time spent stalling for memory due to latency, and time spent stalling for memory due to limited bandwidth. In this paper, time spent "processing" includes all activity above the primary memory system, i.e. it contains all processor execution time and L1 and L2 cache activity. Let T be the total execution time for the realistic simulation; let T_U be the execution time assuming unlimited bandwidth—the results from the simulation that models cacheline-wide buses. Then T_P is the time given by the simulation that models a perfect primary memory system, and we calculate T_L and T_B: $T_L = T_U - T_P$ and $T_B = T - T_U$. In addition, we consider one more component: the degree to which the processor is able to overlap memory access time with processing

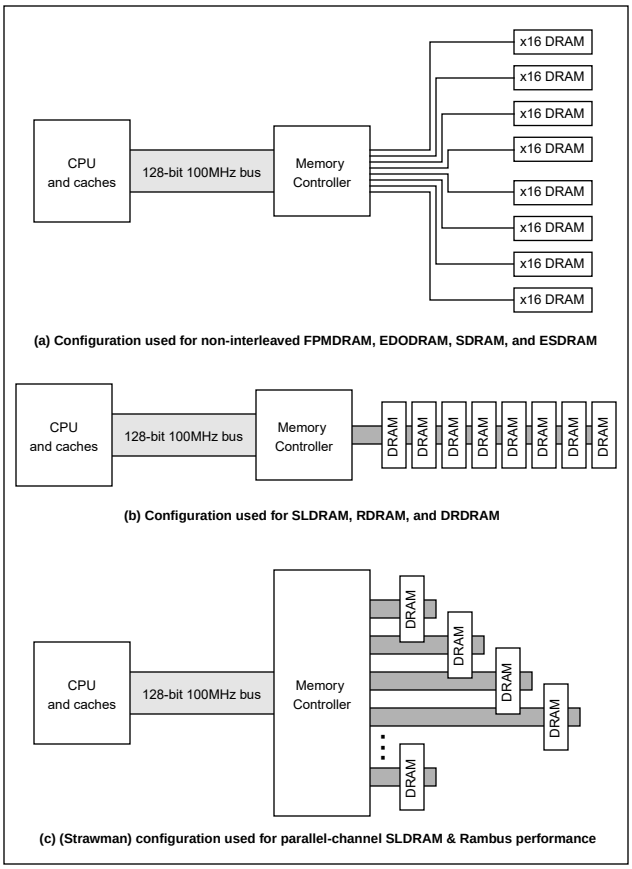

Figure 7: DRAM bus configurations. The DRAM/bus organizations used in (a) the non-interleaved FPM, EDO, SDRAM, and ESDRAM simulations; (b) the SLDRAM and Rambus simulations; and (c) the parallel-channel SLDRAM and Rambus performance numbers in Figure 11. Due to differences in bus design, the only bus overhead included in the simulations is that of the bus that is common to all organizations: the 100MHz 128-bit memory bus.

time. We call this overlapped component T_O, and if T_M is the total time spent in the primary memory system (the time returned by our DRAM simulator), then $T_O = T_P - (T - T_M)$. This is the portion of T_P that is overlapped with memory access.

4.1 DRAM Simulator Overview

The DRAM simulator models the internal state of the following DRAM architectures: Fast Page Mode [35], Extended Data Out [16], Synchronous [17], Enhanced Synchronous [10, 17], Synchronous Link [38], Rambus [31], and Direct Rambus [32].

The timing parameters for the different DRAM architectures are given in Table 1. Since we could not find a 64Mbit part specification for ESDRAM, we extrapolated based on the most recent SDRAM and ESDRAM datasheets. To measure DRAM behavior in systems of differing performance, we varied the speed at which requests arrive at the DRAM. We ran the L2 cache at speeds of 100ns, 10ns, and 1ns, and for each L2 access-time we scaled the main processor's speed accordingly (the CPU runs at 10x the L2 cache speed).

We wanted a model of a typical workstation, so the processor is eight-way superscalar, out-of-order, with lockup-free L1 caches. L1 caches are split 64KB/64KB, 2-way set associative, with 64-byte linesizes. The L2 cache is unified 1MB, 4-way set associative, write-back, and has a 128-byte linesize. The L2 cache is lockup-free but only allows one outstanding DRAM request at a time; note this organization fails to take advantage of some of the newer DRAM parts that can handle multiple concurrent requests. 100MHz 128-bit buses are common for high-end machines, so this is the bus configuration that we model. We assume that the communication overhead is only one 10ns cycle in each direction.

The DRAM/bus configurations simulated are shown in Figure 7. For DRAMs other than Rambus and SLDRAM, eight DRAMs are arranged in parallel in a DIMM-like organization to obtain a 128-bit bus. We assume that the memory controller has no overhead delay. SLDRAM, RDRAM, and DRDRAM utilize narrower, but higher speed busses. These DRAM architectures can be arranged in parallel channels, but we study them here in the context of a single-width DRAM bus, which is the simplest configuration. This yields some latency penalties for these architectures, as our simulations require that the controller coalesce bus packets into 128-bit chunks to be transmitted over the 100MHz 128-bit memory bus. To put the designs on even footing, we ignore the transmission time over the narrow DRAM channel. Because of this organization, transfer rate comparisons may also be deceptive, as we are transferring data from eight conventional DRAM (FPM, EDO, SDRAM, ESDRAM) concurrently, versus only a single device in the case of the narrow bus architectures (SLDRAM, RDRAM, DRDRAM).

The simulator models a synchronous memory interface: the processor's interface to the memory controller has a clock signal. This is typically simpler to implement and debug than a fully asynchronous interface. If the processor executes at a faster clock rate than the memory bus (as is likely), the processor may have to stall for several cycles to synchronize with the bus before transmitting the request. We account for the number of stall cycles in *Bus Wait Time*.

The simulator models several different refresh organizations, as described in Section 5. The amount of time (on average) spent stalling due to a memory reference arriving during a refresh cycle is accounted for in the time component labeled *Refresh Time*.

4.2 Interleaving

For the 100MHz 128-bit bus configuration, the transfer size is eight times the request size; therefore each DRAM access is a pipelined operation that takes advantage of page mode. For the faster DRAM parts, this pipeline keeps the memory bus completely occupied. However, for the slower DRAM parts (FPM and EDO), the timing looks like that shown in Figure 8(a). While the address bus may be fully occupied, the memory bus is not, which puts the slower DRAMs at a disadvantage compared to the faster parts. For comparison, we model the FPM and EDO parts as interleaved as well (shown in Figure 8(b)). The degree of interleaving is that required to occupy the memory data bus as fully as possible. This may actually over-occupy the address bus, in which case we assume that there are more than one address buses between the controller and the DRAM parts. FPM DRAM specifies a 40ns CAS period and is four-way interleaved; EDO DRAM specifies a 25ns CAS period and is two-way interleaved. Both are interleaved at a bus-width granularity.

5 EXPERIMENTAL RESULTS

For most graphs, the performance of several DRAM organizations is given: FPM1, FPM2, FPM3, EDO1, EDO2, SDRAM, ESDRAM, SLDRAM, RDRAM, and DRDRAM. The first two configurations (FPM1 and FPM2) show the difference between always keeping the row buffer open (thereby avoiding a precharge overhead if the next access is to the same row) and never keeping the row buffer open. FPM1 is the pessimistic strategy of closing the row buffer after every access and precharging immediately; FPM2 is the optimistic strategy of keeping the row buffer open and delaying precharge. The dif-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.