

DDR2 and Low Latency Variants

Brian Davis, Trevor Mudge

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

Bruce Jacob, Vinodh Cuppu

Electrical & Computer Engineering
University of Maryland, College Park
{blj, ramvinod}@eng.umd.edu

ABSTRACT

This paper describes a performance examination of the DDR2 DRAM architecture and the proposed cache-enhanced variants. These preliminary studies are based upon ongoing collaboration between the authors and the Joint Electronic Device Engineering Council (JEDEC) Low Latency DRAM Working Group, a working group within the JEDEC 42.3 Future DRAM Task Group. This Task Group is responsible for developing the DDR2 standard. The goal of the Low Latency DRAM Working Group is the creation of a single cache-enhanced (i.e. low-latency) architecture based upon this same interface.

There are a number of proposals for reducing the average access time of DRAM devices, most of which involve the addition of SRAM to the DRAM device. As DDR2 is viewed as a future standard, these proposals are frequently applied to a DDR2 interface device. For the same reasons it is advantageous to have a single DDR2 specification, it is similarly beneficial to have a single low-latency specification. The authors are involved in ongoing research to evaluate which enhancements to the baseline DDR2 devices will yield lower average latency, and for what type of applications. To provide context, experimental results will be compared against those for systems utilizing PC100 SDRAM, DDR133 SDRAM, and Direct Rambus (DRDRAM).

This work is just starting to produce performance data. Initial results show performance improvements for low-latency devices that are significant, but less so than a generational change in DRAM interface. It is also apparent that there are at least two classifications of applications: 1) those that saturate the memory bus, for which performance is dependent upon the potential bandwidth and bus utilization of the system; and 2) those that do not contain the access parallelism to fully utilize the memory bus, and for which performance is dependent upon the latency of the average primary memory access.

1 INTRODUCTION

The expanding performance gap between processor speeds and primary memory speeds has been characterized as a “memory wall.” Until recently this problem has been attacked by designing the CPU to tolerate long memory latencies, examples include: prefetching, out-of-order execution, lock-up free caches, speculative loads, data prediction, and memory-request reordering. Now, we are beginning to see significant changes to DRAM architecture in an attempt to improve both the bandwidth and the latency of the primary memory system from the other side of the “wall.” Most of these approaches use conventional DRAM cores but take advantage of the large amount of data available on the sense amps during an access to dramatically improve bandwidth on and off the memory device. Examples include: the Direct Rambus architecture, the Multibanked approach of Mosys, and SRAM added to the DRAM device (Virtual Channel (VC) SDRAM - VCSDRAM,

Enhanced SDRAM - ESDRAM, and Cached DRAM - CDRAM). The speed of the DRAM core is also being improved, but not at the rate of the processor core [Wulf 1995]. This paper will examine recent commercial DRAM proposals, and focus upon the work being done by the JEDEC low-latency DRAM working group (LLWG), to evaluate them.

Many of the mechanisms used in processors to tolerate long DRAM latency exploit concurrency to improve performance. Parallelism in the memory system may potentially achieve performance gains, similar to those achieved by parallelism in instruction execution. Older DRAM architectures only service a single memory request at a time. Because older processors stall on every cache miss, this arrangement posed no significant problems. However, modern processor designs have added support for concurrency; as a result, modern systems are capable of performing multiple operations in parallel as well as performing independent work during the servicing of a cache miss. To fully exploit this, modern DRAM controllers and devices must support some degree of concurrency to avoid serializing and negating the processor's support for access concurrency. Today's most advanced DRAMs support some degree of concurrency directly by allowing two to three pipelined requests to independent banks. It is not known how much performance this arrangement buys, and what degree of concurrency the DRAM system must support to obtain reasonable results.

2 DRAM Architectures

A memory that uses a single transistor-capacitor pair for each binary value (bit) is referred to as a Dynamic Random Access Memory (DRAM). DRAM are the choice for primary memory in the majority of modern computer systems due to the cost per bit, the access latency, and the non-mechanical, solid-state, nature of the device. As conventional DRAM has become a performance bottleneck, a number of modernized evolutionary as well as "revolutionary" proposals [Przybylski 1996] have been made. In most cases, the innovative portion is the interface or access mechanism, while the DRAM core remains essentially unchanged. Common changes include some form of caching of recent sense amp data and/or interface changes like double data rate (DDR) signalling where unique data is driven and received on both the rising and the falling edge of the clock.

The goals of the DDR2 specification are to create a common, standardized specification for the next generation of SDRAM that will reduce the impact of DRAM latency upon microprocessor performance. A number of DRAM architectures will be discussed and compared in this paper, the DDR2 architecture, two cache-enhanced proposals based upon the DDR2 interface, multiple speed grades of SDRAM, and DRDRAM. The following sections will briefly discuss each of these DRAM architectures.

2.1 SDRAM

The majority of desktop consumer PC's being shipped in 1Q2000 utilize PC100 SDRAM. These designs will be migrated to PC133 (higher clock speed) and DDR133 (or PC2100) that will allow data transitions on both edges of the bus clock. This evolution will allow potential bandwidth to increase from 0.8 GB/s (PC100) to 2.133 GB/s (DDR 133) [Dipert 2000]. SDRAM is a synchronous adaptation of the prior non-synchronous FPM and EDO DRAM architectures. As such, SDRAM is the first DRAM architecture with support for access concurrency. Earlier non-synchronous DRAM had to support access concurrency via externally controlled interleaving. Modern SDRAM devices will continue to share the DRAM market with DRDRAM until and during the emergence of the DDR2 devices.

Table 1: Overview of DRAM Characteristics

	PC133	DDR2	DDR2_VC	DDR2_EMS	DRDRAM
Potential Bandwidth	1.064 GB/s	3.2 GB/s			1.6 GB/s
Interface	<ul style="list-style-type: none"> • Bus • 64 Data bits • 168 pads on DIMM • 133 Mhz 		<ul style="list-style-type: none"> • Bus • 64 Data bits • 184 pads on DIMM • 200 Mhz 		<ul style="list-style-type: none"> • Channel • 16 Data Bits • 184 pads on RIMM • 400 Mhz
Latency to first 64 bits (Min. : Max)	(3 : 9) cycles (22.5 : 66.7) nS	(3.5 : 9.5) cycles (17.5 : 47.5) nS	(2.5 : 18.5) cycles (12.5 : 92.5) nS	(3.5 : 9.5) cycles (17.5 : 47.5) nS	(14 : 32) cycles (35 : 80) nS
Latency Advantage			<ul style="list-style-type: none"> • 16 Line Cache / Dev; 1/4 row line size 	<ul style="list-style-type: none"> • Cache Line per bank; line size is row size 	<ul style="list-style-type: none"> • Many smaller banks • More open pages
Advantage	<ul style="list-style-type: none"> • Cost 	<ul style="list-style-type: none"> • Cost 	<ul style="list-style-type: none"> • Less Misses in "Hot Bank" 	<ul style="list-style-type: none"> • Precharge Always Hidden • Full Array BW Utilized 	<ul style="list-style-type: none"> • Narrow Bus • Smaller Incremental granularity
Disadvantage			<ul style="list-style-type: none"> • Area (~3-6%) • Controller Complexity • More misses on purely linear accesses 	<ul style="list-style-type: none"> • Area (~1.4%) • More conflict misses 	<ul style="list-style-type: none"> • Area (~10%) • Sense Amps shared between adjacent banks

2.2 DDR2 (JEDEC spec)

The DDR2 specification under development by the JEDEC 42.3 Future DRAM Task Group is intended to be the follow-on device specification to SDRAM. While DDR2 will have a new pin-interface, and signalling method (SSTL), it will leverage much of the existing engineering behind current SDRAM. The initial speed for DDR2 parts will be 200 MHz in a bussed environment, and 300Mhz in a point-to-point application, with data transitioning on both edges of the clock. We will focus upon the bussed environment in this paper, as that is the structure of the memory system in a general purpose computer. Other significant changes from past SDRAM architectures are: a fixed burst length of 4 data cycles, a programmable additive latency for enabling posted-CAS¹ transactions, a write latency not equal to one, differential clock signaling and micro-BGA packaging. The DDR2 specification is not finalized at this point, but the information contained here is based upon the most recent drafts for DDR2 devices and conversations with JEDEC members.

2.3 DDR2 with VC Enhancement

NEC has proposed a low-latency variant of the DDR2 SDRAM, based upon their Virtual Channel (VC) architecture. The intent of the VC architecture is to reduce the average latency of a DRAM

1. Posted-CAS allows the CAS information to be buffered for a variable number of cycles at the DDR2 interface before being sent to the core for higher bus utilization. Further discussion of Posted-CAS appears in Section 3.4

access via two mechanisms: (1) to use on-chip SRAM cache to reduce the number of accesses to the DRAM array; and (2) to organize this SRAM cache in such a manner that it allows for multiple open channels to the same bank, reducing latency in cases where you have two or more access streams alternating between different rows in the same bank [NEC 1999]. The VC implementation for DDR2 has yet to be precisely determined, with the number of banks being 2 or 4, the number of cache lines 8, 16 or 32, the size of the cache lines almost certainly 1/4 the DRAM row size, and the associativity of the cache lines being anywhere from fully-associative through 4-way set associative. The VC architecture refers to each of these cache lines as a “Virtual Channel” and relies upon the memory controller to manage the allocation, placement, and write-back. This allows the controller to maintain or write-back dirty channels, or allocate the channels based on any policy from an LRU algorithm to a bus-master oriented algorithm.

2.4 DDR2 with ESDRAM-lite Enhancement

Enhanced Memory Systems (EMS) has proposed a low-latency variant of the DDR2 SDRAM, based on their currently available ESDRAM products. The idea behind this product is again to use a small amount of SRAM as a cache located on the DRAM die. The expectation is that the high bandwidth between the DRAM array and the on-chip cache would reduce the average latency of accesses to the DRAM device. The caching structure proposed by EMS is a single direct-mapped SRAM cache line, the same size as the DRAM row, associated with each bank. This allows the device to service accesses to the most recently accessed row, regardless of whether refresh has occurred and enables the precharge of the DRAM array to be done in the background. This architecture also supports a no-write-transfer mode within a series of interspersed read and write accesses. The no-write-transfer mode allows writes to occur through the sense-amps, without affecting the data currently being held in the cache-line associated with that bank [EMS 2000].

2.5 Direct Rambus (DRDRAM)

Direct Rambus DRAMs use a 400 Mhz 3-byte-wide channel (2 for data, 1 for addresses/commands). DRDRAM devices use DDR signalling, implying a maximum bandwidth of 1.6 Gbytes/s, and these devices have many banks in relation to SDRAM devices of the same size. Each sense-amp, and thus row buffer, is shared between adjacent banks. This implies that adjacent banks cannot simultaneously maintain an open-page, or maintain an open-page while a neighboring bank performs an access. 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 buffer [IBM 1999].

Future Memory Devices

Five years ago the limitations of DRAM were much less significant to system performance, and there were few systems level investigations into what was viewed as a commodity component. With the recognition that DRAM more significantly contributes to system performance, there has been a jump in the number proposals intended improve the latency and/or bandwidth of DRAM. If we look ahead five years, past the above developments, a wide range of possibilities open up from a continuing evolution of SDRAM to something as exotic as a 3 dimensional optical carrier [C-3D 2000]. In this range of possibilities, an alliance with interesting implications is the one formed by and containing Intel, Samsung Electronics, Hyundai Microelectronics, Micron Technology, Infineon Technologies, and the new NEC Hitachi Memory. The last four of these five companies account for more than 80 percent of current global DRAM sales, and their Advanced DRAM Technology (ADT) alliance has a

stated goal of developing a DRAM architecture to succeed or compliment the DDR2 architecture [Robertson 2000].

3 DDR2 - how does it bridge the wall?

The DDR2 specification is an attempt to provide a common design target for many DRAM vendors and system architects. It is hoped that early standardization will minimize design fragmentation and thus benefit the consumer through lower prices. While it contains significant changes, DDR2 utilizes much of the same architecture as the preceding SDRAM. It is interesting to examine the manners in which it differs from direct evolution of existing SDRAM.

3.1 Bandwidth

Bandwidth alone can not solve the problem characterized as the memory wall. Some performance impact is because the processor core sits idle during the DRAM latency for critical data before it can continue execution [Cuppu 1999]. Bandwidth can be increased for any interface or architecture by increasing the bus width or replicating the channel. Thus we see a proposal for the next generation Alpha microprocessor that contains four DRDRAM channels for a potential primary memory bandwidth of 6.4 GB/s [Gwennap 1998]. However, for a fixed size working set and reasonable latency, an interface with higher bandwidth is going to be utilized a lower percentage of the time, meaning an access issued during that time is less likely to be queued up waiting for prior requests to be serviced. In this way bandwidth can have some effect upon observed processor latency.

Additionally, there are applications that are bandwidth limited due to an access pattern which renders caches of limited usefulness. For these applications, and due to the reduced queuing latencies observed with increased bandwidth, increasing the bandwidth to the primary memory system will increase performance. The downside to increasing the bandwidth by expanding bus width is an increase in system cost and the fact that memory must be added in larger granularities. Increasing the bus speed will also increase the bandwidth, without the width costs, but results in higher complexity and higher power consumption. DDR2 with a 64 bit wide desktop bus, switching data signals at 400 Mhz, has a potential bandwidth of 3.2 GB/s; this surpasses any current DRAM architecture. Further, if server bus widths of 256 bits remain constant when DDR2 is introduced to the server architecture, potential bandwidths of 12.8 GB/s will force the re-design of the processor front-side bus to support this throughput.

3.2 Access Concurrency

One of the questions we seek to answer is how much performance may improve with support for concurrency in the DRAM system, and what degree of concurrency the DRAM system must support to obtain reasonable results. There have been no published limit studies that determine how much DRAM-level concurrency a modern CPU can exploit, and there has been no published comparison between DRAM systems with/without support for multiple simultaneous transactions. What has been published suggests that there are gains to be had exploiting concurrency at the DRAM level, but they likely require sophisticated hardware [Cuppu 1999]. DDR2 has support for concurrency in the DRAM system, but no more so than other SDRAM architectures. Additionally, since DDR2 is targeted at large devices (greater than 256 Mbit) with only four banks, it may be that the amount of attainable concurrency is less than that of architectures containing more numerous smaller banks with the same address space. Low latency variants are intended to support for more concurrency, as

they can service requests out of the cache while longer-latency accesses are serviced by the DRAM core.

3.3 Access Granularity of Four

The DDR2 specification, in the most recent draft, sets the access granularity of all reads, and the maximal size write of DDR2 devices to 4 data cycles, or 2 clock cycles. This is in contrast to PC100 parts that allow bursts of 2, 4, 8, or full-page [IBM 1998] and Direct Rambus parts that allow bursts of any power-of-2 octcycles (128 bit quantities) up to the page size [IBM 1999]. What impact does this access granularity limitation impose upon the DDR2 parts? If we examine a 256 byte transaction which would require 4 CAS requests using a burst size of 8 in a PC100 environment, or a single COL packet in a DRDRAM environment, the same transaction will require 8 CAS requests using the fixed access size of 4 in the DDR2 environment. Data bus usage (in bytes) is constant in this example; however the fraction of time that the address bus is utilized increases for DDR2. It remains to be seen if this additional loading of the address bus will impact performance. It may potentially reduce the ability to perform tasks that do not require the data bus (i.e. refresh) in the background while performing reads from independent banks. One motivation for taking this approach is that the DDR2 interface does not support interrupting transactions once they have been initiated. In PC100 or DRDRAM systems bursting a full-page of data, it may be required to terminate the transaction early for another, higher priority, transaction. Since the DDR2 transactions are smaller, the support for termination of in-flight accesses need not be present.

3.4 Additive Latency (Posted-CAS) and Write Latency

The posted-CAS enhancement of the DDR2 specification hinges upon the addition of a parameter called Additive Latency (AL). The AL parameter enables the RAS and CAS packets in a DRAM access to be driven onto the address bus in adjacent clock cycles. The DDR2 device interface then holds the CAS command for the AL latency prior to issuing it to the core. This allows for a more logical ordering of the access packets occurring on the address bus and may allow for a higher address bus utilization than without the posted-CAS enhancement.

Write Latency (WL) for a DDR2 device is not a single cycle, as it is for current SDRAM, it is instead set to Read Latency (RL) minus 1, i.e. $WL = (RL - 1)$. RL is programmable in bus cycles, providing flexibility for devices with differing core parameters. This relationship between RL and WL has the property of eliminating the idle data bus cycles associated with transitioning from a write to a read command in the current SDRAM bus protocols. Similar to the posted-CAS enhancement, a WL greater than 1 also has a simplifying effect upon the access stream timing diagrams, and allows for higher utilization of the data bus.

3.5 Interface concurrency

Interleaving in older non-synchronous DRAM systems allows for a limited amount of concurrency, supported by a memory controller that has independent address busses to each bank of DRAM devices and that controls a mux to share the data bus between all banks. This configuration is costly. It is the intent of synchronous memory systems to enable a similar level of concurrency through the shared synchronous interface. In order to achieve the highest amount of concurrency in a memory system, regardless of technique, accesses adjacent in time should be directed to unique banks. In interleaved memory systems, this involves locating different portions of the memory space in different banks. Similarly, in a synchronous environment it is useful to re-map the processor address space into a DRAM address space with a modified layout. Performing the best re-mapping is

dependent upon a number of parameters: the size of the DRAM pages, the number of banks in the system, the number of devices in the system, and the lowest level cache size and mapping policy [Lin 1999]. The intent of this remapping is to locate banks that are likely to be accessed in close temporal proximity to independent banks in the DRAM address space to enable the highest amount of overlap between the accesses. Remapping of the address space can be accomplished by the DRAM controller [Rambus 1999]. When performing comparisons of DRAM architectures, it is important to consider the impact of re-mapping the address space on maximizing concurrency.

4 Low-Latency DDR2

What we refer to as low-latency DDR2 are two proposals from NEC and EMS to add an SRAM caching structure onto the DDR2 device. There are a number of characteristics that are impacted by the proposed enhancements: area, performance, concurrency, and system complexity. In this section we will examine the differences between the two proposals.

4.1 Area impacts of SRAM on die

The DRAM architectures that we have studied have all made different tradeoffs with regard to area. Beginning with DRDRAM, we examine a design that places many banks-- 32 4Mbit banks in a single 128 Mbit device -- upon a single die [IBM 1999]. Completely apart from other aspects of the Direct Rambus architecture, we would expect that having many banks would yield a higher degree of concurrency in the memory system, because there exists a larger number of independent banks within a fixed address space. However a large number of banks increases area in two ways: (1) the number of redundancy bits located in each array, to provide for high yield, increases as a proportion of total bits, and (2) the amount of area dedicated to sense amplifiers, control circuitry and routing increases. If we examine the VC architecture variation of the DDR2 device, it is clear that this device incurs an area penalty associated with putting an SRAM cache on chip. However NEC claims that by limiting the size of the channel to 1/4 of a DRAM row (or page) size, they can minimize the impact of routing upon the area. The EMS low-latency DDR2 device also incurs an area penalty from SRAM, but since the line size of the SRAM cache match the DRAM array row size, the impact of routing upon area may be more than the VC proposal, although the difference is difficult to assess.

4.2 Cache line size impact

The different DRAM architectures each have a different effective cache line size depending upon a number of system parameters, not the least of which is the number of devices used to cover the system bus width. This is not under the control of the DRAM architect but is rather a question of system requirements. When there is no explicit SRAM cache on the DRAM device, the controller may employ an Open-Page (OP) policy effectively utilizing the sense amplifiers as a cache, subject to the limitations of refresh and latency for precharge when data is not found in the open-page. In this case the effective cache line size is the same size as a DRAM system page. The VC low-latency DDR2 proposal has multiple short cache lines, with a minimum associativity of 4. The EMS proposal has a single cache line associated with each bank, and the cache line is the same size as the DRAM array row size. In applications where there are multiple concurrent access streams directed at the same bank, the VC proposal yields better performance because they do not eject useful data due to cache conflicts. In applications with a single linear stream, or where each stream resides in an independent bank, the EMS proposal yields better performance, due to the longer cache lines and the resulting fewer accesses to the DRAM core.

4.3 Controller Complexity

One aspect of the DRAM system cost that has been under adequately investigated is the design and complexity level of the controller. Research on high performance (and complexity) DRAM controllers is being done by the Impulse group at University of Utah [Carter 1999]. However, with the increase in low-cost system configurations there is motivation to increase controller performance as much as possible while limiting the circuit complexity. In order to function, the controller must maintain information about the state of the DRAM system. A memory controller that utilizes a Close-Page-Autoprecharge (CPA) policy needs to maintain little information about the state of the DRAM -- essentially the sizes of each DRAM, and the banks on those devices. When a memory controller utilizes an Open-Page (OP) policy, it also needs to remember which row of data is currently "open" for each bank. This allows the controller to issue a CAS-only access rather than re-accessing the DRAM core. Not all DRAM systems require precisely the same number of bits of storage to maintain state about the managed memory system. Because of the significantly larger number of banks in a DRDRAM configuration, the controller must have the storage capacity to maintain the open page information for many more banks. Finally, because the VC implementations have channels, in addition to banks, for which the controller must maintain the state, these devices also require more state-space storage on the controller device.

Beyond simply the storage required by a device, complexity is hard to quantify. If we hypothesize that the more distinct operations that a device is capable of, the higher the level of controller complexity, then it could be said that: the packet interface channel of DRDRAM, the caching mechanisms of EMS or VC, or the abilities to perform address space remapping, access re-ordering or access coalescing at the controller, each increases the complexity of the controller design. Similarly, the lack of support within the DDR2 interface for interruption of accesses in flight may decrease the complexity of that controller design. With regard to management of the SRAM cache on a DRAM device, as the number of cache lines or the associativity of the cache lines increases, the complexity of the controller must also increase. Once the functionality and implementation of a memory controller is defined, it is difficult to estimate exactly how much area any additional complexity will occupy without performing the design task.

5 DRAM simulation methodologies

In our examination of the performance differences between DRAM memory systems, we have employed two primary methods of simulation.

5.1 Trace Driven

Trace driven simulation is performed by gathering a trace of system activity, at the instruction level, or in the case of DRAM investigations at the memory controller-DRAM interface; then at some later point using that trace to exercise a simulation engine modeling the target system. One primary advantage of trace driven simulation is speed. If the trace contains strictly DRAM activity, the simulation of the memory system alone can be done very rapidly. The disadvantage of trace driven simulation is that the parameters of the simulation: processor speed, bus speed, bus bandwidth, number of DRAM devices, and so forth, can not be varied without impacting the accuracy of the trace. Another potential disadvantage is the lack of accurate timing information. The worst case would be to have no timing information in the trace, the best situation being timestamps based on a DRAM system similar to that of the target system. Absence of timestamps during simulation has the consequence that the DRAM access stream is compacted in time to the minimal amount of time

required to perform all accesses. However, the presence of timestamps does not necessarily guarantee accuracy, even when using many of the same system parameters. If the DRAM latency has significantly changed from gathered to target systems, the interspatial access timings may be significantly altered, again affecting the accuracy of the DRAM trace.

5.2 Execution Driven

Execution driven simulation is performed by taking a program, typically a compiled binary, and emulating execution of that binary through a simulator such as SimpleScalar [Burger 1997]. Over the course of simulation: (1) this tool will generate all of the DRAM accesses required for program execution; (2) these DRAM accesses can be processed by a DRAM model; and (3) statistics can be produced by the model. The execution driven model is preferred for a number of reasons: The parameters to the simulation can be easily changed, and the DRAM access stream remains accurate. Hypothetical systems that have not been produced, and from which it would therefore be impossible to gather a trace, can be modeled. And, finally while it may be technically difficult or impossible to gather a trace for a application of interest, a binary may typically be produced or acquired. A significant disadvantage of execution driven simulation is that this level of simulation requires more system elements be modeled and thus is very time consuming. One consequence of being dependent upon the execution driven simulator, is that most of these toolsets model the execution of a single binary on a single processor, ignoring OS interaction, multiprocess effects, and SMP configurations.

Since each of these approaches has advantages, it is best to make use of both in examining DRAM structures. Traces can be used to debug simulation models - particularly synthetic traces whose effect upon the DRAM system can be calculated analytically. Traces are also useful for calculating the hit rates on any SRAM structures the DRAM device may hold and for determining the upper limit of the DRAM's bandwidth. Subsequently, execution driven simulations can be done for more focussed studies that require a higher degree of confidence in the results.

6 Results

We have performed preliminary studies of all DRAM discussed herein, as well as competing architectures, using both the trace driven and execution driven simulation. It is our intent to present results comparing the performance of each of these DRAM architectures, on a variety of benchmarks. The framework of our simulations allow for the variation of many parameters. Within the domain of SimpleScalar items such as the number of parallel pipelines, processor clock speed, cache configuration, number of MSHR's, L1 and L2 cache configurations may all be easily modified. Early in simulations we found it impractical to vary all parameters. For this reason we concentrated on parameters most closely associated with the DRAM and controller. For the execution driven simulations, the data presented here is all based upon an 8-way 5 Ghz superscalar microprocessor, with two ports to memory, 16MSHR, 32K I / 32K D split L1 caches, and a 256K unified L2 cache. For the trace driven simulations, the processor and caches are dependant upon the system in which the trace was gathered, but timestamps are ignored due to the differences in primary memory architectures.

The benchmarks simulated are intended to cover all types of applications. The execution driven benchmarks are drawn from a number of sources. Some are members of the SPEC95 suite [SPEC 1995], some are members of the Mediabench suite [Lee 1997], there are two versions of the McCalpin Stream benchmark [McCalpin 2000], each of which was compiled with different optimizations, and finally there is random_walk, that was hand coded as an application which is

bandwidth limited, but has an unpredictable (i.e. not streaming) access pattern. The traces used are drawn from two places, IBM online-transaction-processing (OLTP) trace gathered on a one-way and 8-way SMP, and Transmeta traces for Access, Cpumark, Gcc and Quake. These are certainly not the only benchmarks applicable to this type of a study, but it is our hope that these selections encompass enough breadth that all application behavior patterns are represented.

The intent of controller address remapping is to reduce the number of adjacent (in time) accesses, that map to unique rows in the same bank. This can reduce average latency because adjacent accesses mapping to unique rows in the same bank can not be overlapped to the same degree as accesses to independent banks.

It is appropriate to compare each of the DRAM technologies under circumstances where each is performing at the best level achievable with a reasonable controller. For this reason, prior to comparing all of the technologies against each other, we will perform comparisons covering multiple potential controller policies for each DDR2 architecture.

6.1 DDR2

The first set of architectural simulations shown is for DDR2 as specified by the JEDEC 42.3 Future DRAM Task group. For this architecture, the two DRAM controller parameters that can be varied given the current simulation environment are the basic controller policy, either OP or CPA, and whether or not the controller performs remapping. Figure 1 shows the normalized execution time

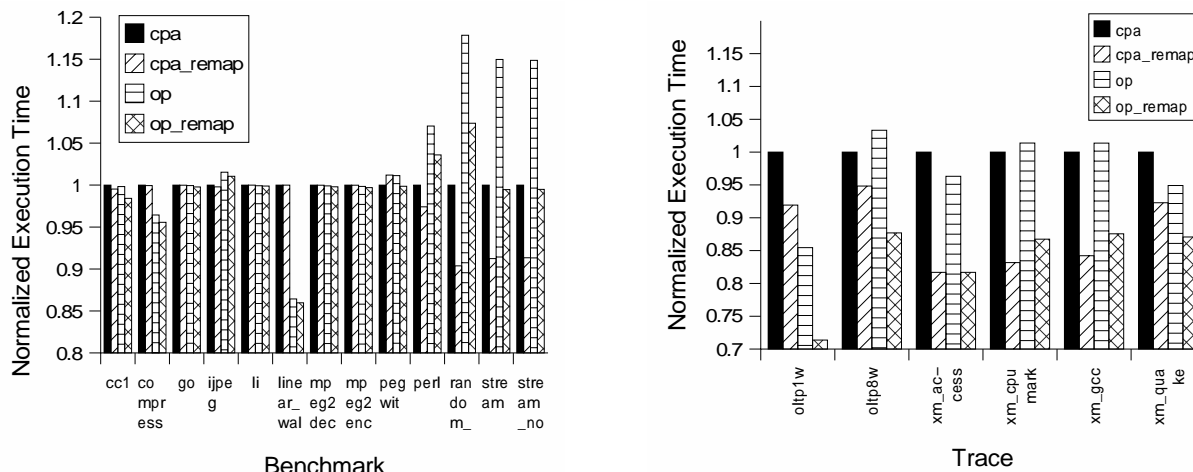


Figure 1: DDR2 Runtime with execution times normalized to CPA policy. Left : Execution driven simulations; Right : Trace driven simulations.

for conventional DDR2 under the four possible permutations of these two parameters. The runtimes have been normalized to CPA in order to enable comparison between the execution time of the various benchmarks.

These simulations show that remapping the address ranges from the processor address space to the DRAM address space yields an increase in performance for both controller policies. In examining the controller policy, these results are indeterminate on whether CPA or OP yields higher performance. For some benchmarks the best performance configuration is cpa_remap, for other benchmarks it is op_remap. The two policies are implemented such that, for CPA to result in a lower execution time than OP, it must be that adjacent accesses to the same bank do not frequently access data from the same row. For bandwidth limited applications, it appears that these applications infrequently perform repeated accesses to the same row, because the CPA policy yields higher

performance. This is at least partially because for Stream there are always two or more streams being accessed so neighboring accesses are not going to be from the same stream. This is an indication that access re-ordering may be able to improve performance significantly.

These results do not show a single controller policy that yields the highest performance across all benchmarks. However, for future comparisons, the open-page with remapping controller configuration was chosen as the policy by which DDR2 will be compared.

6.2 DDR2 EMS

The second architecture examined is DDR2 with the proposed EMS cache enhancements. For DDR2EMS, execution time data was normalized against the OP controller policy. It is foreseen that this policy would never be used with an EMS architecture DRAM, because there is no motivation to maintain an open-page with data in the sense-amps when those sense amps are immediately followed by SRAM latches; the CPA policies provide the advantage over OP that the precharge latency may be hidden in all cases where adjacent accesses do not map to unique rows of the same bank. However for curiosity's sake, the OP policy was simulated, and chosen as the normalization reference. The other four data sets shown all utilize the CPA controller policy, but are unique in whether the controller performs an address re-mapping, and whether writes are performed through the row cache (write-transfer) or not (no-write-transfer). In the no-write-transfer (NWX) case memory consistency is guaranteed by hardware that checks to see if the target line is in the row-cache, and if so this hardware updates the row-cache contents. Figure 2 shows the simulation results for each of these five

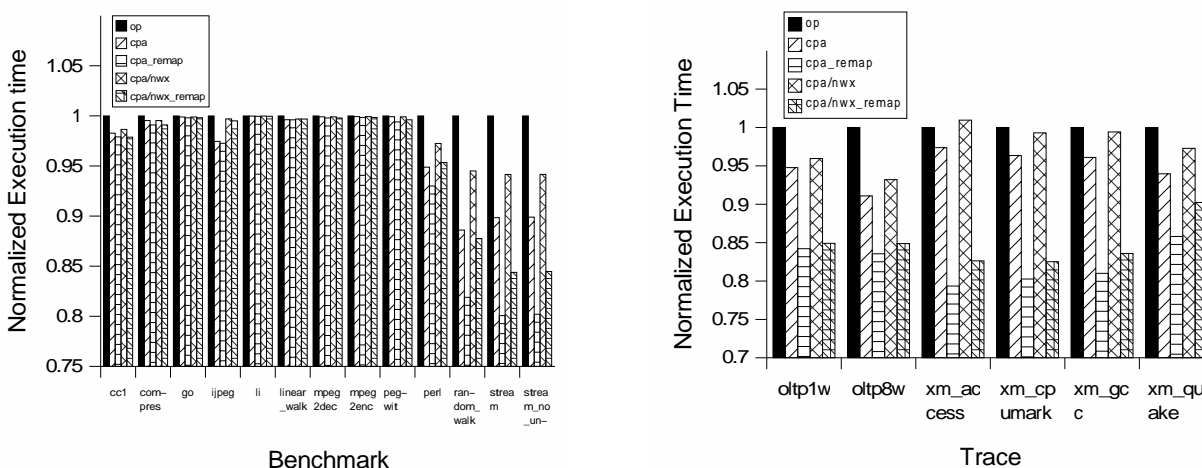


Figure 2: DDR2 w/ EMS cache enhancement execution times normalized to OP policy execution time. Left: Execution driven simulations; Right : Trace driven simulations.

controller policies.

For the DDR2EMS architecture, it appears that there are two controller policies, each of which may yield the highest performance, dependent upon the benchmark. For the majority of benchmarks, the cpa_remap policy has the lowest execution time, however, for a few cases (cc1, mpeg2enc) the cpa/nwx_remap policy, in which all writes are done using the no-write-transfer mechanism yields the lowest execution time. In the two occasions where 100% no-write-transfer writes have superior performance it is only a slight margin. Once again we see from these results that address remapping accounts for a significant performance differential. The performance improvements due to address remapping are most significant for the EMS approach because the EMS approach has the fewest and largest row cache(s) of any of the system configurations, and the

intent of the address remapping is to reduce the fraction of adjacent access that map into unique rows (again large) in the same bank.

In the case of DDR2EMS, the controller policy (of the five examined) that consistently yields the highest performance is `cpa_remap`. This is therefore the controller policy that will be used when comparing DDR2EMS to other DRAM architectures.

6.3 DDR2 VC

For the Virtual Channel architecture, only CPA controller policies were examined, the VC architecture which transfers data in quarter page granularity from the sense-amps to the Virtual Channels does not support maintaining open pages in the DRAM array. For the Virtual Channel architectures the two controller parameters varied are the allocation policy of the channels, either random or least-recently-used (LRU) as well as whether the controller performed an address remapping. It should be noted that a number of items are still undefined in the Virtual Channel specification for DDR2 including the level of associativity between the channels and banks of the device. Both of the allocation policies discussed here assume fully associative channels. Future work may include examination of allocation policies under the constraints of partial associativity. Figure 3

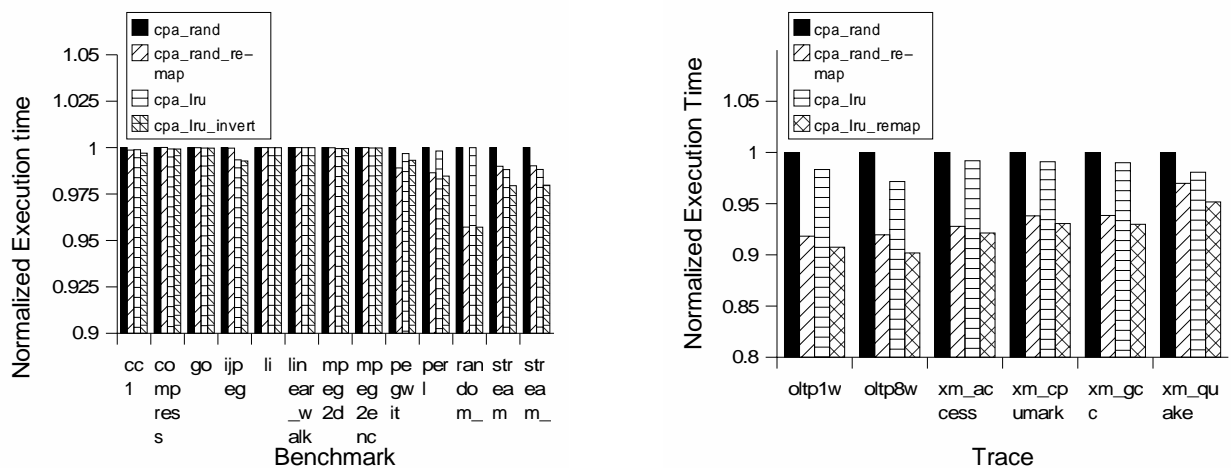


Figure 3: DDR2 w/ VC cache enhancement execution times normalized to `cpa_rand` policy.

shows the results of the simulations for the four permutations of controller policies examined.

It may be noticed that there is less variation (10% in the traces, 5% in the execution driven) for the Virtual Channel architecture than for either baseline DDR2 or DDR2EMS. Once again, the address remapping consistently improves performance for all controller policies, demonstrating that a reduction in the fraction of adjacent accesses targeting the same bank allows for a higher degree of access parallelism. Not surprisingly, it can also be observed in these results that the LRU channel allocation policy provides improved performance over random allocation. Virtual Channel proponents claim that the ability of the memory controller to perform the channel allocation may provide improved performance by allocating channels based upon the bus master initiating the access, unfortunately our simulation methodologies did not provide an opportunity to investigate this style of channel allocation.

The Virtual Channel policy that consistently demonstrates the highest level of performance is CPA with an LRU channel allocation and controller address remapping. This controller configuration will therefore be used for the comparisons between DRAM architectures.

6.4 Architectural Comparison

From the prior three sections, we can choose the best performance controller policy for each of the three DDR2 architectures. For the results shown in Figure 4, the `op_remap` configuration will be used for DDR2, the `cpa_remap` configuration will be used for DDR2EMS, and the `cpa_lru_remap` configuration will be used for DDR2VC. In addition to the three variants of the DDR2 architecture PC100, DDR133 and Direct Rambus (DRDRAM) DRAM will be included for comparison. For these three architectures, an open-page controller policy is used, but as of this time, the controller model for these technologies does not support address remapping, so that is not included. Figure 4 shows the results of simulations for six different DRAM architectures.

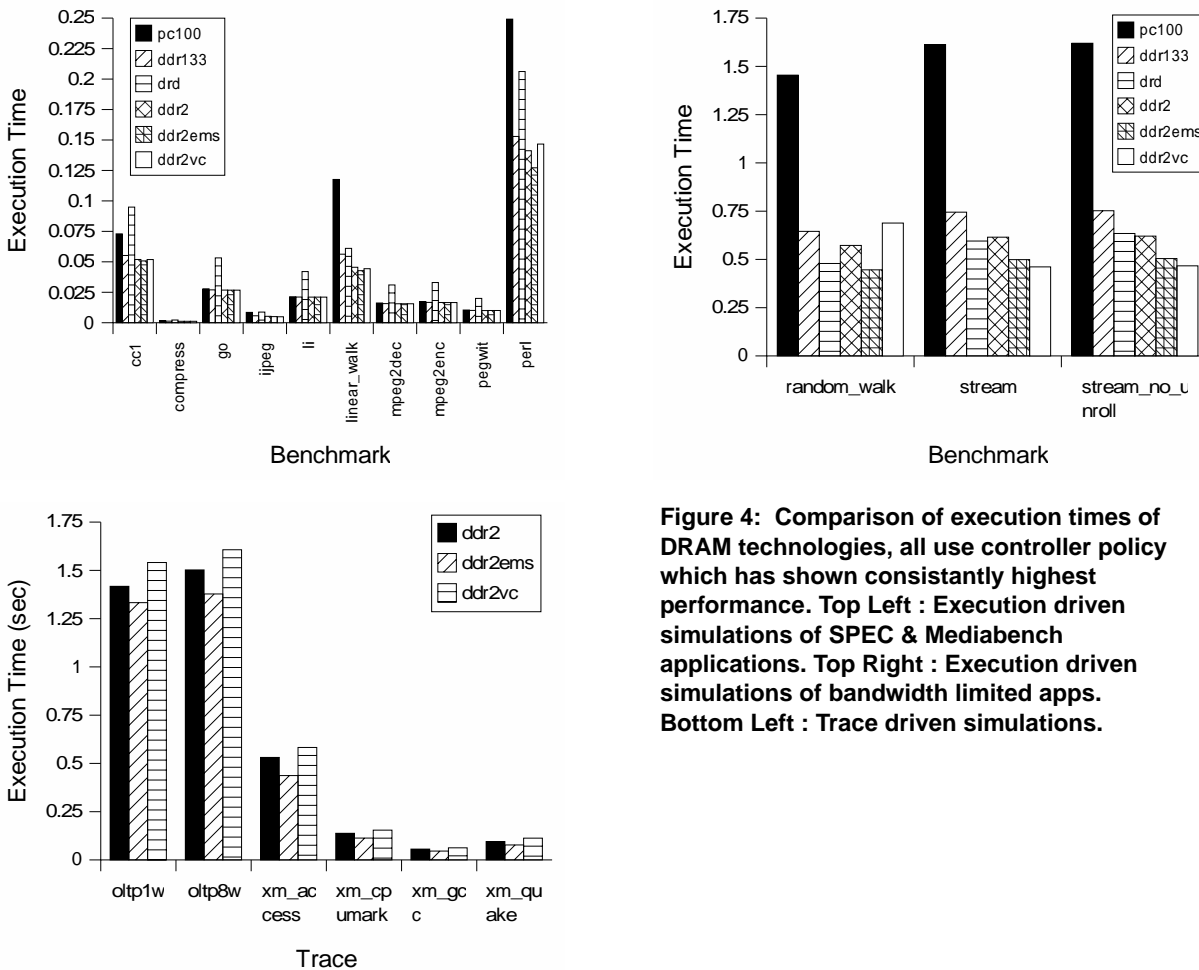


Figure 4: Comparison of execution times of DRAM technologies, all use controller policy which has shown consistently highest performance. Top Left : Execution driven simulations of SPEC & Mediabench applications. Top Right : Execution driven simulations of bandwidth limited apps. Bottom Left : Trace driven simulations.

The results for execution driven simulations clearly show that DDR2 as an interface specification allows for higher performance than any of the other architectures. This is not extremely surprising given that DDR2 has the highest potential bandwidth of any of the interface architectures. Comparing the three DDR2 architectures, again we have a situation where either DDR2EMS or DDR2VC may provide the highest level of performance, based upon the benchmark or trace. Specifically, DDR2VC provides the best performance for both variations of Stream as well as jpeg, presumably because of the multiple access streams of these applications and the increased associativity of the cache structures in the VC architecture. For the remainder of the applications the DDR2EMS architecture provides the highest level of performance.

7 Conclusions

Address remapping is more significant than controller policy for most architectures, which motivates increased investigation into address remapping. The approach used in these studies was simple, based on the requirement that it be applied to a wide range of architectures, and system configurations. Improving this address remapping technique, or implementing similar techniques such as access re-ordering or coalescing to reduce the number of adjacent accesses mapping to unique rows in the same bank may provide additional performance improvements.

There are minimally two categories into which applications may be classified. Applications that completely saturate the memory bus (stream, random_walk and the trace-driven simulations) are performance dependent upon the bandwidth and utilization of the memory bus. These applications can be classified as bandwidth-tracking. The reason that the trace-driven simulations are uniformly bandwidth-tracking is due to the fact that these traces either do not contain access timing information, or the timestamps are disregarded due to the significant difference between the memory system upon which they were collected, and the memory system being simulated. Applications that do not contain the access parallelism to saturate the memory bus (gcc, perl, go, li) are more performance dependent upon the latency of accesses. These applications can be classified as latency-tracking.

The memory wall is not insurmountable, and researchers are pursuing methods to overcome the latency associated with DRAM transactions from both sides of the “wall”. On the processor side, latency tolerating approaches are an active area of research. On the DRAM side, industry is developing, revising, and changing their DRAM technologies on a month-to-month basis. Because DRAM is a commodity part, for which price is extremely important, the JEDEC committee is attempting to standardize upon an architecture to limit market fragmentation and thus preserve the commodity feature. Even within the interface and architecture defined as DDR2 there is room for low-latency variants. For the same reason that it is unwise to fragment the DRAM market, it would also be unwise to fragment this low-latency or cache-enhanced design space. Through ongoing research with the JEDEC committee, it is our hope that a commercially viable low-latency architecture with the highest level of performance, lowest additional system costs, and minimal fragmentation of the design space may be specified.

Acknowledgements

This work has been partially funded by DARPA grant DABT63-97-C-0047. In addition, many individuals have assisted us in this work, first and foremost the members of the Low Latency DRAM Working Group. Among the members we would like to give special thanks to: Bill Gervasi of Transmeta who provided traces, Mike Peters of Enhanced Memory Systems who provided architectural and timing specifications for the EMS enhanced devices, and to Akira Yabu and Jeffery Lee of NEC who provided specifications for the VC enhanced devices.

REFERENCES

- D. Burger and T. M. Austin. 1997. “The SimpleScalar tool set, version 2.0.” Technical Report CS-1342, University of Wisconsin-Madison.
- D. Burger, J. R. Goodman, and A. Kagi. 1996. “Memory bandwidth limitations of future microprocessors.” In *Proc. 23rd Annual International Symposium on Computer Architecture (ISCA '96)*, pages 78–89, Philadelphia PA.

- C-3D. 2000. "C3D Data Storage Technology." Constellation 3D, <http://www.c-3d.net/tech.htm>
- J. Carter, W. Hsieh, L. Stoller, M. Swanson, L. Zhang, and et al. 1999. "Impulse: Building a smarter memory controller." In *Proc. Fifth International Symposium on High Performance Computer Architecture (HPCA '99)*, pages 70–79, January 1999, Orlando FL.
- V. Cuppu, B. Jacob, B. Davis, and T. Mudge. 1999. "A performance comparison of contemporary DRAM architectures." In *Proc. 26th Annual International Symposium on Computer Architecture (ISCA '99)*, pages 222–233, Atlanta GA.
- B. Dilpert. 2000. "The Slammin' Jammin' DRAM Scramble." *Electronic Design News*, Jan 20, 2000, pages 68-80. <http://www.ednmag.com/ednmag/reg/2000/01202000/pdfs/02cs.pdf>
- EMS. 2000. "64Mbit - Enhanced SDRAM". Enhanced Memory Systems, http://www.edram.com/Library/datasheets/SM2603,2604pb_r1.8.pdf.
- L. Gwennap. 1998. "Alpha 21364 to Ease Memory Bottleneck." *Microprocessor Report*, Oct. 26, 1998, pages 12-15.
- IBM. 1998. "64 Mb Synchronous DRAM". International Business Machines, <http://www.chips.ibm.com/products/memory/03K2149/03K2149.pdf>
- IBM. 1999. "128Mb Direct RDRAM". International Business Machines, <http://www.chips.ibm.com/products/memory/19L3262/19L3262.pdf>
- Chunho Lee, M. Potkonjak, W.H. Mangione-Smith. 1997. "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems." In *Proceedings of the 30th Annual International Symposium on Microarchitecture (Micro '97)*.
- W. Lin. 1999. "Prefetching at the Interface between Level-two Cache and Direct Rambus." Research Report, May 4, 1999. University of Michigan.
- J. McCalpin. 2000. "STREAM: Sustainable Memory Bandwidth in High Performance Computers.", <http://www.cs.virginia.edu/stream/>
- B. McComas. 1999. "DDR vs. Rambus: A Hands-on Performance Comparison", November 1999, InQuest Market Research, <http://www.inqst.com/ddrvrmb.htm>.
- S. McKee, A. Aluwihare, B. Clark, R. Klenke, T. Landon, C. Oliver, M. Salinas, A. Szymkowiak, K. Wright, W. Wulf, and J. Aylor. 1996. "Design and evaluation of dynamic access ordering hardware." In *Proc. International Conference on Supercomputing*, Philadelphia PA.
- S. A. McKee and W. A. Wulf. 1995. "Access ordering and memory-conscious cache utilization." In *Proc. International Symposium on High Performance Computer Architecture (HPCA '95)*, pages 253–262, Raleigh NC.
- SPEC. 1995. "SPEC CPU95 Benchmarks." Standard Performance Evaluation Corporation. <http://www.spec.org/osg/cpu95/>.
- NEC. 1999. "128M-BIT Virtual Channel SDRAM". NEC Electronics Inc, [http://www.necel.com/home.nsf/ViewAttachments/M14412EJ3V0DS00/\\$file/M14412EJ3V0DS00.pdf](http://www.necel.com/home.nsf/ViewAttachments/M14412EJ3V0DS00/$file/M14412EJ3V0DS00.pdf).
- S. Przybylski. 1996. *New DRAM Technologies: A Comprehensive Analysis of the New Architectures*. MicroDesign Resources, Sebastopol CA.
- Rambus. 1999. "Direct RMC.d1 Data Sheet" Rambus, <http://www.rambus.com/developer/downloads/RMC.d1.0036.00.8.pdf>.
- J. Robertson. 2000. "Advanced DRAM alliance works on specs for release later this year", *Semiconductor Business News*, Mar. 07, 2000. <http://www.semibiznews.com/story/OEG20000307S0006>.
- W. Wulf and S. McKee. 1995 "Hitting the Memory Wall: Implications of the Obvious." *ACM Computer Architecture News*. Vol 23, No. 1. March 1995.