A Single Chip Multiprocessor Integrated with DRAM

Tadaaki Yamauchi[1], Lance Hammond[2], and Kunle Olukotun[2]

[1]ULSI Laboratory
Mitsubishi Electric Corporation
Itami, 664 Japan

[2]Computer Systems Laboratory
Stanford University
Stanford, CA 94305

Abstract

We evaluate the performance of a single chip multiprocessor integrated with DRAM. We compare the performance of this architecture with that of a more conventional chip which only has on-chip SRAM. The DRAM-based architecture with four processors performs an average of 52% faster than the SRAM-based architecture on floating point applications with large working sets. This is performance difference is significantly better than a uniprocessor DRAM-based architecture, which only performs an average of only 4% faster than an SRAM-based architecture on the same applications. These results demonstrate that a multiprocessor takes better advantage of the large bandwidth provided by the on-chip DRAM than a uniprocessor.

1 Introduction

Recently, microprocessor chips with integrated DRAM have been developed [3] to close the speed gap between processors and memory [1,2]. In these chips, the DRAM and the processor are connected using a wide internal data bus. The high speed data transfer over the internal data bus improves memory latency, because the load capacitance of the internal data bus is small compared to that of an external data bus. The majority of proposals for single chip processor-memory integration use a very simple processor and fill the remaining area on the chip with DRAM. In this paper we show that when DRAM is combined with a more powerful microprocessor architecture such as a multiprocessor, the on-chip DRAM bandwidth is more efficiently used and the overall performance of the chip is dramatically improved.

In a system where all the main memory is on the processor chip it is possible that some large applications may run out of main memory. These applications will require access to off-chip memory. We investigate the performance of an OS-based page-fault mechanism that provides this support.

The reminder of this paper is organized as follows. In Section 2, we discuss the architectural model that we use to evaluate system performance. In Section 3, we discuss the simulation methodology. In Section 4, a single chip multiprocessor integrated with DRAM main memory is evaluated. The influence of page faults on integrated DRAM performance is estimated in Section 5. Finally we conclude in Section 6.

2 Architectural Models

2.1 A Single Chip MP with on-chip DRAM

A single chip multiprocessor is a promising candidate for future high performance microprocessor design [4]. When present DRAM processes are used to fabricate high performance processors, the processor's logic gates typically suffer speed and area penalties. However, some DRAM manufacturers are developing merged process technologies which can provide high density for the embedded DRAM without degrading the logic performance. In this paper we will assume that the single chip multiprocessor integrated with high density DRAM can run at a clock frequency of 500MHz.

Fig. 1 shows the block diagram of a single chip multiprocessor integrated with DRAM. Four 2-way superscalar processors are interconnected with a 256 bit wide read/replace data bus whose bit width is identical to the cache line size. As a result, each line replacement between the cache and DRAM occupies the bus for only one CPU cycle. Since the four processors share the read/replace bus, arbitration for this resource requires an extra cycle. Each of the four processors has a 16KB SRAM instruction cache and a 16KB SRAM data cache, both accessible in a single 2 ns clock cycle. Since each cache can only be accessed by a single processor or its single load/store unit, no additional arbitration overhead is required. A write back with write-miss allocation policy is implemented to reduce the traffic on the read/replace data bus. Coherency among the individual data caches is maintained using an update coherence protocol. The data written by each processor is broadcasted to all other data caches through the 64 bit update bus. Each data cache has two I/O ports to minimize the interference caused by the update operation.

The capacity of the embedded DRAM main memory is assumed to be 256Mb, organized as 16 independent banks. Having multiple banks reduces the number of bank conflicts. Requests to access the main memory are queued in the memory buffer until the proper memory bank is free to accept the access. To manage the multi-bank DRAM properly and generate necessary DRAM control signals such as RAS and CAS, an extra cycle of latency is incurred as accesses are pulled out of the memory buffer. Unlike the read/replace bus, the memory bus cycle is twice as long as a CPU clock cycle. As a result, the peak memory bandwidth without bank conflicts is 8 GB/s. All data that is read from the DRAM is queued into the memory return buffer. The cache line reaches the processor after a cycle spent successfully arbitrating for the read/replace bus. As a result, 5 cycles are required to control the multi-bank DRAM and its supporting resources, as follows: two cycles for the two bus arbitrations, two cycles buffering requests going to or from the processors, and one cycle for DRAM control. The structure of the memory buffers allows memory accesses to be pipelined. In this architecture, the number of memory requests is primarily limited by the number of embedded DRAM banks, and secondarily by contention for the read/replace bus.

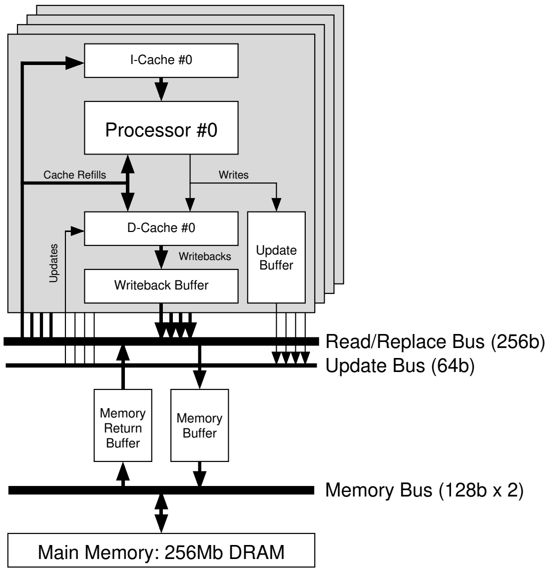


Figure 1 Block diagram of a single chip multiprocessor integrated with 256Mb DRAM on the same die.

The raw access and cycle times of the embedded DRAM are assumed to be 30ns and 60ns, respectively. These values are quite reasonable for a 256Mb DRAM using a 0.25µm technology [5].One possible problem is that the multi-bank architecture will increase the area of the embedded DRAM. For example, a 16 bank embedded DRAM is 1.4 times larger and the 32 bank one is 1.8 times larger than a 4 bank one [6]. This area penalty will significantly reduce the density of the embedded DRAM which can be economically integrated with a processor. To achieve high system performance with a minimal area penalty the hierachical multi-bank DRAM architecture, proposed in [6], could be used. In this architecture the memory banks are divided into the main banks and sub-banks. The main bank is identical to a conventional independent bank. The sub-banks are implemented within main banks. All of the sub-banks within a main bank share I/O circuitry, decoders, and sense amplifiers. In this architecture, while full bank conflicts can occur between neighboring sub-banks, more distant sub-banks within each main bank may overlap their row cycles.

Fig. 2 shows the floor plan for a single chip multiprocessor integrated with a 256Mb DRAM. For the sake of clarity, the embedded DRAM consists of four main banks. In order to reduce the critical path of each memory access, the multiprocessor is located between the upper and lower halves of the embedded DRAM. With a 0.25µm merged process technology, which allows high density memory arrays and logic gates, we estimate that the chip size will be approximately 24 millimeters on a side. The size of the 4 x 2-way multiprocessor section of the chip is extrapolated from the current MIPS R10000 processor [7].

Another possible problem with this integrated DRAM is that 32MBytes of main memory may not be enough in high-end computer systems. Since a fixed amount of memory is integrated on the die, it is difficult to adjust the amount of memory in different systems. In this case, off-chip DRAM may be added to the system to form another memory hierarchy level below the on-chip main memory. Data movement between the two can be controlled by a software modification of the existing virtual memory system. All of the applications which we are using to evaluate system performance in this paper fit within a single 32MByte main memory, so virtual memory operation is not required. However, in section 5, we will evaluate the effects of delays caused by swapping pages into the on-chip DRAM during page faults simply by reducing the on-chip main memory size.

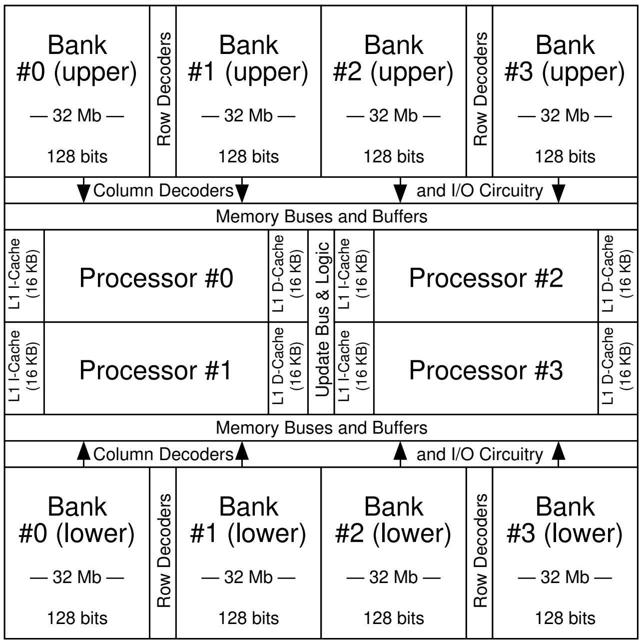


Figure 2 The floor plan for a single chip multiprocessor integrated with a 256Mb DRAM consisting of four main banks. Note that this is simpler than the 16-bank architecture used for measurements in the paper. More banks can be added by making each bank narrower. When the hierachical multi-bank architecture in [6] is used, each main bank of two 32Mb memory arrays consists of up to 32 sub-banks so the entire four-bank DRAM array performs comparably to a conventional 32 bank architecture.

2.2 The SRAM-Based Alternative

In a more conventional system, an SRAM L2 cache, which has approximately the same area as the embedded 32MB DRAM in the configuration in section 2.1, is used. There are various types of SRAM memory cells, with different densities and performance characteristics: 4 transistor cells with high resistance polysilicon passive pullups, 6 transistor cells with TFT PMOS active pullups, and 6 transistor cells with conventional PMOS active pullups. Six transistor SRAM cells, which are commonly used for cache memory on processor dies, are about 21 times larger than DRAM memory cells consisting of only a single transistor and capacitor, in comparable technologies [8]. In commodity SRAMs, 4 transistor cells with high resistance polysilicon passive pullups or 6 transistor cells with TFT PMOS active pullups are widely used to achieve higher cell densities. For example, in the 0.35µm-based 16Mbit SRAM generation, 6 transistor cells with TFT PMOS are commonly used, and are only 5 times less dense than commodity DRAM memory cells manufactured in a comparable process [9,10]. Even with the handicap of driving off-chip data lines, these high-density SRAMs can achieve access times of less than 10ns.

Additionally, the area of the sense amplifiers embedded in each DRAM array is about 25% of the total area of the memory cells when the 128 cells are connected with each bit line [9]. SRAMs typically do not need nearly as much sense amplifier circuitry as DRAMs. As a result of the large amount of other logic included in each memory array, the area difference between SRAM and DRAM is not as extreme as one might expect simply by comparing cell sizes.

The higher density 64Mb SRAMs using more aggressive process technologies have not appeared yet. However, in this paper we estimate that the density of any practical SRAM cache, after considering the area penalty of the L2 tag circuitry, is 8 to 16 times less than that of an equal-area DRAM main memory. The raw access time of the embedded L2 cache is assumed to be 8ns, or 4 CPU cycles. This small access time is quite reasonable even with high density SRAMs based on the data presented in [10]. In addition, an arbitration cycle is incurred to acquire the read/replace bus. The whole access period is arbitrated at once because of the fixed, short access time of the embedded SRAM. Therefore, the extra cycles required to allow for the memory buffers and the second arbitration in the embedded DRAM are not necessary.

The high speed off-chip DRAM main memory in this configuration is connected to the processor through a 64-bit memory bus operated at 250MHz. This data rate could be achieved by double data rate synchronous DRAM (DDR-SDRAM) which major DRAM manufacturers have recently developed. The off-chip DRAM is handled using buffering mechanisms similar to those used to control the embedded DRAM. The main memory latency is assumed to be 100ns —50ns of communication overhead and 50ns of DRAM access time. The off-chip DRAM is configured as four independent banks.

We summarize both of the system configurations in Table 1:

	Integration with DRAM	Integration with SRAM
number of CPUs	4	4
CPU configuration	2-way superscalar	2-way superscalar
CPU frequency	500MHz	500MHz
L1 configuration	Independent cache for each CPU	Independent cache for each CPU
L1 capacity	16KB I + 16KB D	16KB I + 16KB D
L1 associativity	4-way	4-way
L1 write policy	write-back with write-miss allocation	write-back with write-miss allocation
L1 line size	32B	32B
L1 access time	1 CPU cycle	1 CPU cycle
L2 configuration	-	Common for every CPU
L2capacity	-	2MB or 4MB unified
L2 associativity	-	2-way
L2 write policy	-	write-back with write-miss allocation
L2 line size	-	64B
Control overhead	-	1 CPU cycle (Arbitration)
L2 access time	-	4 CPU cycles
memory configuration	on-chip 32MB DRAM	off-chip DRAM
number of banks	16	4
memory bus width	256b	64b
bus frequency	250MHz	250MHz
Control overhead	5 CPU cycles (Arbitrations, buffer delay, and DRAM control)	5 CPU cycles
DRAM access time	30ns	100ns
Row cycle time	60ns	100ns

Table 1 Simulated models of a single chip superscalar MP.

3 Methodology

3.1 Simulation Environment

We execute applications in the SimOS simulation environment [11]. With SimOS, the processors, the memory hierarchy, and cache coherence issues are modelled in detail. Special attention is paid to modelling contention between processors due to shared resources such as the central data bus. SimOS emulates a multiprocessor running the full MIPS-II instruction set interacting with a realistic set of I/O components, allowing the full Silicon Graphics IRIX 5.3 operating system to be executed under our benchmarks. SimOS supports three kinds of CPU simulators, which allow tradeoffs to be made between simulation speed and accuracy. In this paper, the slowest, most detailed CPU simulator is used. This model supports multiple instruction issue in each processor, along with full emulation of dynamic scheduling, speculative execution, and non-blocking memory references. The cache and memory system components shown in Fig. 1 are completely event-driven and interface to the SimOS processor models.

3.2 Applications

To evaluate the system performance, five realistic applications are used. Table 2 shows the five applications: one SPEC95 integer benchmark (compress), one SPEC92 integer benchmark (eqntott), and three SPEC95 floating point benchmarks (swim, tomcatv, and applu). The applications are parallelized in different ways to run on a multiprocessor. The compress benchmark cannot be effectively parallelized, so only one of the four processors is used. Eqntott is parallelized manually by modifying a single bit vector comparison routine that is responsible for 90% of the execution time of the application [12]. It is characterized by a small working set. The SPEC95 floating point benchmarks are automatically parallelized by the SUIF compiler system [13] across loop iterations at a reasonably coarse level.

	Floating Point Applications
swim	shallow water model with 1K x 1K grid
tomcatv	mesh-generation with Thompson solver
applu	solver for parabolic/elliptic partial differential equations
	Integer Applications
compress	compresses and uncompresses files in memory
eqntott	translates logic equations into truth tables

Table 2 Applications.

4 Performance Comparison

In this section the performance effect of the embedded DRAM is evaluated in our three system configurations using the five applications described in section 3.2. In the configurations with an on-chip SRAM L2 cache, a 2MB L2 cache is used to measure pessimistic results while a 4MB L2 is used to measure optimistic results, because an on-chip SRAM with area equal to an embedded 32MB DRAM will probably have an area somewhere between these sizes. Additionally, 2-way superscalar *uni*processor systems are evaluated with all three memory systems to measure the benefit provided by the single-chip multiprocessor.

The average miss ratio among the four 16KB L1 data caches and the local miss ratio of 2MB and 4MB L2 unified caches in 4 x 2-way multiprocessor systems are shown in Fig. 3. The L2 local miss rates also consider I-cache accesses to the L2 cache, but these have only a minor effect in our benchmarks. In the unparallelized compress benchmark, only data for the single active L1 cache is shown. The average miss ratios for the floating point benchmarks in the four L1 data caches are 7.6% in swim, 4.6% in tomcatv, and 2.7% in applu. These three applications all have large working set sizes, greater than 16MB. As a result, in the pessimistic SRAM system with only 2MB of L2 cache, 28.8%, 29.8%, and 30.2% local miss ratios are observed in swim, tomcatv, and applu, respectively. The larger 4MB SRAM caches improve matters only slightly, since the large working sets are still not captured within the cache. These high miss ratios force many references to access the slow, off-chip main memory. On the other hand, the embedded DRAM does not need to wait for off-chip references, even with these relatively large benchmarks. As a result, the embedded DRAM achieves a much better average memory access time.

The average memory latency including both caches and main memory is depicted in Fig. 4. As Fig. 3 demonstrated, the on-chip SRAM cache based architecture exhibits large local miss rates in

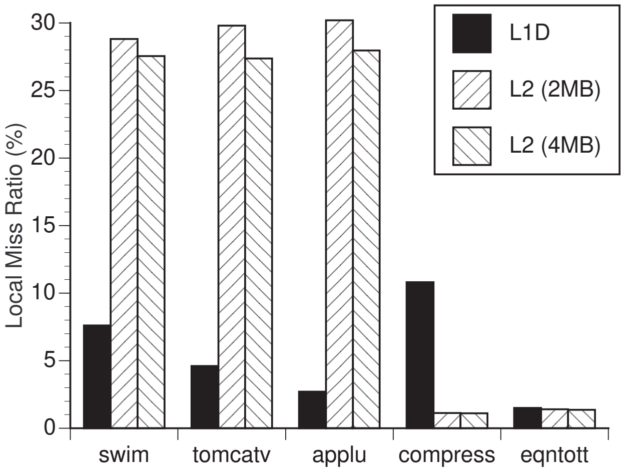
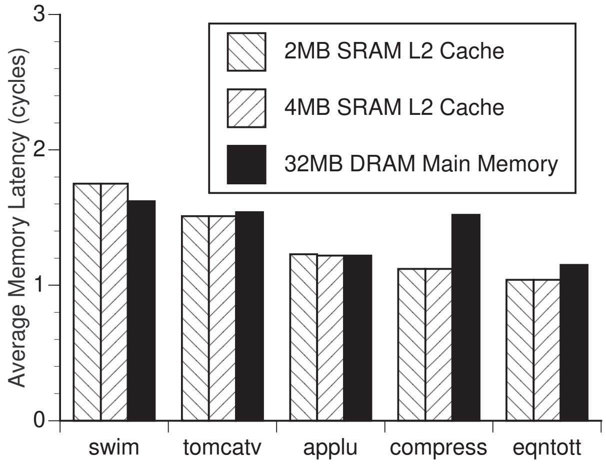


Figure 3 Average miss ratio among four 16KB L1 data caches and the local miss ratio of 2MB and 4MB L2 unified caches in 4 x 2-way multiprocessor systems. For the unparallelized compress benchmark, only the single active L1 data cache is considered.

(a) a 2-way uniprocessor system:



(b) a 4x2-way multiprocessor system:

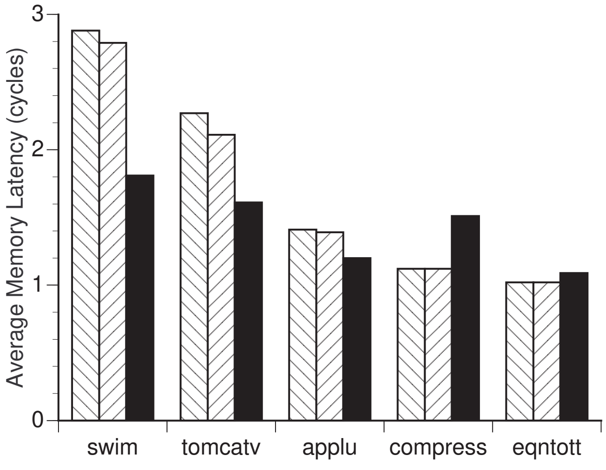


Figure 4 Average Memory Latencies, in processor cycles.

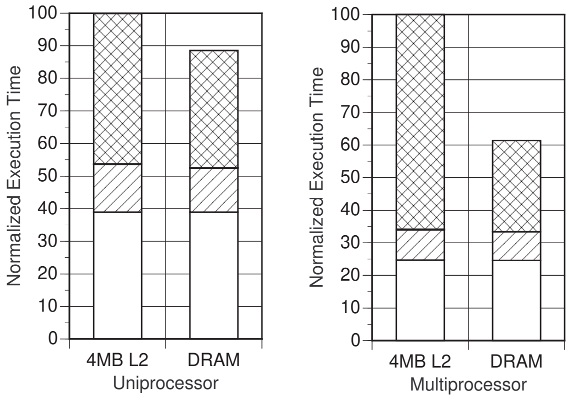
the L2 cache in the three floating point benchmarks. As a result, these applications require significant numbers of memory accesses to off-chip DRAM in order to handle the L2 misses. In the uniprocessor system this causes a performance loss, but not a dramatic one. Since the single processor has all of the limited off-chip bandwidth and all four off-chip DRAM banks dedicated to it, bandwidth limitations never affect performance dramatically. However, the four processors in the multiprocessor system, working together, are able to demand much more bandwidth from the memory system. The integrated DRAM's many banks can easily process multiple references in flight from all four processors at once and still return data in a timely manner over the high-bandwidth on-chip bus. On the other hand, the limited bandwidth provided by the off-chip DRAM bus and the reduced number of banks that can be economically implemented with discrete chips cannot handle bursts of cache misses from multiple processors without becoming a bottleneck. This causes a significant increase in the average latency seen by the individual processors on each memory access in the multiprocessor system. The key insight is that the on-chip DRAM bandwidth is more *efficiently* used in a single chip multiprocessor system than in a uniprocessor system. The uniprocessor system essentially has too much memory and too little computation resources.

Next we will consider the simulated results of the integer benchmarks. The working set size of the two integer benchmarks, compress and eqntott, is small. Even with the pessimistic 2MB SRAM L2 cache, most data references hit in the cache — the local miss ratio is less than 2% in both benchmarks. As a result of this, and since SRAM access times are shorter than DRAM ones, these applications have lower average memory latencies and better performance with an SRAM L2 cache. The average miss ratio among the four L1 data caches is only 1.5% in eqntott, better than any of the FP applications, due to the relatively small size of the data set. On the other hand, the miss ratio of the single active L1 data cache in the unparallelized compress is high, 10.8%. Because most of these L1 misses are satisfied in the on-chip SRAM L2 or DRAM, the performance differences are essentially just the difference between the SRAM and the DRAM access time, multiplied by the L1 miss ratios. As a result, since compress exhibits a higher L1 miss rate, the SRAM-based configurations demonstrated a much larger latency advantage over the DRAM configuration with that application.

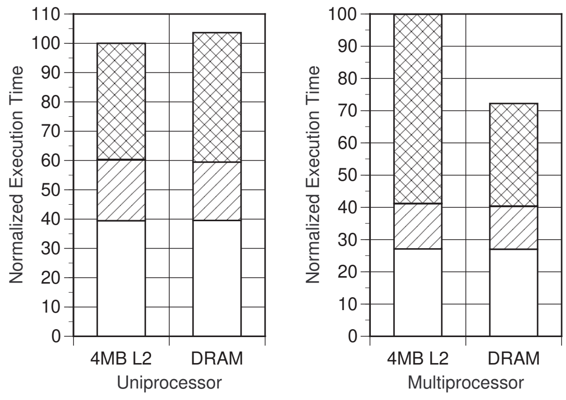
We present the simulated results for each application, normalized to the speed of the 4MB SRAM L2 based architecture in Figs. 5. Each graph breaks down the execution time of uniprocessor and multiprocessor systems. For the unparallelized compress benchmark, only the uniprocessor case is shown. This analysis allows us to focus on the percentage of time spent waiting for delays caused by the memory systems. The time spent waiting for a spin lock or for barrier synchronization is included in the CPU execution time. The speed of the load-linked and store-conditional memory operations used to implement these synchronization primitives affects the amount of the time the processors spend synchronizing. These synchronization operations make it difficult to directly compare the multiprocessor and uniprocessor performance numbers using Fig. 5. Later we will discuss the effective IPC, counting only useful instructions completed per cycle, in order to look at the numbers together.

Figs. 5(a,b,c) show the results of the swim, tomcatv, and applu benchmarks, respectively. In the uniprocessor system, the ratio of the non-memory time (CPU execution time and pipeline stall time) is about 55%, 60%, and 75% in swim, tomcatv, and applu, respectively. The main memory integration causes a small performance enhancement in the swim benchmark. However, tomcatv and applu show little performance difference between the two architectural models since they spend less time waiting for memory than swim . With a high-performance single chip multiprocessor instead, main memory integration has a more significant effect. The data cache stall time increases dramatically in the on-chip L2 cache architecture as a result of large contention for the off-chip main memory among the four processors. The off-chip bus, which is 4 times narrower than the on-chip memory bus, increases the data cache stall time tremendously since memory accesses in flight simultaneously

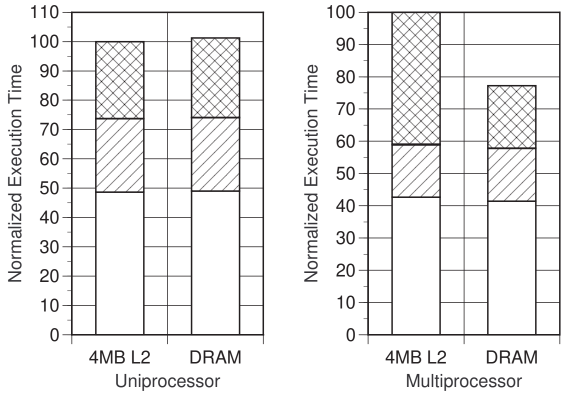
(a) swim:



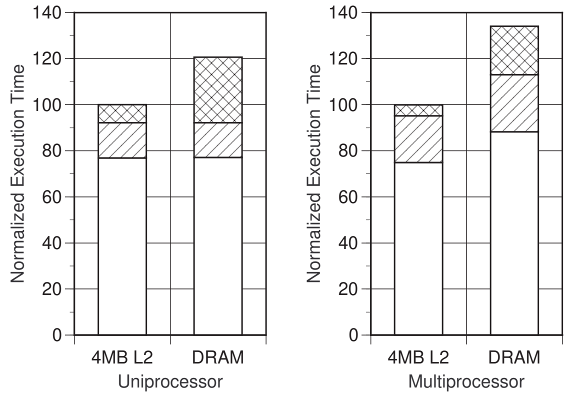
(b) tomcatv:



(c) applu:



(d) eqntott:



(e) compress:

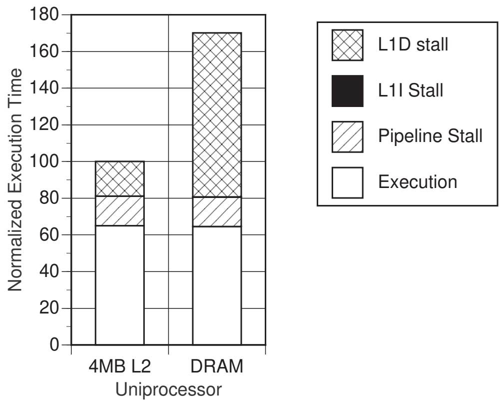
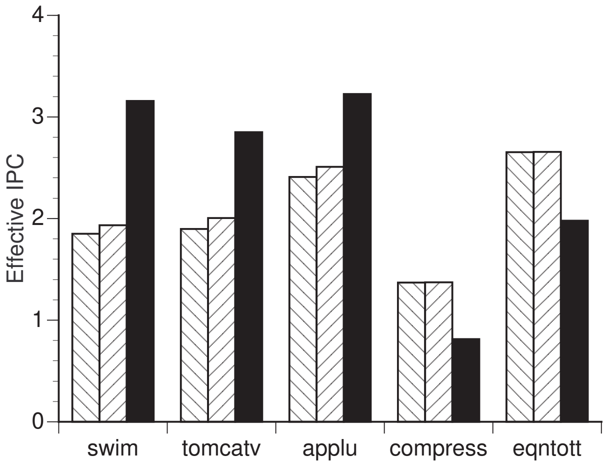


Figure 5 Normalized execution times of the 4MB on-chip L2 cache system and the on-chip DRAM architecture. Results for both the 2-way uniprocessor system and the 4x2-way multiprocessor system are presented for the first four benchmarks. For the unparallelizable compress benchmark, only the results from the 2-way uniprocessor system are shown.

are frequently queued in the memory buffers. On the other hand, the integrated main memory maintains almost the same ratio of data cache stall time to the whole execution time even when the number of processors increases from one to four. The high-bandwidth embedded main memory can easily manage many memory requests in flight at once.

Fig. 5(d) shows the results of the eqntott benchmark, which is characterized by a small working set and a high communication to computation ratio [12]. The high communication to computation ratio is due to the fact that the eqntott is parallelized at the innermost vector comparison loop. Every time this loop is executed, the four processors synchronize at a barrier and the master processor transmits copies of the last three quarter-vectors being compared to the slave processors so they can perform their portion of the comparison. In this paper an update coherence protocol is used in the multiprocessor systems, taking advantage of the high bandwidth of the on-chip update bus. In addition, two data cache ports are implemented to minimize any interference caused by the update operation. This architecture can maintain high hit ratios in the data caches even when sharing frequently happens in benchmarks such as the eqntott. In the on-chip SRAM L2 cache system, the memory system accounts for less than 10% of the total execution time. As Fig.2 shows, its small working set can fit into the L1 data caches—it is a SPEC92 benchmark, and this result demonstrates the largest limitation of that benchmark suite clearly. The on-chip L2 cache and off-chip main memory accesses are less frequent than in any other benchmark. As a result, eqntott does not obtain the benefits of main mem-

(a) Effective IPC of the 4x2-way multiprocessor:



(b) IPC of the 2-way uniprocessor:

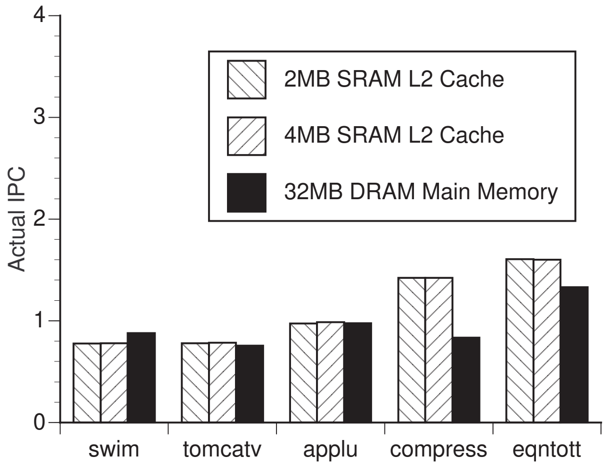


Figure 6 IPCs of evaluated systems while executing several applications. The dramatic performance increase provided by on-chip DRAM with FP applications is obvious in the 4x2-way case. The effective multiprocessor IPCs are calculated without considering instructions from synchronization overhead.

ory integration. However the difference in eqntott execution times between the two memory architectures is smaller than that for compress, because the low number of references that miss in the L1 cache do not stress the lower portions of the memory system at all.

Fig. 5(e) shows the results of the compress benchmark. Only results of the uniprocessor system case are shown because this benchmark cannot be effectively parallelized. Unlike the eqntott benchmark, the data cache miss ratio of the compress benchmark is quite high. Its working set is a few hundred KB, so it doesn't fit into the L1 caches, but fits nicely into the L2s. In this application, most of the data cache misses are satisfied in the on-chip SRAM L2 cache. As a result, the data cache stall time in the embedded main memory architecture is larger than that in the on-chip SRAM L2 cache architecture, just because the embedded DRAM access time is larger than the on-chip L2 cache access time.

Next we will discuss the overall system performance. Fig. 6 (a) shows 4 x 2-way multiprocessor system performance in all configurations using the effective IPC, counting only useful instructions completed per cycle. IPCs of the 2-way uniprocessor systems are shown in Fig. 6 (b).

A single chip multiprocessor integrated with DRAM main memory enhances the three floating point applications which have large working sets. The swim application, which has the lowest global hit ratio in the L2 cache, reduces the memory latency most significantly. As a result, the embedded DRAM system obtains the largest performance enhancement over the 2MB SRAM L2 cache configurations — 70%. In this case, the embedded DRAM system is still 63% better than even the optimistic 4MB SRAM L2 cache system.

In uniprocessor systems the performance enhancement obtained from embedded DRAM main memory is relatively small, at most 13% over the SRAM L2 cache systems while running swim. On the other two floating point benchmarks, tomcatv and applu, performance differences between the memory system configurations are negligible. These results show that the effects of DRAM main memory integration are most significant in fairly complex high-performance processors, such as single chip multiprocessors or very wide-issue superscalar processors, that can have many memory requests in progress at once. Such processors are able to take advantage of the large bandwidth provided by the wide, on-chip bus to main memory while being able to hide the longer latency of DRAM accesses due to reasonably large amounts of ILP.

The DRAM-based configuration performs 40% and 25% worse than the SRAM configurations on the integer benchmarks compress and eqntott, respectively. As we discussed previously, the working sets for these applications are smaller than even the pessimistic 2MB L2 cache, so few accesses need to go off-chip in either configuration. As a result, the raw access speed of the L2 SRAM cache allows that configuration to outperform the embedded DRAM configuration easily.

5 Page Fault Effects with On-Chip DRAM

All of the applications which we are using to evaluate system performance in this paper fit within a single 32MByte main memory. Therefore virtual memory operation was not considered in the previous discussion—the on-chip main memory was always sufficient. However, page faults may occur when the working set size of applications doesn't fit into the embedded main memory. To examine on-chip DRAM performance in the presence of page faults, we simply reduce the 32MB size of the embedded DRAM to 16MB. When 16KB pages are used, the probability of page faults on any main memory reference is 0.035%, 0.076%, and, 0.17% in the swim, tomcatv, and applu benchmarks, respectively. Judging from the nonzero page fault probability, the working set size of the three floating point benchmarks is clearly larger than the 16MB embedded main memory size. To reduce the overhead of page swapping, off-chip DRAM should be added to the system to form the second level of the main memory hierarchy, as discussed in Section 2.1. When the off-chip DRAM consists of high speed DRAM like that used in the SRAM based architecture, the maximum bandwidth is up to 2GB/s. The page transfer time of a 16KB page is roughly estimated to be 8μs. In this paper, to estimate the performance penalty of page faults, the page transfer time is simply added to the main memory latency and every processor is stalled during the page swap. This is the pessimistic case, since the other three processors in a multiprocessor may be able to make forward progress even while a page swap is in progress, possibly reducing the penalty down to as little as a quarter of the value we estimated. With longer page swap times and a multiprogramming environment, it may also be possible to have another process run on the faulting processor while the page fault is being handled.

Fig. 7 shows the effective IPC of 4 x 2way multiprocessor systems integrated with 16MB DRAM main memory for the FP programs. The two integer benchmarks completely fit into a 16MB embedded main memory, and are thus not considered here. The page penalty of 8μs is considered in Fig. 7. To clarify the effects of main memory integration, 2MB and 1MB on-chip SRAM L2 cache architectures—half the sizes of the previously examined SRAM caches—are also examined to provide a point of comparison. The off-chip

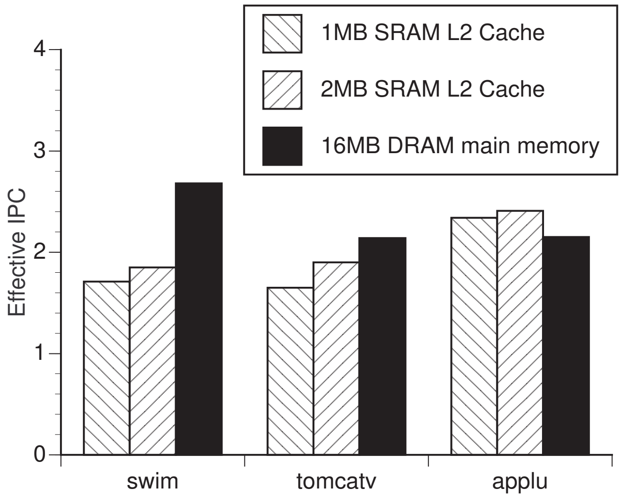


Figure 7 Effective IPC of 4 x 2-way multiprocessor systems, including page fault effects due to insufficient on-chip main memory.

main memory in both cases is assumed to be large enough not to incur further page faults to lower levels of the memory hierarchy.

The page faults degrade the effective IPC among our three floating point benchmarks. The IPC degradation caused by page faults is relatively small in the swim benchmark. The embedded DRAM system still obtains a large performance enhancement over the 1MB SRAM L2 cache configuration — 57%. In this case, the embedded DRAM system is still 44% better than even the optimistic 2MB SRAM L2 cache system. This is less than the original improvement of 70% and 63%, but still significant. However, the performance enhancement due to main memory integration is much smaller in tomcatv, and applu actually slows down.

Fig. 8 shows the relative performance of the 16MB integrated DRAM system and the optimistic 2MB on-chip SRAM L2 cache one with varying page fault penalty times. When the page fault penalty increases to just over 30µs, due to off-chip bandwidths of only 500MB/s to a lower level of DRAM, performance improvements disappear even in swim.

The rapid performance reduction across the range of small penalties shown makes it clear that it is very important to reduce the page fault penalty in embedded main memory systems that may require off-chip memory resources. High speed off-chip DRAM should compose another main memory hierarchy level below the on-chip DRAM. Also, the operating system should be optimized to control the main memory hierarchy with minimal page swap penalties.

6 Conclusion

Microprocessors integrated with DRAM on the same die have the potential to enhance system performance by reducing the average memory latency experienced by many applications, in addition to reducing energy consumption. The effect of integrating DRAM main memory with a processor depends on the performance of the processor and the applications being executed on the system.

In this paper we evaluate the potential of on-chip DRAM main memory by analyzing three system configurations running several applications. The performance of applications which have large working sets is significantly enhanced, especially when a high performance CPU is integrated with DRAM. Our results show that when a single chip multiprocessor that can supply many memory requests to the DRAM simultaneously is integrated with DRAM main memory, the performance is on average 52% better on FP applications than if the same multiprocessor is integrated with a comparable on-chip SRAM L2 cache system. However, when a single 2-way uniprocessor is integrated instead, the average performance gain is only 4% with these applications. Also, applications with small working sets slowed down in the DRAM configurations by an average of 33%, because the large size of the DRAM was not helpful in reducing the memory access time.

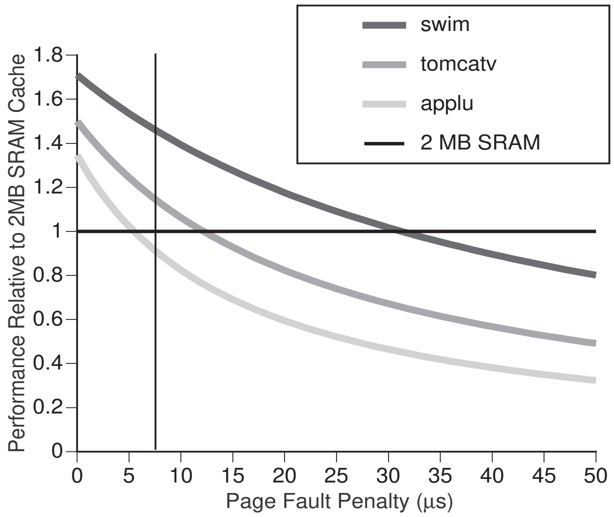


Figure 8 Relative performance of the 16MB on-chip DRAM memory system to the on-chip 2MB SRAM L2 cache system, while varying the page fault penalties incurred during off-chip accesses from the DRAM. This graph shows penalties associated with off-chip DRAM support—penalties associated with disk accesses would be far off the right end of this graph. The vertical line indicates where the 8µs values used in Fig. 7 were sampled.

When the integrated main memory isn't large enough for some applications, page faults frequently occur to bring in memory from off-chip resources. We also evaluate the effect of page faults by reducing the embedded main memory size. In this paper, we simply stall processors while handling a page fault. This method is very pessimistic, but it is clear that page faults can remarkably degrade system performance if there is not enough on-chip DRAM. To reduce the page fault penalty, off-chip high speed DRAM should be added to the system to form another memory hierarchy level below the on-chip DRAM main memory. Data movement between the two can be controlled by a software modification of the existing virtual memory system.

These results show that a small uniprocessor cannot effectively take advantage of the large bandwidth provided by the embedded DRAM architecture. However, when DRAM is integrated as main memory with single chip multiprocessors, system performance can be dramatically improved in applications that have large working sets yet still fit into the integrated main memory.

References

[1] A. Saulsbury, F. Pong, and A. Nowatzyk, "Missing the Memory Wall: The Case for Processor/Memory Integration," *23th International Symp. on Computer Architecture*, pp. 90-101, Philadelphia, PA 1996.

[2] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "Intelligent RAM (IRAM): Chips that Remember and Compute," *Digest of Technical Papers, 1997 IEEE International Solid State Circuits Conference*, pp. 224-225, San Francisco, CA 1997.

[3] T. Shimizu, J. Korematu, M. Satou, H. Kondo, S. Iwata, K. Sawai, N. Okumura, K. Ishimi, Y. Nakamoto, M. Kumanoya, K. Dosaka, A. Yamazaki, Y. Ajioka, H. Tsubota, Y. Nunomura, T. Urabe, J. Hinata, and K. Saitoh, "A multimedia 32b RISC microprocessor with 16Mb DRAM," *Digest of Technical Papers, 1996 IEEE International Solid State Circuits Conference*, pp. 216-217, San Francisco, CA 1996.

[4] K. Olukotun, K. Chang, L. Hammond, B. Nayfeh, and K. Wilson, "The Case for a Single-Chip Multiprocessor," *Proceedings of the 7th International Symp. Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, pp. 2-11, Cambridge, MA 1996.

[5] M. Asakura, T. Ohishi, M. Tsukude, S. Tomishima, H. Hidaka, K. Arimoto, K. Fujishima, T. Eimori, Y. Ohno, T. Nishimura, M. Yasunaga, T. Kondoh, S. Satoh, T. Yoshihara, and K. Demizu, "A 34ns 256Mb DRAM with Boosted Sense-Ground Scheme," *Digest of Technical Papers, 1994 IEEE International Solid State Circuits Conference*, pp. 140-141, San Francisco, CA 1994.

[6] T. Yamauchi, L. Hammond, and K. Olukotun, "The Hierachical Multi-Bank DRAM: A High-Performance Architecture for Memory Integrated with Processors," to be presented at *17th Conference on Advanced Research in VLSI*, Ann Arbor, MI 1997.

[7] K. Yeager, "The MIPS R10000 Superscalar Microprocessor," *IEEE Micro*, vol. 16, No. 2, pp. 28-40, April 1996.

[8] R. Fromm, S. Perissakis, N. Cardwell, B. McGaughy, C. Kozyrakis, D. Patterson, T. Anderson, and K. Yelick, "The Energy Efficiency of IRAM Architectures," *24th International Symp. on Computer Architecture*, Denver, CO 1997.

[9] T. Yamauchi et al., "Fully Self-timing Data-Bus Architecture for 64-Mb DRAMs," IEICE TRANS. ELECTRON., VOL. E78-C, NO.7, pp. 885-865, 1995.

[10] K. Seno et al., "A 9ns 16Mb CMOS SRAM with Offset Reduced Current Sense Amplifier," *Digest of Technical Papers, 1993 IEEE International Solid State Circuits Conference*, pp. 248-249, San Francisco, CA 1994.

[11] M. Rosenblum, S. Herrod, E. Witchel, and A. Gupta., "The SimOS approach," *IEEE Parallel and Distributed Technology*, vol.4, no. 3, 1995.

[12] B. Nayfeh, L. Hammond, and K. Olukotun, "Evaluation of Design Alternatives for a Multiprocessor Microprocessor," *Proceedings of the 23th International Symposium on Computer Architecture*, pp. 66-67, Philadelphia, PA 1996.

[13] R. Wilson, R. French, C. Wilson, S. Amarasinghe, J. Anderson, S. Tjiang, S.-W. Liao, C.-W. Tseng, M. Hall, M. Lam, and J. Hennessy, "The SUIF Compiler System: A Parallelizing and Optimizing Research Compiler," Stanford University Technical Report No. CSL-TR-94-620, May 1994.