A Single Chip Multiprocessor Integrated with High Density DRAM

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

[1]ULSI Laboratory, Mitsubishi Electric Corporation

[2]Computer Systems Laboratory, Stanford University

Abstract

A microprocessor integrated with DRAM on the same die has the potential to improve system performance by reducing memory latency and improving memory bandwidth. In this paper we evaluate the performance of a single chip multiprocessor integrated with DRAM when the DRAM is organized as on-chip main memory and as on-chip cache. We compare the performance of this architecture with that of a more conventional chip which only has SRAM-based on-chip cache. The DRAM-based architecture with four processors outperforms the SRAM-based architecture on floating point applications which are effectively parallelized and have large working sets. This performance difference is significantly better than that possible in a uniprocessor DRAM-based architecture, which performs only slightly faster than an SRAM-based architecture on the same applications. In addition, on multiprogrammed workloads, in which independent processes are assigned to every processor in a single chip multiprocessor, the large bandwidth of on-chip DRAM can handle the inter-access contention better. These results demonstrate that a multiprocessor takes better advantage of the large bandwidth provided by the on-chip DRAM than a uniprocessor.

Keywords

DRAM, on-chip DRAM, embedded DRAM, L2 caches, on-chip L2 caches, SRAM caches, multiprocessors, multiprocessor-on-a-chip

1 Introduction

Recently, microprocessor chips with integrated DRAM have been developed [1] to close the speed gap between processors and memory [2,3]. In these chips, the DRAM and the processor are connected using a wide internal data bus. 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 then require access to off-chip memory. We investigate the performance of an OS-based page-fault mechanism that provides this support. Alternatively, the on-chip DRAM may be treated as a very large on-chip cache instead of main memory. Off-chip main memory is required in this case, but caches which consist of DRAM can have much larger capacities than the more conventional SRAM cache designs with equal die area.

The reminder of this paper is organized as follows. In Section 2, we discuss the architectural model used 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. The alternative configuration with on-chip DRAM designed as a very large on-chip cache is evaluated in Section 6. Finally we conclude in Section 7.

2 Architectural Models

2.1 A Single Chip MP with on-chip DRAM

High speed DRAMs, such as synchronous DRAMs, have been developed to achieve high data transfer rates for serial accesses. This speed-up offers a large benefit for bandwidth-intensive applications. However, improvements in memory latency have not kept up with the almost exponential increase in processor speed, so system performance is often limited by the memory access time [2].

Recently, very simple microprocessors integrated with DRAM main memory have been developed for the embedded systems [1]. In this architecture, the DRAM and the processor are connected using a wide internal data bus on a single die. The high speed data transfer through the internal data bus can improve the memory latency, and therefore the performance, because the load capacitance of the internal data bus is negligibly small compared to that of an external data bus. We propose adding a more complex processor to an integrated processor and DRAM chip. Since a single chip multiprocessor has been demonstrated as a promising candidate for future high performance microprocessor design [4], we use one in the simulations in this paper.

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 a single 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 broadcast 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 updates.

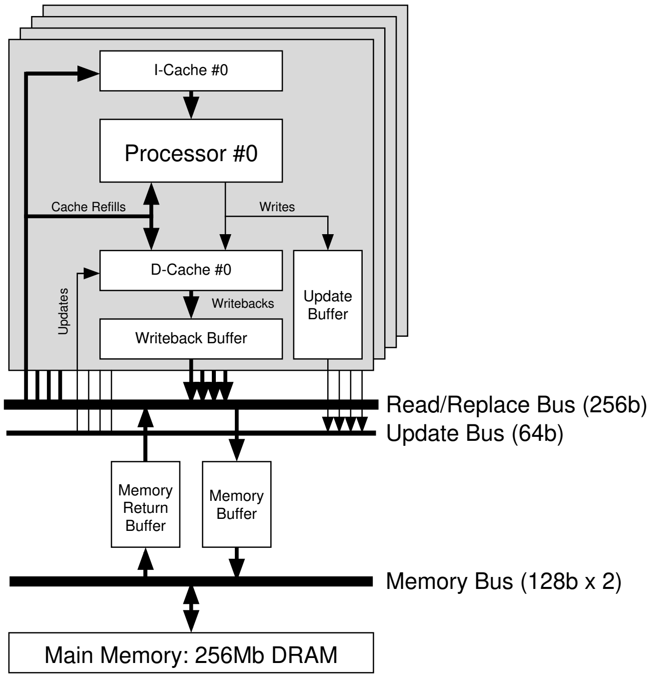


Figure 1 Block diagram of a single chip multiprocessor integrated with 256Mb DRAM on the same die.

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, which can have a significant effect on the single-chip multiprocessor system [5]. 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. Therefore, 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 for 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.

The access and cycle times of the embedded DRAM are assumed to be 30ns and 60ns, respectively. These values are quite reasonable for a mass-produced 256Mb DRAM using a 0.25μm technology. These times are also reasonable compared with modern discrete DRAMs, which typically have 50-60ns access times, including the delay of the pin I/O drivers.

A 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 normally 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

Another way to handle the problem discussed above is to utilize the on-chip DRAM as a very large on-chip L2 cache instead of on-chip main memory. The large hit ratio possible with an on-chip DRAM L2 cache can considerably reduce the number of off-chip memory accesses even though the entire working set of the application cannot fit into the on-chip DRAM.

On multiprogrammed workloads, the different processors in an on-chip multiprocessor may all be running processes with separate, disjoint data sets. When the working sets of all active processes don't fit into the on-chip DRAM, the multiple processes will compete for space in the on-chip DRAM, causing many page faults. A caching on-chip DRAM may provide superior performance in this case.

2.2 The SRAM-Based Alternative

An SRAM L2 cache, which has approximately the same area as the embedded 32MB DRAM in the configuration in section 2.1, is used in a more conventional system. 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 [6]. 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 [7,8]. 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 128 cells are connected to each bit line [7]. SRAMs typically do not need nearly as much sense amplifier circuitry as DRAMs. As a result of the large amount of support 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.

Higher density 64Mb SRAMs using more aggressive process technologies have not appeared yet. However, in this paper we assume that the density of any practical SRAM cache 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 [8]. In addition, an arbitration cycle is incurred to acquire the read/replace bus which is shared among four processors. Only one memory reference is accepted through the shared resources, such as the read/replace bus and L2 cache tag circuitry, in a single clock cycle. However, these resources are fully pipelined in order to handle the numerous accesses from all four processors. The entire L2 access cycles may be properly arbitrated at once because of the fixed, short access time of the embedded SRAM. Therefore, the extra cycles required in the embedded DRAM case to allow for the memory buffers and the second bus arbitration 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 Fully pipelined access
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 [9]. 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, six realistic applications are used. Table 2 shows the six applications: one SPEC95 integer benchmark (compress), one SPEC92 integer benchmark (eqntott), three SPEC95 floating point benchmarks (swim, tomcatv, and applu) and a multiprogramming workload (VCS). 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 [10]. It is characterized by its small working set. The SPEC95 floating point benchmarks are automatically parallelized by the SUIF compiler system [11] across loop iterations at a reasonably coarse level. Our final workload, a multiprogrammed engineering workload, consists of several memory intensive applications. Every process is a copy of the commercial verilog simulator VCS, each simulating large VLSI circuit. VCS compiles the simulated circuit into code, and the resulting large code segments, whose size is approximately 4MB, cause a high user instruction stall time. Each process has a working set smaller than 8MB, including data and code. Thus, four VCS processes benchmark running together fit into the 32MB on-chip DRAM main memory. Once the operating system assigns the four processes to four processors, each processor continues to execute the same process without context switching because there are no extra processes competing for processor time and the entire working sets fit into the 32MB on-chip DRAM. Only one process is executed on the uniprocessor system in order to evaluate both systems in the absence of context switches. The six process VCS benchmark, whose working set doesn't fit into the 32MB on-chip main memory, is also evaluated using the caching on-chip DRAM. Here SimOS simulates the behavior of a realistic operating system, and, the multiprogrammed workload is scheduled by UNIX priority scheduling with affinity.

	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
	Multiprogrammed Workload
VCS	VCS compiled Verilog simulation of a large VLSI circuit. 1, 4, or, 6 VCS processes which are based on independent binary code and data are executed

Table 2 Applications.

4 Performance Comparison

In this section the performance effect of the embedded DRAM is evaluated for our three system configurations using the six 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 and 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 uniprocessor 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 and instruction caches and the local miss ratio of 2MB and 4MB L2 unified caches in 4 x 2-way multiprocessor systems are shown in Fig. 2. In the unparallelized compress benchmark, only data for the single active L1 cache is shown. The average miss ratios of the L1 instruction caches for all of our benchmarks except VCS are negligibly small, 0.01%. The average L1 data cache miss ratios for the floating point benchmarks 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, 27.3%, 29.1%, and 29.6% 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 doesn't 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. 3. As Fig. 2 demonstrated, the on-chip SRAM cache based architecture exhibits large local miss rates in

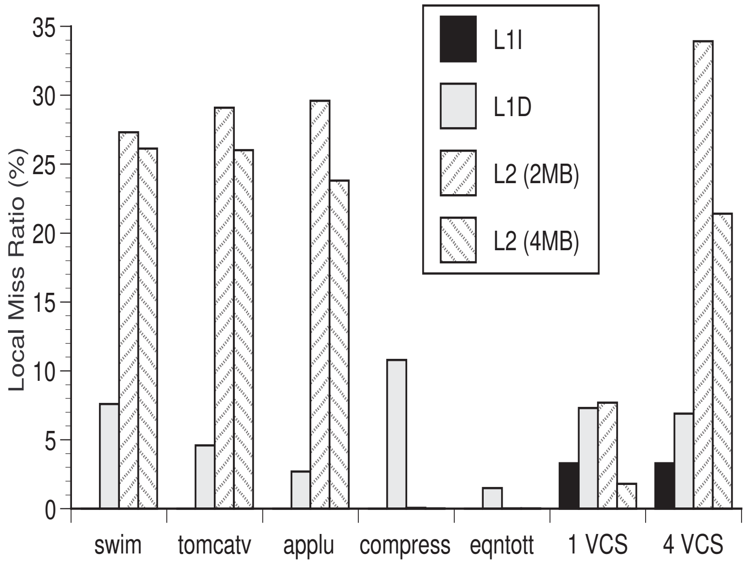
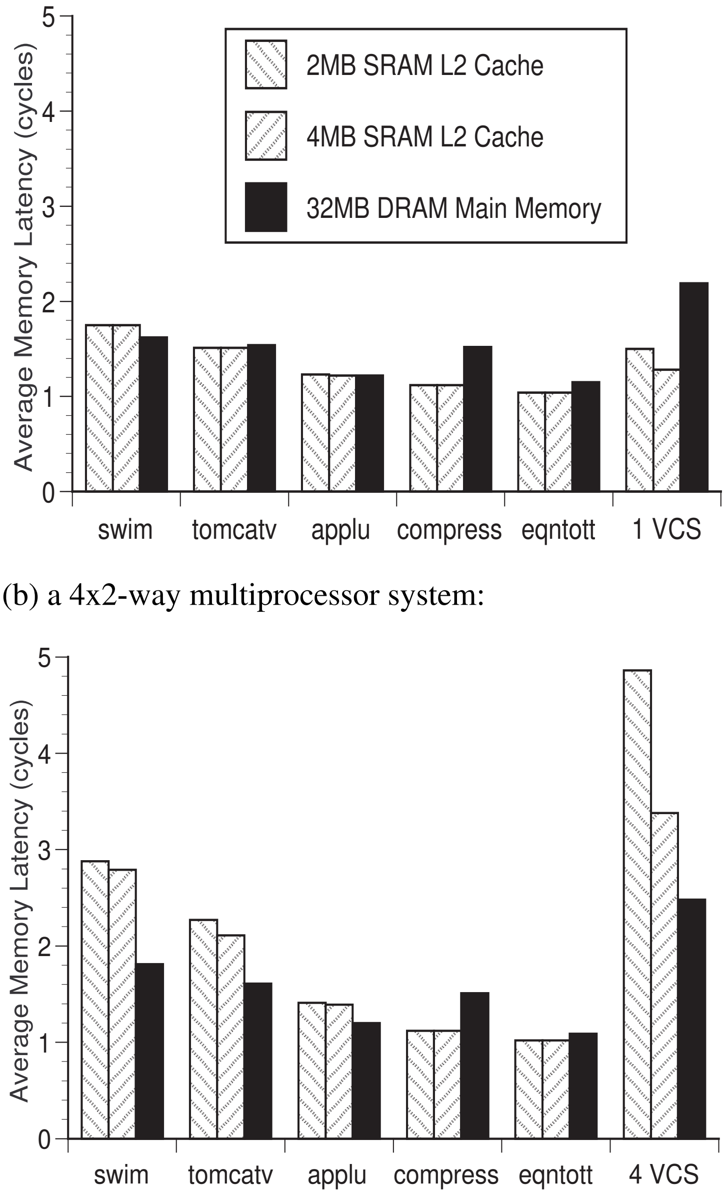


Figure 2 Average miss ratio among four 16KB L1 data caches and the local miss ratio of 2MB and 4MB L2 unified caches in 4 x 2-way multiprocessor systems. For the unparallelized compress benchmark and one process VCS workload in the uniprocessor system, only the single active L1 data cache is considered.

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 ac-

(a) a 2-way uniprocessor system:



(b) a 4x2-way multiprocessor system:

Figure 3 Average Memory Latencies, in processor cycles.

cess 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 computational 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 1% 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.

Third, we consider the simulated results of the multiprogrammed benchmarks. As discussed before, the 4 x 2-way multiprocessor system executes four VCS processes whose whole working sets fit into 32MB on-chip DRAM. Only one process is executed in the uniprocessor system in order to avoid context switches, keeping the comparison to a 4 x 2-way multiprocessor fair. The large instruction memory requirements, which are about 4MB for each process, cause the relatively high 3.3% L1 instruction cache miss ratio. The L1 data cache miss ratio is 6.9%, on average, in the multiprocessor system. Each of the four processors has independent L1 instruction and data caches, and executes its process without context switching. As a result, the average miss ratios of L1 caches in the multiprocessor system are almost identical to those in the uniprocessor one. On the other hand, the shared L2 caches perform much worse in the multiprocessor system than in the uniprocessor one, with 21.4% versus 7.7% local miss rates, respectively, with the larger 4MB cache. This is due to the larger aggregate working set of the four processes running on the multiprocessor.

In the uniprocessor system, both of the SRAM-based configurations, which achieve low local miss rates with only one running process, demonstrate much larger latency advantages over the DRAM configuration. As discussed above, the on-chip SRAM based architecture in the multiprocessor system exhibits large local L2 miss rates. As a result, significant numbers of memory accesses to off-chip DRAM are required, and therefore the SRAM-based configurations in the multiprocessor system exhibit larger average memory latency than in the uniprocessor system. Moreover, since all of L1 misses are satisfied in the 32MB on-chip DRAM, the DRAM configuration in the multiprocessor system shows only a small increase in the memory latency compared to the uniprocessor system. Thus multiprogrammed workload efficiently uses the on-chip DRAM bandwidth in the multiprocessor system.

We present the simulated results for each application, normalized to the speed of the 4MB SRAM L2 based architecture in Fig. 4. 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. 4. Later we will discuss the effective IPC, counting only useful instructions completed per cycle, in order to look at the numbers together.

Figs. 4(a,b,c) show the results of the swim, tomcatv, and applu benchmarks, respectively. In the uniprocessor system, the percentage 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 mem-

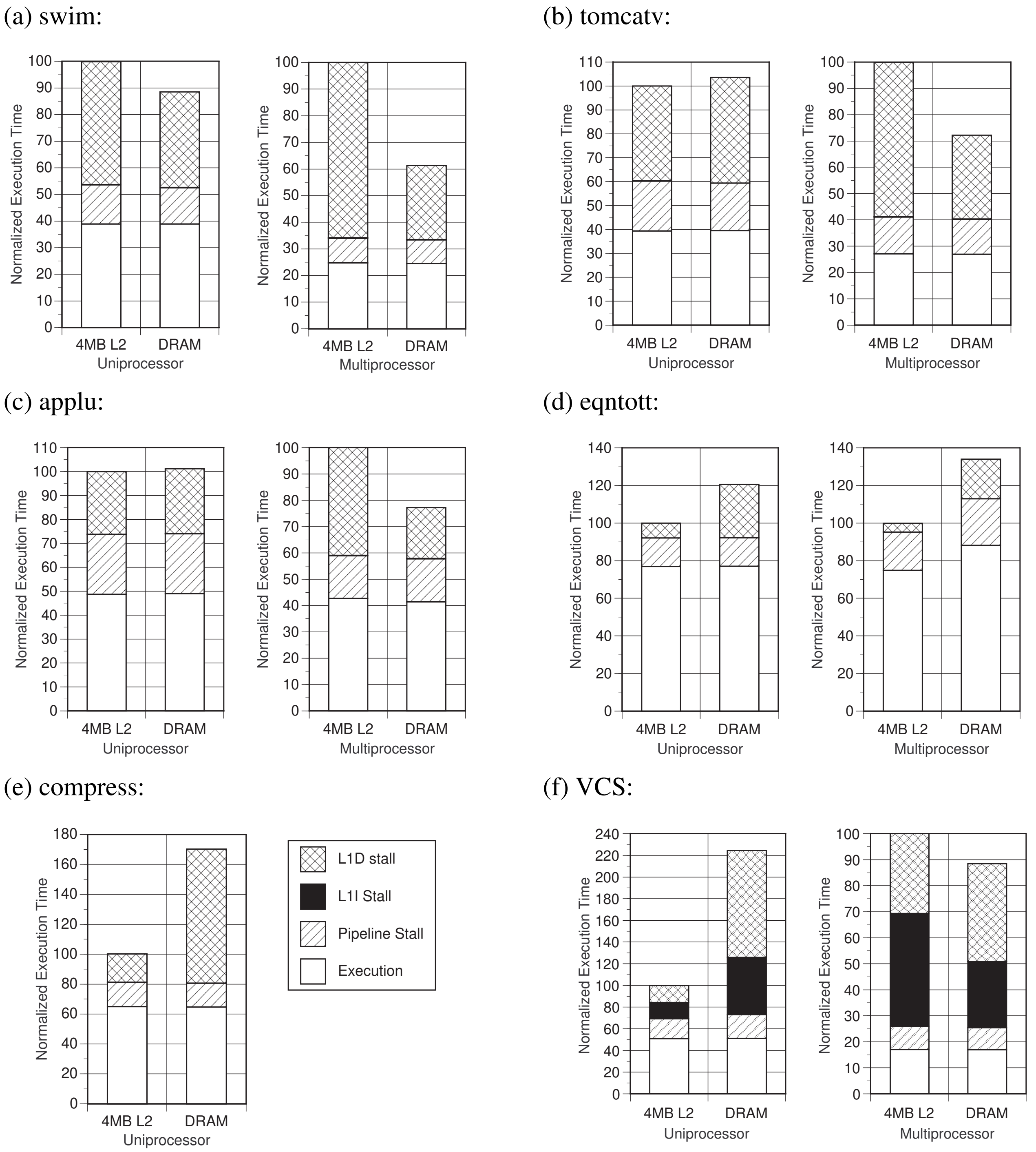
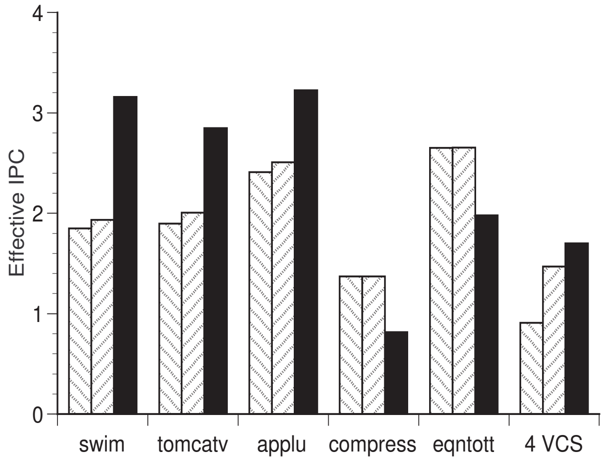


Figure 4 Normalized execution times of the 4MB on-chip L2 cache system and the on-chip DRAM architecture. Results for both the 2-way uniprocessor system and the 4x2-way multiprocessor system are presented for the first four benchmarks. For the unparallelizable compress benchmark, only the results from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

ory 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 however, 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 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 entire 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. 4(d) shows the results of the eqntott benchmark, which is characterized by a small working set and a high communication to computation ratio [10]. 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. This update coherence protocol used between L1 caches can maintain high hit ratios in the data caches even when sharing is frequent, as in 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 benefits from main memory integration. The difference

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



(b) IPC of the 2-way uniprocessor:

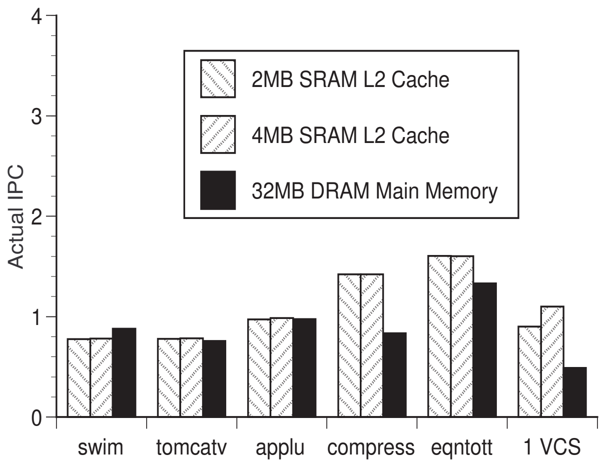


Figure 5 IPCs of evaluated systems while executing several applications. The dramatic performance increase provided by on-chip DRAM with FP applications is obvious in the 4x2-way case. The effective multiprocessor IPCs are calculated without considering instructions from synchronization overhead.

in 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. 4(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 it does fit 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.

Fig. 4(f) shows the results of the multiprogrammed benchmark. Compared to other benchmarks, the large instruction segment results in significant L1 instruction cache stalls. In the uniprocessor system with the 4MB on-chip SRAM L2 cache, the ratio of the non-memory time is large, 69%. As with the two integer benchmarks, the 4MB on-chip SRAM L2 cache can handle most of the L1 cache misses. As a result, the on-chip DRAM based architecture is slower because of its inherently slower access time.

In the multiprocessor system, the situation is quite different. The 4MB on-chip SRAM L2 cache cannot hold the aggregate working set of four VCS processes at once. In addition, accesses from the four processors incur queuing delays to the off-chip main memory. As a result, the memory system accounts for the significant ratio of the total execution time in the on-chip SRAM L2 cache system, 74%. On the other hand, the on-chip DRAM architecture maintains almost the same ratio of memory stall time between the multiprocessor and uniprocessor systems. This is because the large bandwidth of the on-chip DRAM can easily handle the multiple memory accesses in flight.

Next we will discuss the overall system performance. Fig. 5 (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. 5 (b).

A single chip multiprocessor integrated with DRAM main memory performs best on the three floating point applications. 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 their ability to exploit 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.

As with the two integer benchmarks, the DRAM-based configuration performs 55% worse than the SRAM configurations in the uniprocessor system on the one process VCS benchmark. When four processes are assigned to four processors, however, the multiprocessor takes advantage of the large bandwidth of the on-chip DRAM. As a result, the DRAM-based configuration performs 12% and 83% better than the optimistic 4MB SRAM cache system and the pessimistic 2MB one, respectively. Unlike our floating point benchmarks, the large aggregate working sets exhibit a distinct difference in the performance between 4MB and 2MB SRAM cache systems.

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%, 0.17%, and, 2.5% in the swim, tomcatv, applu, and, 4 process VCS multiprogrammed benchmarks, respectively. As the nonzero page fault probability attests, the working set size of the three floating point and the multiprogrammed benchmarks is clearly larger than 16MB. 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 in our three floating point benchmarks. 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. While the processors frequently synchronize each other during the execution of three floating point benchmarks, the processes in the 4 process VCS multiprogrammed benchmark are completely independent. As a result the other three processors can continue to execute their own processes during the page swapping on any one processor. Therefore, the behavior will be much closer to the optimistic case, and we simulate a page transfer time by stalling all processors for a quarter of the page transfer time. With longer page swap times, it may also be possible to have another process run on the faulting processor while the page fault is being handled.

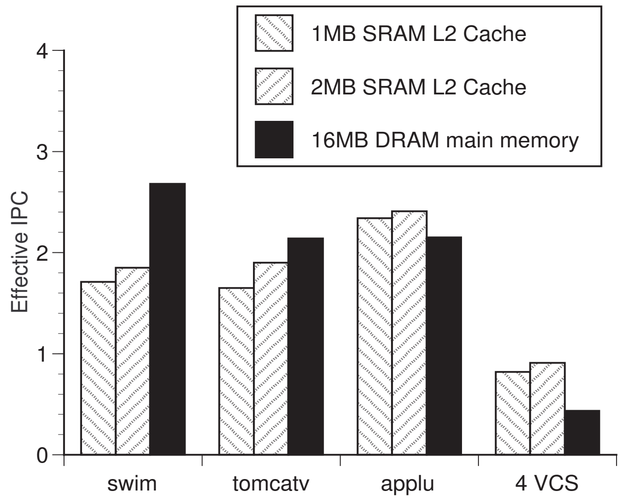


Figure 6 Effective IPC of 4 x 2-way multiprocessor systems, including page fault effects due to insufficient on-chip main memory.

Fig. 6 shows the effective IPC of 4 x 2way multiprocessor systems integrated with 16MB of DRAM main memory for the FP and multiprogrammed applications. 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. 6. 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 main memory in both cases is assumed to be large enough to not incur further page faults to lower levels of the memory hierarchy.

The page faults degrade the effective IPC among our three floating point benchmarks and the multiprogrammed benchmark. 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 63%, but still significant. However, the performance enhancement due to main memory integration is much smaller in tomcatv, and applu actually slows down. The multiprogrammed benchmark is even worse, with the DRAM-based configuration performing approximately 50% worse than the SRAM configurations.

Fig. 7 shows the relative performance of the 16MB integrated DRAM system and the optimistic 2MB on-chip SRAM L2 cache system with varying page fault penalty times on our three floating point and multiprogrammed benchmarks. 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, most likely requiring off-chip silicon memory resources. High speed off-chip DRAM should compose another main memory hierarchy level below the on-chip DRAM, and the operating system should be optimized to control the main memory hierarchy with minimal page swap penalties. Alternatively, the limited on-chip DRAM capacity could be controlled as very large on-chip cache instead of as on-chip main memory. In Section 6, we evaluate the performance of such a system.

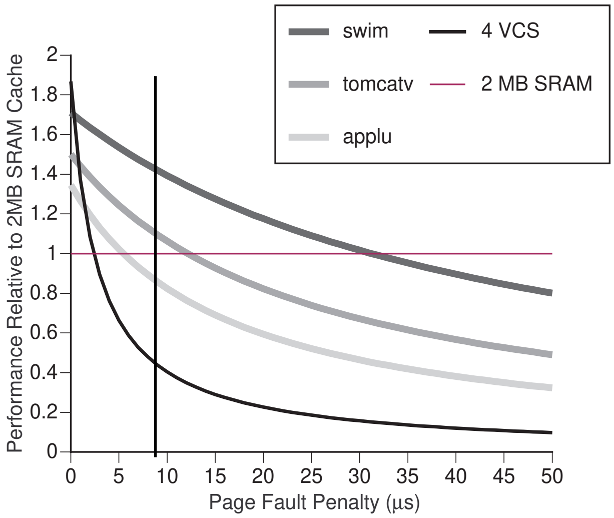


Figure 7 Relative performance of the 16MB on-chip DRAM memory system to the on-chip 2MB SRAM L2 cache system, while varying the page fault penalties incurred during off-chip accesses from the DRAM. This graph shows penalties associated with off-chip DRAM support—penalties associated with disk accesses would be far off the right end of this graph. The vertical line indicates where the 8µs values used in Fig. 6 were sampled.

6 Performance of DRAM Cache

As the previous section shows, page faults to off-chip memory when the on-chip main memory isn't sufficient can significantly impact the performance of a system with on-chip DRAM. A different solution is to design the on-chip DRAM as the large L2 cache instead of as main memory. Unlike the OS-based paging approach, hardware resources such as cache tag circuitry must be added in order to handle the on-chip DRAM as a cache. However, the miss penalty of the on-chip DRAM can be minimized, because processors can execute successive instructions as soon as the critical word is returned to them, as they can with the on-chip SRAM L2 cache.

For this section, we use our three floating point and the multiprogrammed benchmarks. In the previous discussion, at most four processes are considered in the multiprogrammed benchmark in order to fit the whole working set into the 32MB of on-chip main memory and to avoid context switches. In this section, we also consider a scenario with six processes running on four processors in order to evaluate the performance including OS-controlled process switching. Under SimOS, any workload runs on top of the Silicon Graphics IRIX5.3 operating system which will initiate process switches about every 30ms. The stored data in the on-chip cache is gradually replaced after each process switch, since the on-chip caches are not large enough to contain the working sets of all the processes. Since less replacement occurs with larger caches, the size of the on-chip cache greatly affects the performance of this benchmark.

We evaluate the 16MB on-chip DRAM cache using the three floating point benchmarks and four- and six-process multiprogrammed benchmarks. The on-chip DRAM cache is direct mapped, with a line size of 64Bytes. With the on-chip DRAM cache system, the off-chip DRAM system used by the SRAM L2 cache system shown in Table 1 is also used to handle cache misses.

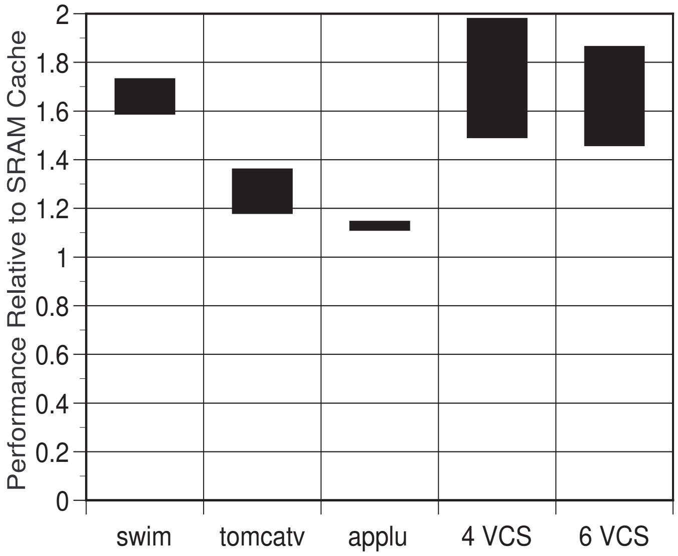
Fig. 8(a) shows the performance range of the on-chip DRAM cache system relative to the on-chip SRAM cache performance in the multiprocessor system. For the 16MB on-chip DRAM cache system, the top and the bottom of the bar show the performance relative to the pessimistic 1MB and the optimistic 2MB on-chip SRAM caches, respectively. As a result, the relative performance of the on-chip DRAM system should be within the range depicted by the vertical bars. As Fig. 8(a) shows, on-chip DRAM cache system outperforms even the optimistic on-chip SRAM cache one in all evaluated benchmarks. In the swim benchmark, the 16MB on-chip DRAM cache system achieves almost the same performance enhancement over the SRAM cache systems as the 32MB on-chip DRAM main memory system. In the other two floating point benchmarks, tomcatv and applu, performance enhancements over the 16MB on-chip main memory system are observed. As discussed in the Section 5, while page swapping in the on-chip DRAM main memory greatly degrades the performance of the multiprogrammed benchmarks, the on-chip DRAM cache still shows large performance enhancements. In the four process multiprogrammed benchmark, the 16MB on-chip DRAM cache system is 98% better than the pessimistic 1MB SRAM system and 49% better than the optimistic 2MB one. This is much more than the original improvement of 86% and 16% in the 32MB on-chip DRAM main memory system. Compared to the local miss ratio of 21.4% in the 4MB L2 cache, the higher local miss ratios of 33.9% and 43.1% are observed in 2MB and 1MB L2 caches, respectively. The large working set of the benchmark greatly increases the miss ratio of the smaller L2 SRAM caches. As a result, a wide range of the relative performance is observed. The on-chip DRAM cache system outperforms the on-chip SRAM cache systems both in the four and six process VCS multiprogrammed benchmarks. Simply because of their lower local miss ratios which are 5.3% and 8.3%.

Fig. 8(b) shows the performance range of the on-chip DRAM cache system relative to the on-chip SRAM cache one in the uniprocessor system. The on-chip DRAM cache with the uniprocessor system is slightly better than the on-chip SRAM cache one in the swim benchmark. In other two floating point benchmarks, the effects of the on-chip DRAM cache disappear. Also, the on-chip DRAM cache system is still worse on the one process VCS benchmark.

These results again show that the effects of the on-chip DRAM cache are most significant in fairly complex high-performance processors, such as single chip multiprocessor.

.

(a) a 4x2-way multiprocessor system:



(b) a 2-way uniprocessor system:

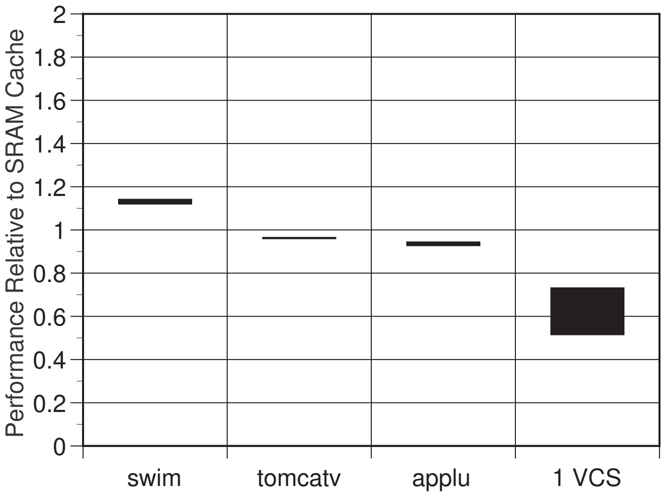


Figure 8 Performance range of the 16MB on-chip DRAM L2 cache system relative to the on-chip SRAM L2 cache system.

7 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, which 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. Similarly, a multiprogrammed benchmark with large working sets is improved 86% in the multiprocessor system but only 16% in the uniprocessor one. On the other hand, 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.

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 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. We also evaluate a system with the on-chip DRAM is designed as a very large on-chip cache. Even when the large working sets don't fit into the on-chip DRAM, the performance enhancement allowed by the reduction in off-chip memory traffic is significant in the multiprocessor system. However, performance improvements almost disappear in a uniprocessor system.

These results show that a small uniprocessor cannot effectively take advantage of the large bandwidth provided by the embedded DRAM architecture. However, when high density DRAM is integrated with a fairly complex high performance processor, such as a single chip multiprocessor, system performance can be dramatically improved in applications that have large working sets and can therefore use processor resources effectively.

References

[1] 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.

[2] 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.

[3] 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.

[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] 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.

[6] 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.

[7] T. Yamauchi, K. Tanaka, K. Furutani, Y. Morooka, H. Miyamoto, and H. Ozaki, "Fully Self-timing Data-Bus Architecture for 64-Mb DRAMs," IEICE TRANS. ELECTRON., VOL. E78-C, NO.7, pp. 885-865, 1995.

[8] 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.

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

[10] 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.

[11] 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.