The Energy Efficiency of IRAM Architectures

DRAFT COPY, November 10, 1996

Richard Fromm, Stylianos Perissakis, Neal Cardwell, Christoforos Kozyrakis
Bruce McGaughy, and David Patterson

University of California at Berkeley
Berkeley, CA 94720

Abstract

Portable systems demand energy efficiency in order to maximize battery life. IRAM architectures, which combine DRAM and a processor on the same chip in a DRAM process, have both performance and energy efficiency advantages compared to conventional systems. Such an organization of the memory hierarchy permits a much larger amount of memory on-chip than a traditional SRAM cache design in a logic process. IRAM provides a low-latency, high-bandwidth interface between the processor and memory which can lead to increased performance. Depending on the benchmark and the slowdown of a processor implemented in a DRAM process, the performance of IRAM varies from 2 times slower to 1.5 times faster. Since most or all IRAM memory accesses can be satisfied on-chip, there is much less need to drive high-capacitance off-chip buses, which contribute significantly to the energy consumption of a system. By using less energy for a given task, an IRAM will more efficiently utilize the available battery life in a portable system. We find that the IRAM memory hierarchy is as much as 4.5 times as energy efficient as a conventional memory hierarchy.

1 Introduction

Energy efficient computing is growing in importance. Sales of laptop and notebook computers have been steadily climbing, and applications for portable computing are likely to continue to grow in the near future, in areas such as PDAs (personal digital assistants), smart phones, and GPS (global positioning system) receivers, and other "anywhere-anytime" consumer computing devices [19][12].

This area has received a lot of research attention [7][18][8][9][22]. The increasing prevalence of portable computing has promoted energy efficiency from a concern primarily of circuit designers to an issue of general interest to the computer architecture community. Both the panel session at ISCA 1996 on industry recommendations for research topics [26] and the NSF workshop on Future Directions in Computer Architecture [5] recommended it as a promising area of research.

The share of the energy in a system consumed by the CPU and external memory is growing [21], and a large percentage of CPU energy is devoted to on-chip memory [25][32][33][18]. Our goal is to reduce the energy consumed by the memory system.

Integrating a microprocessor and DRAM memory on the same die, an idea that we call Intelligent RAM (IRAM)[27], offers the potential for dramatic improvements in the energy consumption of the memory system. DRAM is much denser than SRAM, which is traditionally used for on-chip memory. Therefore, an IRAM will have much fewer external memory accesses, which consume a great deal of energy to drive high-capacitance off-chip buses. Even on-chip accesses will be more energy efficient, since DRAM consumes less energy than does SRAM. Finally, an IRAM has the potential for higher performance than a conventional approach. Since higher performance for some fixed energy consumption can be translated into equal performance at a lower amount of energy, the performance advantages of IRAM can be translated into lower energy consumption.

Previous work has examined how IRAM can be utilized to improve performance [28][34]. This has been motivated by the exponentially growing performance gap between processors, which are increasing in performance at a rate of 60% per year, and memory access times, which are only getting approximately 7% faster per year [20][37]. An IRAM has both lower latency and higher bandwidth between the processor and the memory compared to a conventional architecture, offering the potential for high performance. The contribution of this paper is to evaluate to what extent an IRAM design reduces the energy consumption of the memory system.

We compare an IRAM to a conventional approach for two different die sizes, using appropriate ranges to account for the potential differences in area between DRAM and SRAM and the speed differences of a processor in a DRAM vs. logic process. Energy consumption of the memory hierarchy and performance are calculated for the various architectures. We find that the IRAM memory hierarchy can be as much as 4.5 times more energy efficient, and is never significantly less efficient. Depending on the slowdown of a processor implemented in a DRAM process and the benchmark, the IRAM varies from 2 times slower to 1.5 times faster.

The remainder of the paper is organized as follows: Section 2 discusses power and energy metrics. Section 3 discusses energy concerns in current computer systems. Section 4 presents several alternative architectures and our methodology for evaluating these architectures. Section 5 presents the results of our evaluations. Section 6 summarizes related work, Section 7 suggests future work, and Section 8 presents our conclusions. A more detailed discussion of the energy consumption models used for our evaluations is included as an Appendix.

2 Power and Energy Metrics

Various metrics provide different perspectives on the characteristics of a portable computer system.

2.1 Power

Traditionally, the unit of concern for energy efficient designs has been power, measured in Watts. Power represents the amount of energy consumed per unit time: $Power = Energy/Time$.

Power is a useful metric for certain concerns. The amount of power required by a system determines, for a fixed voltage, how much current a battery or other power source needs to be able to deliver. Also, higher power leads to high temperature which in turn leads to more expensive packaging and a cooling system that can be more expensive, larger, and/or heavier.

2.2 Energy

As anyone who has ever used a portable computer can attest, the overriding concern for a user of such a system is battery life. Battery life is measured in units of energy, not power. Energy is the product of power and time, i.e., $Energy = Power \cdot Time$. For a given amount of work, what matters most to the user is how much energy is required to do that work.

We believe that energy efficiency is a better metric than power for measuring how a given machine best utilizes limited battery life. Energy per Instruction and MIPS per Watt are inversely proportional to each other, as can be seen from the following relationship:

$$\frac{Energy}{Instruction} = \frac{Energy/Sec}{Instruction/Sec} \propto \frac{Watts}{MIPS}$$

Power can be a deceiving metric, since it does not directly relate to battery life. For instance, if the clock rate of a processor is cut in half, then the processor will consume approximately half the amount of power, since $Power = Frequency \cdot Capacitance \cdot Voltage^2$. However, if all else is kept equal, it will now take approximately twice as long to execute a sequence of instructions. Thus the energy required for some given amount of work will be roughly the same. As an extreme case, slowing the frequency down to zero and putting the processor in sleep mode wins if power is the only metric, yet this allows no work to be accomplished.

2.3 Performance

Besides energy efficiency, users are also concerned about performance. Users of portables are willing to tolerate lower performance than a desktop system has, but performance still matters. Portable systems are being used increasingly for more demanding applications that require higher performance. Examples include preparing large, graphical presentations, audio and video playback, handwriting recognition, and speech recognition. The goal is to be able to deliver high performance while still maintaining energy efficiency.

2.4 Other Metrics

Both performance (which can be expressed as $MIPS$) and energy efficiency (which can be expressed as $MIPS/Watt$) are relevant metrics. Gonzalez and Horowitz propose that their product (namely $MIPS^2/Watt$) be used as a single metric with which to rate a system, since this meets the objective of both factors in a single number [18]. They refer to this as the *energy-delay product*.

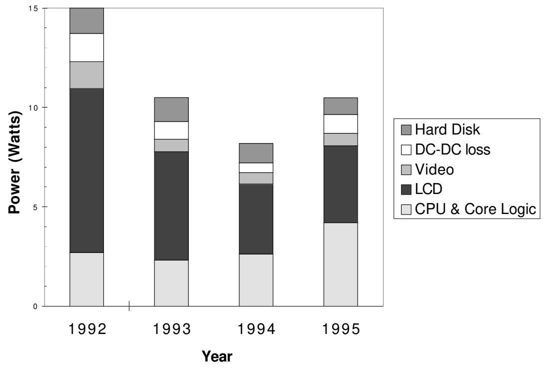

Figure 1: Notebook Power Budget Trends

3 Energy Concerns in Current Systems

3.1 Processors and Memory

Given that portable devices, and hence energy efficient systems, are becoming more prevalent, it is useful to examine where the energy is consumed in such a device. Figure 3.1 shows the breakdown of the power consumption over time in IBM ThinkPad notebook computers [21]. Whereas the power used to be dominated by the screen, over time the CPU and memory are becoming an increasingly significant portion of the power budget. In other portable devices, in which there may not be a large, high resolution, color LCD screen, the power consumption of the CPU and memory would be an even higher percentage of the total.

The demand for energy efficient systems and the increasing relative importance of the power consumption by the CPU and memory system have prompted the design of general purpose microprocessors with very low power consumption. For example, StrongARM [25][32], a microprocessor by Digital that implements the ARM instruction set, delivers 183 Dhrystone MIPS at 160 MHz while dissipating under 0.5 W. As the energy consumed by the CPU core is reduced, this places an even greater emphasis on the energy consumed by the memory system. On StrongARM, the 32 KB of on-chip caches consume 43% of the CPU power.

Similarly, Sato, Nagamtsu, and Tago [33] use a power simulator of a generic RISC processor, examining both the current drawn by the various components and the percentage of time those components are used. Their results indicate that the instruction and data caches consume 60% of the total processor power.

Gonzalez and Horowitz [18] show in their simulations that 50-80% of the energy on a microprocessor is dissipated by the clock and on-chip caches, with approximately half of that coming from the caches. They note that although each individual memory cell dissipates very little energy, the total represents a very significant sum. They are pessimistic that the energy-delay product (as defined in Section 2.4) can improve significantly since the energy is dissipated in so many small units. They feel that only a "radical new architecture" can significantly improve this situation.

3.2 Processor-Memory Integration

Although the concept of IRAM may represent a somewhat "radical" implementation, the integration of processor and memory has the potential to improve both energy efficiency and performance even with a relatively simple system architecture.

There are numerous ways in which processor and memory can be integrated: 1) putting SRAM memory with a CPU in a logic process; 2) putting CPU logic with memory in a DRAM process; 3) putting DRAM memory with a CPU in a logic process. The first idea represents the conventional approach, using SRAM caches for on-chip memory. The second idea is what we are suggesting for IRAM. The third idea has a number of technological disadvantages, including a significant loss in memory density and a much higher refresh rate for the DRAM, due to DRAM process optimizations which are not present in a logic process [15]. Therefore, we only consider the first two approaches.

IRAM has a number of advantages over a conventional approach in achieving energy efficiency. First, DRAM is more energy efficient than SRAM, so accesses to the on-chip memory consume less energy. More importantly, since DRAM is much denser (see Section 4.1), more memory can reside on-chip, allowing all memory accesses to stay on-chip for a wider range of applications or significantly reducing the frequency of off-chip accesses for a given application. Driving high-capacitance off-chip buses requires a tremendous amount of energy, so significantly reducing the number of off-chip accesses will greatly reduce the overall energy consumption.

IRAM also has the potential for higher performance than a conventional approach. Even without considering architectural models specifically tailored to exploit the low-latency, high-bandwidth characteristics possible from an IRAM organization (see Section 7), the significant density improvement for DRAM over SRAM can result in higher performance for IRAM. As with the energy efficiency advantages, the big win for IRAM is that many more memory accesses can stay on-chip, which can significantly reduce the average memory access time. However, there are two limitations which might offset this benefit. First, access to a DRAM array is somewhat slower than an SRAM array. Second, the logic of the CPU core is likely to run slower if implemented in a DRAM process instead of a logic process (see Section 4.2). The improvements due to the reduction in off-chip memory accesses will have to be greater than these slowdowns for there to be an overall increase in performance.

4 Methodology

To quantitatively investigate the energy efficiency and performance merits of IRAM, we compared the architectures listed in Figure 2. The models assume commercially available logic and DRAM semiconductor processes. They estimate the behavior of a small, conventional CPU in a logic process (Conventional-Small); that same CPU implemented in a DRAM process (IRAM-Small); a large DRAM die with a CPU added (IRAM-Large); and a similarly large die implemented in a logic process (Conventional-Large).

In order to determine the characteristics of the models used to evaluate IRAM vs. conventional architectures, two questions that need to be addressed are the area differences of DRAM and SRAM and the speed differences of logic in a DRAM vs. logic process.

4.1 Area Differences

As mentioned previously, the density of a DRAM array in a DRAM process is *much* higher than that of SRAM in a logic process. The ratio of the number of cells per unit area for DRAM vs. SRAM is much greater than the 4:1 or 6:1 figure that one would assume if the only relevant factor was the number of transistors, 4T or 6T for an SRAM cell vs. a single transistor for a DRAM cell [15]. DRAM storage cells use pseudo-3-dimensional trench or stacked capacitors to achieve very small cell sizes [29]. As Figure 3 shows, the DRAM cell size for a 64 Mb DRAM implemented in a 0.4 μm CMOS process [24] is 16 times smaller than the SRAM cell size for StrongARM [San96b]. If the DRAM feature size is scaled down so that the comparison is for the same size process (0.35 μm), then the cell size is 21 times smaller.

What is more important, however, is to compare the total amount of memory that can fit in a given area when all circuits and interconnect that make up the entire memory array are taken into account. Examining the cell efficiency (bits of memory per unit area) shows that the 64 Mb DRAM is effectively 39 times more dense than the StrongARM. After again scaling the DRAM parameters to make an equal process comparison, the DRAM is 51 times more dense!

These numbers can be used to obtain an approximate value for the differences in memory capacity of a conventional and an IRAM architecture, given a fixed chip area. However, it is difficult to precisely quantify this for all cases, due to a number of extenuating circumstances. The comparisons above use chips with very different die areas, and the actual ratio obtained between two implementations is somewhat dependent on the absolute size of the array – it is easier to make a memory array denser as it gets larger. Also, logic circuits in a DRAM process will be somewhat larger than logic circuits in a corresponding logic process. Therefore, to obtain equal total area, somewhat less area will be available for the memory array. Recognizing these factors, we are conservative when picking DRAM to SRAM capacity ratios used in the models, and instead of limiting our analysis to a single value, we use a range. The bounds of this range are obtained by rounding down the cell size and bits per unit area ratios to the nearest factors of 2, namely 16:1 and 32:1.

4.2 Speed Differences

Logic processes are optimized for fast transistors and interconnect, and DRAM processes are optimized for density and retention. Due to these different process characteristics, logic gates are slower in a standard DRAM process than a standard logic process, resulting in a CPU speed that is likely to be slower in an IRAM than a conventional processor [29]. Just how much slower logic operates in a DRAM process is not yet precisely known, although this is an active area for future research. The most pessimistic estimate we have been given is a slowdown by a factor of 2; optimistic estimates give approximately a 30% slowdown. To address this unknown factor, we calculate our results for a range of CPU speeds for the architectures implemented in DRAM technologies. We vary the CPU speed from half as fast (on the pessimistic side) to equal in speed (an optimistic upper bound) compared to the architectures implemented in logic processes.

4.3 Future Trends

While logic circuits implemented in a DRAM process have some drawbacks with respect to area and speed,

	Conventional-Small	IRAM-Small	Conventional-Large	IRAM-Large
CPU frequency	160 MHz	80-160 MHz	160 MHz	80-160 MHz
technology	0.35 μm logic	0.35 μm DRAM	0.35 μm logic	0.35 μm DRAM
L1 config	16 KB I + 16 KB D	8 KB I + 8 KB D	8 KB I + 8 KB D	8 KB I + 8 KB D
L1 associativity	32-way	32-way	32-way	32-way
L1 write policy	write-back	write-back	write-back	write-back
L1 block size	32 Bytes	32 Bytes	32 Bytes	32 Bytes
L1 type	SRAM on-chip	SRAM on-chip	SRAM on-chip	SRAM on-chip
L1 access time	1 cycle	1 cycle	1 cycle	1 cycle
L2 config	–	256 to 512 KB unified	256 to 512 KB unified	–
L2 associativity	–	direct-mapped	direct-mapped	–
L2 write policy	–	write-back	write-back	–
L2 block size	–	128 Bytes	128 Bytes	–
L2 type	–	DRAM on-chip	SRAM on-chip	–
L2 access time	–	30 ns	3 cycles (18.75 ns)	–
memory type	8 MB DRAM off-chip	8 MB DRAM off-chip	8 MB DRAM off-chip	8 MB DRAM on-chip
memory latency	180 ns	180 ns	180 ns	30 ns
bus width	narrow (32 bits)	narrow (32 bits)	narrow (32 bits)	wide (32 Bytes)

Figure 2: Architectural Models Used for Evaluation. The "Small" and "Large" models roughly correspond to the die sizes given in Figure 3. It is economically feasible to build large memory arrays by using redundancy. StrongARM's caches, in contrast, use no redundancy. Memory capacity and speed differentials between the "Conventional" and "IRAM" models are based on the arguments presented in Sections 4.1 and 4.2 respectively. All caches are write-back to minimize energy consumption from unnecessarily switching internal and/or external buses. The 30 ns on-chip DRAM access time is based on [24], the 180 ns off-chip access time is based on [13], and the 18.75 ns on-chip L2 SRAM cache access time is chosen to be slightly larger than the on-chip L2 cache of the Alpha 21164A [4], which is slightly smaller (96 KB). The "narrow" bus width matches StrongARM, and the "wide" bus width takes advantage of the additional bandwidth available to main memory in an IRAM configuration.

	StrongARM	64 Mb DRAM
process	0.35 μm CMOS	0.40 μm CMOS
memory cell size	26.41 μm^2	1.62 μm^2
number of memory bits	32 KB + tags = 287,744	64 Mb = 67,108,864
total chip area	49.9 mm^2	186.0 mm^2
total area of memory	27.9 mm^2	168.2 mm^2
bits per mm^2	10.31	398.9

Figure 3: Memory Cell Parameters For Typical Microprocessor [25][31] and DRAM Chips [24] Presented at ISSCC

ongoing trends in the DRAM industry may alleviate some of these difficulties. Future DRAM generations are likely to have additional metal layers, enabling more optimal layout of logic. DRAM processes are also beginning to incorporate faster transistors to support synchronous DRAM. And some DRAM manufacturers are already developing merged logic and DRAM processes in an attempt to get the best features of both on a single die.

4.4 Architectural Models

The Conventional-Small design is architecturally similar to StrongARM [25][32], a low-power implementation of the ARM architecture by Digital that delivers 183 Dhrystone MIPS while dissipating under 0.5 W at 160 MHz.[1] [2]

The IRAM-Small model addresses what the Conventional-Small CPU would look like if implemented in a DRAM process given current technology, and if the overall chip area was kept constant. Since a single-cycle first level cache access is desirable for good performance, and access to an on-chip DRAM array is slower than that, we chose *not* to simply change the 32 KB of SRAM cache into a DRAM cache of similar area for the IRAM-Small model. Instead, we split the area originally allocated for the cache into two. Half of the area is allocated to a conventional L1 SRAM cache, and the remaining area is implemented as DRAM, organized as an L2 cache.

We consider L2 cache sizes of 256 KB and 512 KB, which corresponds to a density increase for DRAM compared to SRAM of 16:1 and 32:1 respectively (see Section 4.1). We consider CPU speeds between 80 and 160 MHz, which covers a range of speeds for logic in a DRAM process from a factor of 2 slowdown to no slowdown (see Section 4.2).

The IRAM-Large model approaches the problem from a different angle: instead of starting with an existing CPU architecture and modifying it to be implemented in a DRAM process (with comparable total area), we start with a DRAM chip and add a CPU. We choose a 64 Mb DRAM chip, which is roughly comparable to 0.35 μm logic technology; both represent the state of the art available today in commercial implementations. This gives 8 MB of DRAM memory on-chip, which is large enough to be considered the main memory for many applications, especially in portable systems. We therefore assume that the on-chip DRAM array is main memory rather than an L2 cache. All memory references are assumed to be satisfied on-chip. Just as in the previous case (IRAM-Small), we desire for the CPU to be able to access most memory references in 1 cycle, so we again add a first level SRAM cache (again 8 KB I + 8 KB D). Based on the same arguments as before (see Section 4.2), we consider CPU speeds of 80 and 160 MHz to address the speed differential between logic in DRAM and logic processes.

The Conventional-Large model rounds out the four architectures studied by assuming roughly the same die area as the IRAM-Large model, but using a logic process instead. The large, on-chip memory array is therefore composed of SRAM cells. Based on the same arguments as before (see Section 4.1), we model both a 16:1 and 32:1 ratio of DRAM to SRAM array areas. This gives SRAM array sizes of 256 KB and 512 KB. This is unlikely to be adequate for main memory, even for portable applications. Consequently, we assume that this is treated as an L2 cache, with conventional (off-chip) DRAM main memory as the next level in the memory hierarchy.

4.5 Simulations

Energy consumption and performance are calculated for each of the architectural models for the benchmarks shown in Figure 4.[3] The first four benchmarks are from the SPECint95 benchmark suite [3]. The final benchmark, hsfsys, is a handwriting recognition package from the National Institute of Standards and Technology (NIST) [16]. All of these are representative of current applications for a portable computing environment.

Each of the benchmarks was simulated for each of the architectural models using the SimOS simulation environment [30]. The simulations include both user and operating system activity for a system using the MIPS ISA. Activity at each of the levels of the memory hierarchy was recorded. Additionally, we

[1] While 32-way set associativity may seem somewhat excessive, the StrongARM designers note that only 4-way associativity was desired for performance goals. Additional design requirements of single cycle access and bank selection led to the highly associative cache as a result. See [32] for details. We choose to keep the same 32-way set associativity for the L1 cache for all of our models to enable fair comparisons.

[2] NOTE TO REVIEWERS: While our energy calculations assume 32-way set associativity, our current performance numbers use 4-way set associativity. This was due to an existing limitation in the SimOS simulator. We note that the performance benefits of 32-way vs. 4-way associativity are marginal [20][13], so we do not expect this to significantly impact our results. We will update the simulator and redo our calculations using 32-way set associativity prior to the due date for the final version.

[3] NOTE TO REVIEWERS: Due to time constraints, we were not able to run the complete data sets for all of the benchmarks. We feel that the cases presented here are sufficiently long to adequately demonstrate our results and conclusions. We will run the simulations fully to completion prior to the due date for the final version.

benchmark	instructions	description
gcc	39 billion	GNU C compiler; generates SPARC assembly
compress	44 billion	Compresses and decompresses file in memory
li	65 billion	LISP interpreter
perl	45 billion	Manipulates strings (anagrams) and prime numbers in Perl
hsfsys	5.4 billion	Form-based handwriting recognition system

Figure 4: Benchmarks Used For Evaluation

determined the base cycles per instruction (CPI), as if there were no stalls due to memory references, using `pixie`, a code profiling utility. Final performance numbers were computed by combining the base CPI with the miss rates and latencies at the various levels of the memory hierarchy. Our CPU model uses no branch prediction, assumes a single cycle branch delay, and stalls on cache read misses, like StrongARM. The off-chip latency is the time to return the critical word. Like StrongARM, we assume the CPU continues execution while the rest of the cache block is fetched. We assume a write buffer big enough so that the CPU does not have to stall on write misses.

The models for energy consumption capture the dominant effects in DRAM and SRAM memories. Technology parameters were taken from typical DRAM circuits of the 64 Mb generation [24] [38] [36] and contemporary microprocessor SRAM caches generations [13] [11] (see Figure 5), which represent the state of the art commercially available in 1996. See the Appendix for a more detailed explanation of how the simulation results were used to calculate energy consumption.

It should be noted that for our energy efficiency comparisons we are only addressing the energy consumed by the memory hierarchy, and not by the CPU core. Previous work [7][12] has already addressed techniques for reducing the CPU energy consumption; this makes the energy consumption of the memory hierarchy more significant.

5 Evaluation

5.1 Energy

Figure 6 shows the energy consumption of the memory hierarchy for each of the benchmarks for each of the models. The energy consumption is divided into its various components, including L1 instruction and data caches, L2 cache, and main memory. The energy at each level includes the bus to access that level. Only a single value is given for each configuration, even the DRAM configurations where the CPU speed is varied, since the energy consumed by the memory system does *not* depend on CPU frequency.

As can be seen from the figure, the various IRAM configurations can significantly reduce the amount of energy consumed per instruction compared to the corresponding conventional cases. For the small chips, the IRAM architectures are as much as 4.5 times more energy efficient; for the larger chips the IRAM is as much as 3 times more energy efficient. In the worst case, the IRAM memory hierarchy energy efficiency is approximately the same as for a conventional memory hierarchy.

This can be easily explained by considering the following equation that relates the energy consumption at each level of the memory hierarchy with the frequency of accesses to the various levels. It is closely modeled after the familiar equation for average memory access time [20]:

Energy per instruction = *L1 access energy* + *L1 miss rate* · (*L2 access energy* + *L2 miss rate* · *off-chip access energy*)

It is clear from this equation that there are two ways to reduce the energy consumption per instruction: 1) reduce the energy to access various levels of the memory hierarchy; 2) reduce the frequency of accesses to lower levels (i.e. reduce the miss rate). IRAM has the ability to reduce both of these.

For accesses that stay on-chip, accessing a DRAM array consumes less energy than accessing an SRAM array. For accesses that go off-chip, the energy consumption is roughly the same, since the major component in this case is driving the high capacitance off-chip buses.

The big win for IRAM is that by having a DRAM array on-chip instead of an SRAM array, the on-chip memory can be *much* larger. Consequently, the IRAM configurations will have much lower miss rates and will not have to pay the significant energy penalty for going off-chip as frequently. In the IRAM-Large case, where the main memory array is on-chip, all memory accesses can be satisfied without paying this high energy cost.

There are some minor offsetting factors; for example, the IRAM-Small configuration has an L1 cache that is half of the size of the Conventional-Small con-

	DRAM	SRAM (L1)	SRAM (L2)
Internal power supply	2.2V	1.5V	1.5V
Bank width	256 bits	64 bits	128 bits
Bank height	512 bits	64 bits	512 bits
Bit line swing (read)	1.1V	0.5V	0.5V
Bit line swing (write)	1.1V	1.5V	1.5V
Sense amplifier current	–	150μA	150μA
Bit line capacitance	250fF	256fF	2048fF

Figure 5: Major Technology Parameters Used in Memory Hierarchy Models

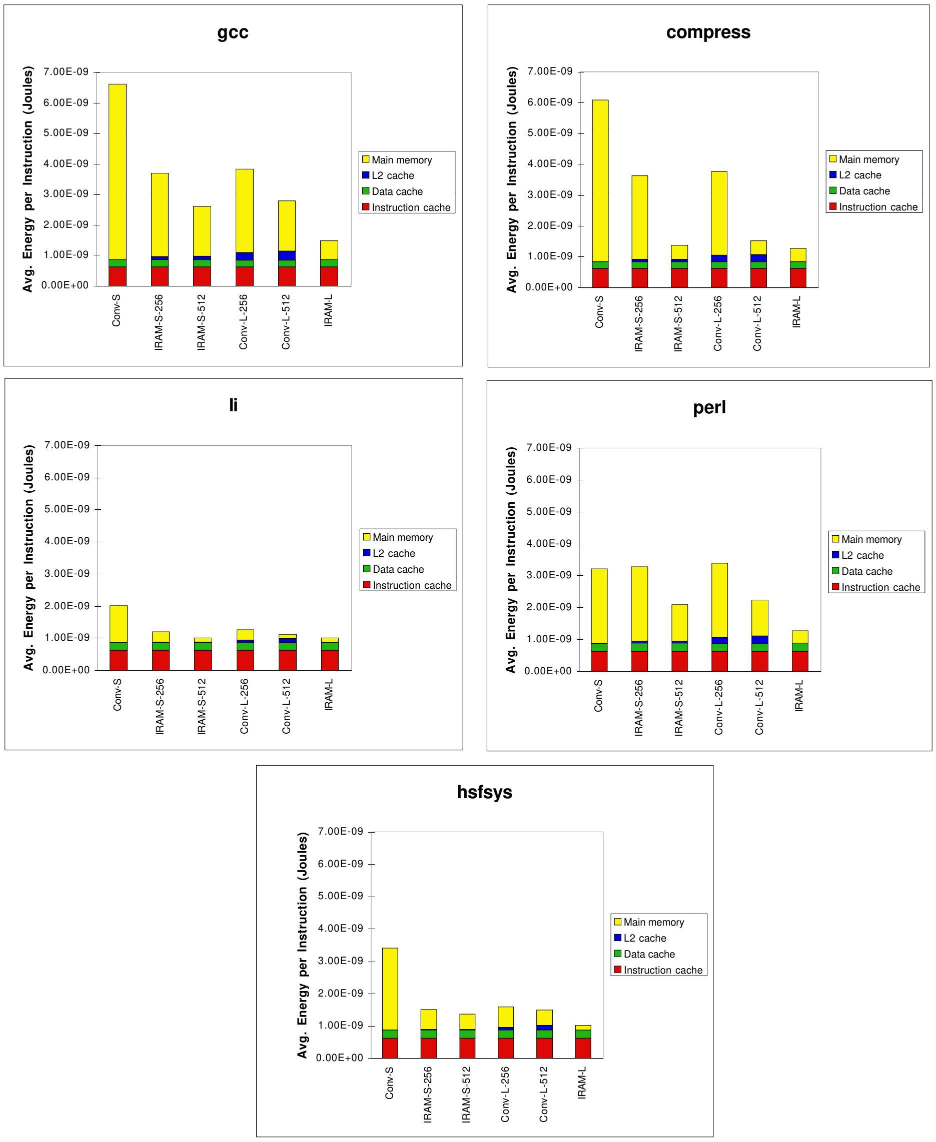

Figure 6: Energy Consumption of Memory Hierarchy

figuration, giving it a higher L1 miss rate and forcing it to access its next level (L2 DRAM cache) on some occasions in which the Conventional-Small case hits in its L1 cache. This factor is so small compared to the savings from going off-chip less often that, in most cases, there is a significant reduction in the energy consumption of the memory hierarchy by integrating the processor with DRAM.

5.2 Performance

Figure 7 shows the performance of each of the models on each of the benchmarks. This is shown for a range of frequencies for the CPU core implemented in a DRAM process compared to a logic process: 0.5 means half the speed of a CPU in a logic process, and 1.0 implies equal. The IRAM cases are represented as lines that span the range of frequencies, because, as noted earlier, the precise slowdown of logic in a DRAM process is unknown. The conventional cases are really just single points at the right edge of the graphs. We draw them as flat dotted lines across the range of frequencies to enable easy performance comparisons. IRAM gives higher performance at the regions of operation for which it lies above the corresponding conventional line.

For the small chips, the IRAM architectures range from 1.8 times as slow to 1.5 times as fast as the corresponding conventional cases. For the large chips, IRAM ranges from 2 times as slow to 1.1 times as fast.

There are two opposing factors that determine the speed of the IRAM architectures relative to the conventional architectures. There is the potential for IRAM to have lower performance than a conventional processor, because the processor's operating frequency may be limited by the DRAM process in which it is implemented. Yet there is also the potential for IRAM to have higher performance, because many more of the memory accesses can be satisfied in the low-latency, high-bandwidth on-chip memory, due to the much higher density of DRAM than SRAM.

Which of these factors will dominate is a function of both the physical organization and the application. For an application that is CPU-bound and is not heavily dependent on the memory hierarchy, it is possible that a naive IRAM architecture might hurt performance. But for an application that is heavily dependent on the memory hierarchy and is already not fully utilizing the CPU, IRAM has the potential for a large performance gain. For instance, [10] found that a 300 MHz Alpha 21164 microprocessor spends about 75% of its time in the memory hierarchy for database and matrix computations. As the performance gap between processors and memory continues to increase, as applications have more demanding memory requirements, and as DRAM capacities continue to increase beyond the 64 Mb used in this study, the performance advantages of IRAM will grow.

6 Related Work

Other researchers have investigated a closer integration of processors and memory. However, previous discussions have concentrated on the potential performance benefits. This paper is the first to address the energy efficiency advantages as well.

Mitsubishi has announced the M32R/D, which integrates a processor and 2 MB of DRAM and is targeted for personal digital assistants [35]. They concluded that integrating a processor with memory significantly reduced power dissipation.

Researchers at Sun [34] evaluated the performance of a RISC processor on a 256 Mb DRAM, using the sense amps as caches. They arbitrarily limited their additions to be about 10% of the DRAM size and found that they could achieve comparable integer performance and about half the floating point performance compared to a 1995 Alpha.

A chip by IBM is similar to our IRAM-Small approach in that it implements an L2 cache with DRAM [17].

NEC has developed a chip that combines 16 Mb of DRAM with 128 1-bit processors for image-processing applications [6].

Finally, there have been numerous efforts at integrating processors and memory for parallel processing, including Execube [23], a chip with eight 16-bit processors each with 64 KB of memory, and the M-machine [14], with four superscalar processors and 128 KB per processor.

Both the Galileo project at the University of Wisconsin [1] and the PPRAM project at Kyushu University in Japan [2] are also exploring topics in processor-memory integration.

7 Future Work

There is much more work to be done in this area, concerning both low level circuit issues and high level architectural issues. The precise speed and area tradeoffs of processors and memory in both logic and DRAM processes need to be better quantified. The physical implications (including temperature and noise) of closely integrating logic and memory need to be studied. For instance, it has been suggested [17] that a rule of thumb is that for every

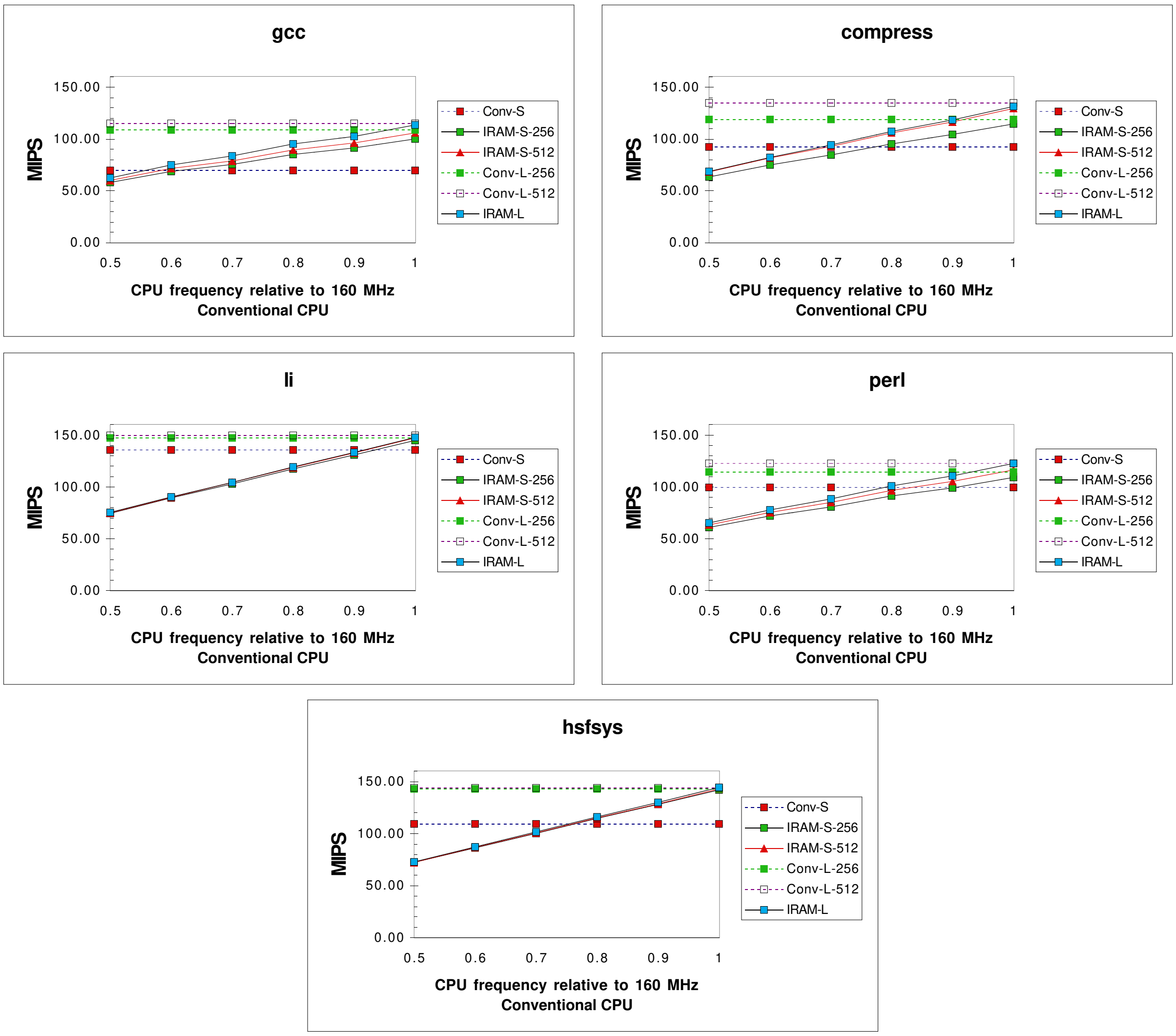

Figure 7: Performance as a function of processor clock rate

increase of 10 degrees Celcius, the minimum necessary refresh time of a DRAM is cut in half. Research in process development would be useful in these and other areas. Perhaps the best realization of processor-memory integration can be achieved in a hybrid CMOS process that incorporates the best features of both logic and DRAM processes.

Finally, an IRAM organization gives us the opportunity for both lower latency and dramatically higher bandwidth between the processor and memory. Investigating new ideas and organizations that can turn these opportunities into significantly increased performance is an exciting and open area of research in architecture as well as compilers and operating systems. While this study concentrated on the benefits of IRAM for energy-conscious systems, this is certainly not the only realm in which IRAM may play an important role in redefining our notion of processor and memory system design.

8 Conclusion

Last year at ISCA, panel members asked the Computer Architecture research community to examine power and also to study the integration of a processor and DRAM. This paper is one answer to that call. We examined two head-to-head comparisons between a conventional logic implementation and an IRAM. The IRAM memory hierarchy is as much as 4.5 times as energy efficient as conventional memory hierarchies. Depending on the benchmark and the slowdown of a processor implemented in a DRAM process, the IRAM varied from 2 times slower to 1.5 times faster.

Acknowledgments

This research was supported by DARPA (DABT63-C-0056), the California State MICRO Program, and by research grants from Intel and Sun Microsystems. We are grateful to Dan Dobberpuhl and Sribalan Santhanam of Digital Equipment Corporation for their helpful answers to our questions about StrongARM. Thanks to Krste Asanovic for his role as an indispensable technical reference and to Randi Thomas and Scott McPeak for useful feedback on earlier drafts of this paper.

Appendix

Energy dissipation modeling

The dominant factors of energy consumption in SRAM caches, DRAM caches, and external memory were captured in a spreadsheet. Typical values of circuit parameters, such as bit and word line capacitances and memory bank dimensions, were used [24] [38] [22] [36] [13] [11]. (see Figure 5).

The dominant factor in DRAM energy dissipation is the capacitance of the bit lines being driven to the power supply rails. SRAM power dissipation is dominated by the sense amplifiers when reading, because the swing of the bit lines is low. However, to write the SRAM, the bit lines are driven to the rail, so their capacitance becomes the dominant factor when writing.

For all models, the first-level instruction and data caches were closely modeled after the StrongARM caches, which are 32-way set-associative and are implemented as 16 banks. The tag arrays are implemented as Content-Addressable Memories (CAMs). This was done mainly to reduce power, since the conventional way of accessing a set-associative cache, reading all the lines in a set and then discarding all but one, is clearly not energy efficient. The second level unified cache is assumed to consist of the appropriate number of 512-by-256 DRAM banks, or 512-by-128 SRAM banks. This is organized in the conventional way, as opposed to CAM-based, since it is direct mapped and the CAM array is not area-efficient.

The IRAM model consists of 512 128Kbit subarrays, like most high-density DRAMs. On-chip L2 caches, as well as the on-chip main memory, have 256-bit wide interfaces to the first level caches. In the case of IRAM, this is a significant departure from the common 4- or 8-bit wide memories and one of the main performance advantages of IRAM-based architectures.

For external memory, for a fair comparison we used a single 64Mbit chip. This of course assumes that such chips with 32-bit wide interfaces will be available. This choice clearly minimizes the external memory power, both in the DRAM chips themselves (just one in our case) and in the bus, that is as small as possible. If such chips are not available, external power consumption will be higher and the IRAM advantage more pronounced.

Having calculated the energy dissipated per access to each level of the memory hierarchy, we combine these results with the miss rates reported by SimOS to calculate the average energy per instruction as described in Section 4.5.

References

[1] Galileo project. http://www.cs.wisc.edu/ galileo/. Computer Sciences Department, University of Wisconin - Madison.

[2] PPRAM project. http://kasuga.csce.kyushu-u.ac.jp/ ppram/. Computer Science Division, Kyushu Univeristy, Japan.

[3] SPEC CPU95 benchmarks. http://www.specbench.org/ osg/cpu95/.

[4] Alpha 21164 microprocessor hardware reference manual, 1995.

[5] NSF workshop on critical issues in computer architecture research. http://www.cise.nsf.gov/ mips/MSAWorkshop96/, May 1996.

[6] Y. Aimoto et al. A 7.68 GIPS, 3.84 gb/s 1W parallel image-processing RAM integrating a 16 Mb DRAM and 128 processors. In *Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, pages 372–373, 476, San Francisco, CA, February 1996.

[7] B. Burd and R. Broderson. Energy efficient CMOS microprocessor design. In *Proroceedings of the Twenty-Eighth Hawaii International Conference on System Sciences*, volume 1, pages 288–297, Los Alamitos, CA, January 1995.

[8] A. Chandrakasan. *Low Power Digital CMOS Design*. PhD thesis, University of California at Berkeley, 1994.

[9] A. Chandrakasan et al. System design of a multimedia I/O terminal. In *Proceedings of IEEE Workshop on VLSI Signal Processing*, pages 57–65, Veldhoven, Netherlands, October 1993.

[10] Z. Cvetanovic and D. Bhandarkar. Performance characterization of the Alpha 21164 microprocessor using TP and SPEC workloads. In *Proceedings, Second International Symposium on High-Performance Computer Architecture*, pages 270–280, San Jose, CA, February 1996.

[11] David DiMarco et al. A 200MHz 256kB second-level cache with 1.6GB/s data bandwidth. In *Proceedings IEEE International Solid-State Circuits Conference*, volume 39, pages 158–159, February 1996.

[12] D. Dobberpuhl. The design of a high performance low power microprocessor. In *Digest of Technical Papers, 1996 International Symposium on Low Power Electronics and Design*, pages 11–16, Montery, CA, August 1996.

[13] D. Dobberpuhl. Personal communication, November 1996.

[14] M. Fillo et al. The M-Machine multicomputer. In *Proceedings of the 28th Annual International Symposium on Microarchitecture*, pages 146–156, Ann Arbor, MI, November 1995.

[15] R. Foss. Implementing application specific memory. In *Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, pages 260–261, 456, San Francisco, CA, February 1996.

[16] M. Garris et al. *Public Domain OCR: NIST Form-Based Handprint Recognition System*. National Institute of Standards and Technology, http://www.nist.gov/ itl/div894/878.03/databases/defs/nist_ocr.html.

[17] G. Giacalone et al. A 1 MB, 100 MHz integrated L2 cache memory with 128b interface and ECC protection. In *Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, pages 370–371, San Francisco, CA, February 1996.

[18] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, September 1996.

[19] E. Harris et al. Technology directions for portable computers. In *Proceedings of the IEEE*, volume 83, pages 636–658, April 1995.

[20] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, Inc, San Francisco, CA, 1996.

[21] T. Ikeda. Thinkpad low-power evolution. In *Digest of Technical Papers, 1995 IEEE Symposium on Low Power Electronics*, pages 6–7, San Jose, CA, October 1995.

[22] K. Itoh, K. Sasaki, and Y. Nakagome. Trends in low-power ram circuit technologies. In *Proceedings of the IEEE*, volume 83, pages 524–543, April 1995.

[23] P. Kogge. Execube - a new architecture for scaleable mpps. In *Proceedings of the 1994 International Conference on Parallel Processing*, volume 1, pages 77–84, Raleigh, NC, August 1994.

[24] H. Koike et al. A 30ns 64Mb DRAM with built-in self-test and repair function. In *Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, pages 150–151, 270, San Francisco, CA, February 1996.

[25] J. Montanaro et al. A 160MHz 32b 0.5W CMOS RISC microprocessor. In *Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, pages 214–215, 447, San Francisco, CA, February 1996.

[26] D. Patterson. Panel: Research opportunities and critiques: An industrial perspective, May 1996.

[27] D Patterson, T. Anderson, and K. Yelick. A case for intelligent DRAM: IRAM. In *HotChips VIII*, pages 75–93, Stanford, CA, 1996.

[28] D. Patterson et al. Intelligent ram (IRAM): Chips that remember and compute. In *To appear in 1997 IEEE International Solid-State Circuits Conference*, San Francisco, CA, February 1997.

[29] Steven A. Przybylski. *New DRAM Technologies: A Comprehensive Analysis of the New Architctures*. MicroDesign Resources, Sebastopol, California, 1994.

[30] M. Rosenblum et al. Complete computer system simulation: The SimOS approach. In *IEEE Parallel and Distributed Technology: Systems and Applications*, volume 3, pages 34–43, Winter 1995.

[31] S. Santhanam. Personal communication, October 1996.

[32] S. Santhanam. StrongARM SA110: A 160MHz 32b 0.5W CMOS ARM processor. In *HotChips VIII*, pages 119–130, Stanford, CA, 1996.

[33] T. Sato, M. Nagamatsu, and H. Tago. Power and performance simulator: ESP and its application for 100MIPS/W class RISC design. In *Digest of Technical Papers, 1994 IEEE Symposium on Low Power Electronics*, pages 46–47, San Diego, CA, October 1994.

[34] A. Saulsbury, F. Pong, and A. Nowatzyk. Missing the memory wall: The case for processor/memory integration. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 90–101, Philadelphia, PA, May 1996.

[35] T. Shimizu et al. A multimedia 32b RISC microprocessor with 16Mb DRAM. In *Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, pages 216–217, 448, San Francisco, CA, February 1996.

[36] Hitoshi Tanaka et al. Stabilization of voltage limiter circuit for high-density drams using miller compensation. *Electronics and Communications in Japan, Part 2*, 76(4):60–70, 1993.

[37] W. Wulf and S. McKee. Hitting the memory wall: Implications of the obvious. *Computer Architecture News*, 23(1):20–24, March 1995.

[38] Toshio Yamada et al. A 64Mb DRAM with meshed power line and distributed sense-amplifier driver. In *Proceedings IEEE International Solid-State Circuits Conference*, volume 34, pages 108–109, February 1991.