A CASE FOR INTELLIGENT RAM

David Patterson

Thomas Anderson

Neal Cardwell

Richard Fromm

Kimberly Keeton

Christoforos Kozyrakis

Randi Thomas

Katherine Yelick

University of California, Berkeley

Merging processing and memory into a single DRAM chip could revolutionize the semiconductor industry.

Two trends call into question the current practice of fabricating microprocessors and DRAMs as different chips on different fabrication lines. The gap between processor and DRAM speed is growing at 50% per year; and the size and organization of memory on a single DRAM chip is becoming awkward to use, yet size is growing at 60% per year.

Intelligent RAM, or IRAM, merges processing and memory into a single chip to lower memory latency, increase memory bandwidth, and improve energy efficiency. It also allows more flexible selection of memory size and organization, and promises savings in board area. This article reviews the state of microprocessors and DRAMs today, explores some of the opportunities and challenges for IRAMs, and finally estimates performance and energy efficiency of three IRAM designs.

Why there is a problem

The division of the semiconductor industry into microprocessor and memory camps has many advantages. First and foremost, manufacturers can tailor a fabrication line to suit the device. Microprocessor fab lines offer fast transistors to make fast logic and many metal layers to accelerate communication and simplify power distribution. DRAM fabs, on the other hand, offer many polysilicon layers to achieve both small DRAM cells and low leakage current to reduce the DRAM refresh rate.

Separate chips also mean separate packages. So microprocessors can use expensive packages that dissipate high power (5 to 70 W) and provide hundreds of pins to make wide connections to external memory. And DRAMs can use inexpensive packages that dissipate low power (1 W) and use only a few dozen pins.

Separate packages, in turn, mean computer designers can scale the number of memory chips independently of the number of processors. Most desktop systems have one processor and 4 to 32 DRAM chips, but most server systems have 2 to 16 processors and 32 to 256 DRAMs. Memory systems have standardized on single in-line memory module (SIMM) or dual in-line memory module (DIMM) packaging, which allow the end user to scale the amount of memory in a system.

Quantitative evidence of the industry's success is its size: In 1995, DRAMs were a $37-billion industry, and microprocessors were a $20-billion industry. In addition to financial success, the technologies of these industries have improved at unparalleled rates. DRAM capacity has quadrupled on average every three years since 1976, while microprocessor speed has done the same since 1986.

However, the split into two camps has its disadvantages as well. Figure 1 shows that while microprocessor performance has been improving at a rate of 60% per year, the access time to DRAM has been improving at less than 10% per year. Hence computer designers are faced with an increasing processor-memory performance gap, which is now the primary obstacle to improved computer system performance.

System architects have attempted to bridge this gap by introducing deeper and deeper cache memory hierarchies. Unfortunately, this depth makes the memory latency even longer in the worst case. For example, Table 1 lists CPU and memory performance in a recent high-performance computer system—a 300-MHz Alpha 21164 processor running on an AlphaServer 8200 computer.

Note that this system's main memory latency is a factor of four larger than the raw DRAM access time. This difference is due to the time to drive the address off the micro-

0272-1732/97/$10.00 © 1997 IEEE

processor, multiplex the addresses to the DRAM, and turn around the bidirectional data bus, plus the memory controller overhead, the SIMM connectors' latency, and the time to drive the DRAM pins first with the address and then with the return data.

Despite huge on- and off-chip caches and very sophisticated processors with out-of-order, dynamically scheduled, superscalar pipelines that execute multiple instructions per clock cycle, the long latency and limited bandwidth to main memory dominate performance for some applications. For example, Table 2 shows clock cycles per instruction (CPI), cache misses, and fraction of time spent in each component of the Alpha 21164 for several benchmark programs. These include the SPEC92 integer and floating-point CPU benchmarks, a database program running a debit-credit benchmark, and a sparse matrix calculation called Sparse Linpack.[2]

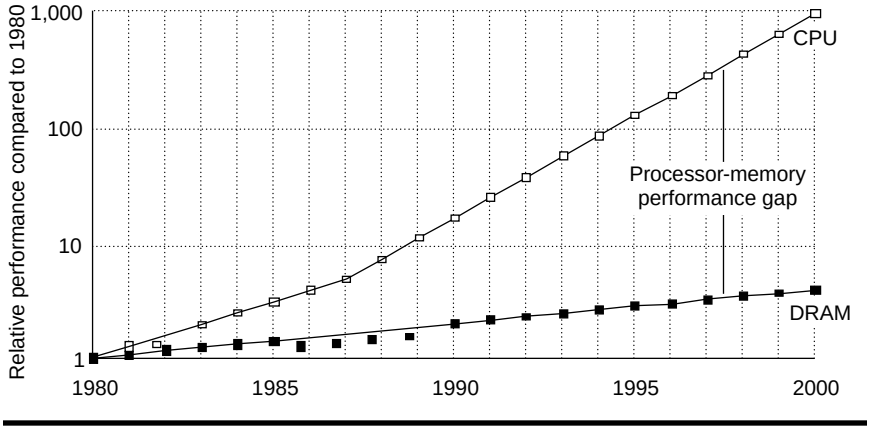

Figure 1. Processor-memory performance gap.[1]

The database and matrix computations spend about 75% of their time in the memory hierarchy. Although the 21164 can execute four instructions per clock cycle for a peak CPI of 0.25, the average CPI for these applications was 3.0 to 3.6. Digital has since begun shipping a 437-MHz version of the same processor with the same external memory system. With a clock almost 50% faster, this processor will spend an even larger fraction of application time waiting for main memory. These extraordinary delays in the memory hierarchy occur despite the tremendous resources dedicated to bridging the processor-memory performance gap.

We call the percent of die area and transistors dedicated to caches and other latency-hiding hardware the memory gap penalty. Table 3 (next page) quantifies the penalty. In several processors, it has grown to 60% of the area and almost 90% of the transistors.[3] In fact, the Pentium Pro offers a package with two die, the larger being the 512-Kbyte second-level cache.

While the processor-memory performance gap has widened to the point where it dominates performance for some applications, the cumulative effect of two decades of 60% per year improvement in DRAM capacity has resulted in huge individual DRAM chips. This has put the DRAM industry in something of a bind. Figure 2 (next page) shows that over time the number of DRAM chips required for a reasonably configured PC has been shrinking. This trend is continuing to the point where soon many PC customers may require only a single DRAM chip.

The required minimum memory size, reflecting application and operating system memory use, has been growing at

Table 1. Latency and bandwidth of the memory system of a 300-MHz Alpha 21164 processor running on an AlphaServer 8200 computer.

Memory module	Size (Kbytes)	Location	Latency ns	Latency Clock cycles	Bandwidth (Mbytes/s)
Instruction cache	8	On chip	6.7	2	4,800
Data cache	8	On chip	6.7	2	4,800
Level-two cache	96	On chip	20.0	6	4,800
Level-three cache	≈4,000	Off chip	26.0	8	960
Main memory subsystem	64,000	Off chip	253.0	76	1,200
Single DRAM component	16,000	Off chip	≈60.0	18	≈30–100

Table 2. CPI, cache misses, and time spent in Alpha 21164 for four programs.

Program	CPI	Misses per 1,000 instructions* I	D	L2	L3	Fraction of time spent in Processor	I cache misses	D cache misses	L2 cache misses	L3 cache misses
SPECint92	1.2	7	25	11	0	0.78	0.03	0.13	0.05	0.00
SPECfp92	1.2	2	47	12	0	0.68	0.01	0.23	0.06	0.02
Database	3.6	97	82	119	13	0.23	0.16	0.14	0.20	0.27
Sparse	3.0	0	38	36	23	0.27	0.00	0.08	0.07	0.58

* I: instruction cache; D: data cache; L2: level-two cache; L3: level-three cache

Table 3. Memory gap penalty for recent microprocessors.

Year	Processor	On-chip cache size (Kbytes) I	D	L2	Memory gap penalty (% die area*)	(% transistors)	Die area (mm^2)††	Transistors (millions)
1994	Digital Alpha 21164	8	8	96	37.4	77.4	298	9.3
1996	Digital StrongARM SA-110	16	16		60.8	94.5	50	2.1
1993	Intel Pentium	8	8		31.9	≈32	≈300	3.1
1995	Intel Pentium Pro	8	8	512	64.2**	87.5†	242 (P) 282 (L2)	5.5 (P) 31.0 (L2)

* Not counting pad ring
** 22.5% of processor +100% of L2 cache = 64.2% total
† 18.2% of processor + 100% of level-two cache = 87.5% total
†† P: processor; L2: level-two cache

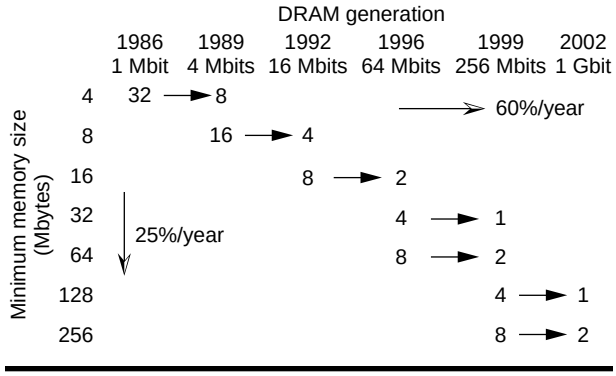

Figure 2. Number of DRAM chips for a minimum memory size PC. Although the desirable minimum memory size is increasing, the capacity per DRAM chip is increasing more quickly.

only about half to three-quarters the rate of DRAM chip capacity. For example, consider a word processor that consumes 8 Mbytes; if its memory needs had increased at the rate of DRAM chip capacity growth, that word processor would have had to fit in 80 Kbytes in 1986 and 800 bytes in 1976.

Despite the decreasing number of required memory chips per PC, a system must have at least enough DRAMs so that their collective width matches the width of the microprocessor's DRAM bus. This width is 64 bits in the Pentium and 256 bits in several RISC machines. Figure 3 shows that each fourfold increase in capacity—the traditional difference between DRAM generations—must be accompanied by a fourfold increase in width to keep the minimum memory size the same.

The minimum memory increment is simply the number of DRAM chips times the capacity of each chip. Figure 4 plots the memory increment with each DRAM generation and DRAM width.

The difficulty is that narrower DRAMs have always provided the lowest cost per bit: A 4-bit-wide part can be, say, 10% cheaper than 16-bit-wide part. Reasons for this difference include a cheaper package, less testing time, and smaller die size. Testing time is shorter because it is a function of both chip width and chip capacity. Die size is smaller because the wider DRAMs have wider buses on chip and require more power and ground pins to accommodate more signal pins. This cost savings makes SIMMs containing narrower parts attractive.

Wide DRAMs are also more awkward to use in systems that provide error correction on data buses. A 64-bit data bus, for example, typically has 8 check bits, meaning that the width of memory is no longer necessarily a power of 2. It might seem that a SIMM could use a new wide part for the data and an older narrow part for the check bits. The problem here is that new high-bandwidth DRAM interfaces—such as synchronous DRAM, which interleaves memory banks internally—do not work well with older DRAMs. In such a world, it is much more efficient to use 8-bit-wide DRAMs than 32-bit-wide DRAMs, as unused memory bits increase the effective cost.

Figures 2 to 4 suggest that users may no longer automatically switch to a new, larger capacity DRAM as soon as it matches the cost per bit of the earlier generation. This is because the new generation's minimum memory increment may be much larger than needed. To avoid waste, larger capacity DRAMs will need wider configurations that are more expensive per bit than the narrow version of the smaller DRAMs. In addition, the wider capacity may not match the width needed for error checking and could hence result in even higher costs.

SIMM packaging isolates users from the DRAMs within the package, so it is likely that they will prefer the cheapest SIMM. This might well have several smaller, narrower DRAMs rather than fewer larger, wider DRAMs. We expect either the 256-Mbit DRAM or 1-Gbit DRAM to see such a critical reception.

Why IRAM is a potential solution

Given the growing processor-memory performance gap and the awkwardness of high-capacity DRAM chips, we believe that it is time to consider unifying logic and DRAM on a single chip. We call such a chip an IRAM, for intelligent RAM, because most transistors on this merged chip will be devoted to memory.

The reason to put the processor in DRAM rather than increasing the on-processor SRAM is that DRAM is in practice approximately 25 to 50 times denser than cache memory in a microprocessor.[4] (The ratio is much larger than the

transistor ratio because DRAMs use 3D structures to shrink cell size). Thus, IRAM would enable a much larger amount of on-chip memory than is possible in a conventional architecture.

Although others have examined this issue in the past, IRAM is attractive today for several reasons. First, the gap between the performance of processors and DRAMs has been widening at 50% per year for 10 years. So, despite heroic efforts by architects, compiler writers, and applications developers, memory speed limits more applications today than it did in the past.

Second, since the actual processor today occupies only about one third of the die (Table 3), the upcoming Gbit DRAM has enough capacity that whole programs and data sets could fit on a single chip. In the past, so little memory could fit on chip with the CPU that researchers considered IRAMs mainly as building blocks for multiprocessors.[5-7]

Third, DRAM die have grown about 50% each generation and are being made with more metal layers to accelerate the longer lines that come with the larger size. Also, DRAM manufacturers are starting to offer fast transistors and many metal layers to make their technology more attractive to IRAM applications. At a cost of, say, 20% more per DRAM wafer, logic in IRAM can be just as fast and dense as a conventional logic process.

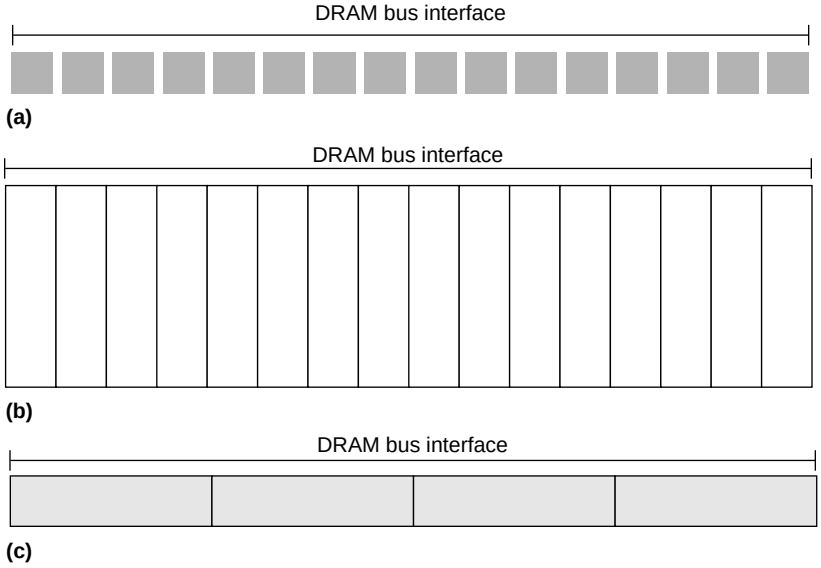

Figure 3. Relationship of microprocessor's DRAM bus width to data width of DRAM chip, minimum number of DRAM chips, and, hence, minimum memory capacity. Each rectangle represents a DRAM chip, with the rectangle's area indicating the capacity and its width indicating the number of data pins. For example, a 4-bit-wide version of the 16-Mbit DRAM requires 16 chips for a 64-bit bus, providing 32 Mbytes (64 bits/4 bits × 16 Mbits/8 bits per byte) of storage (a). A 4-bit-wide version of the 64-Mbit DRAM with the same bus also requires 16 chips, but yields 128 Mbytes (64/4 × 64/8) (b). With the 16-bit-wide version of the 64-Mbit DRAM, the minimum memory increment returns to 32 Mbytes (64/16 × 64/8), because we need just four chips for the 64-bit bus (c).

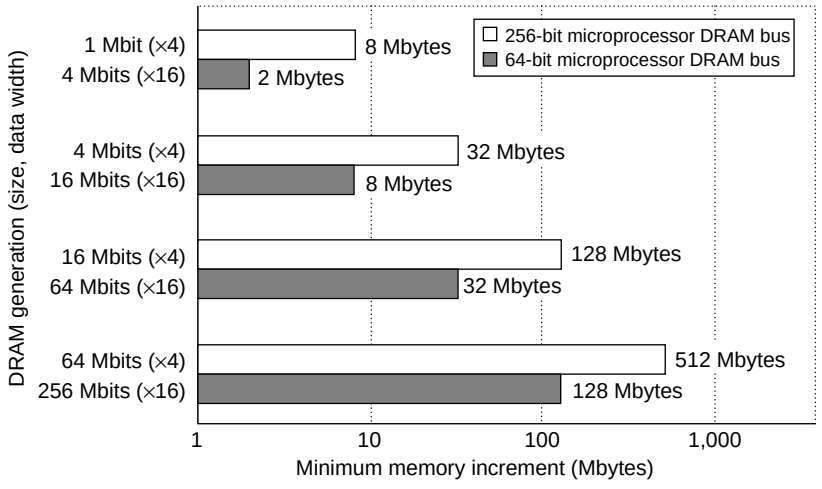

Figure 4. Minimum memory size for two DRAM bus widths as DRAM capacity changes. One bar shows the size for a 64-bit interface, used on the Intel Pentium, and the other is for a 256-bit interface, used by several RISC chips.

Potential advantages

Let's review the potential advantages of IRAM.

Higher bandwidth. A DRAM naturally has extraordinary internal bandwidth, essentially fetching the square root of its capacity each DRAM clock cycle; an on-chip processor could tap that bandwidth.

The potential bandwidth of the Gbit DRAM is even greater than its logical organization indicates. Because it is important to keep the storage cell small, the normal solution is to limit the length of the bit lines, typically with 256 to 512 bits per sense amplifier. This quadruples the number of sense amplifiers. To save die area, each block has a small number of I/O lines, which reduces the internal bandwidth by a factor of about 5 to 10 but still meets the external demand. One IRAM goal is to capture a larger fraction of the potential on-chip bandwidth.

Table 4. Performance factors for estimating IRAM performance in a standard DRAM process.

Microprocessor portion	Optimistic time factor	Pessimistic time factor
Logic	1.3	2.0
SRAM	1.1	1.3
DRAM	0.1	0.2

For example, Mitsubishi and Samsung each presented prototype 1-Gbit DRAMs at the International Solid-State Circuits Conference in 1996. As mentioned earlier, to cope with the long wires inherent in the Gbit DRAMs' 600-mm^2 die, vendors are using more metal layers: Mitsubishi uses three, and Samsung uses four. The Mitsubishi DRAM has a total of 512 two-Mbit memory modules on chip; the Samsung chip has 1,024 one-Mbit modules total.

Thus, a Gbit IRAM might have 1,024 memory modules each 1 Kbit wide. Not only would there be tremendous bandwidth at the sense amps of each block, but the extra metal layers would enable more cross-chip bandwidth. Assuming either multiple buses or a crossbar switch, the internal IRAM bandwidth should be as high as 100 to 200 Gbytes/s.

For comparison, the sustained memory bandwidth of the AlphaServer 8400—which includes a 75-MHz, 256-bit memory bus—is 1.2 Gbytes/s. Recently, several computer architecture researchers have made the case that memory bandwidth will increasingly limit performance;[8-10] IRAM will not have this limit.

Lower latency. To reduce latency, IRAM's wire length should be kept as short as possible. This suggests that the fewer bits per block the better. In addition, the DRAM cells farthest from the processor will be slower than the closest ones. Rather than restricting the access timing to accommodate the worst case, the processor could be designed to access slow and fast memory differently.

Designers could obtain some additional reduction in latency simply by not multiplexing the address—there is no reason to do so on an IRAM. Also, being on the same chip with the DRAM, the processor avoids driving the off-chip wires, potentially turning around the data bus, and accessing an external memory controller. In summary, the access latency of an IRAM processor need not be limited by the same constraints as a standard DRAM part. Designers may obtain much lower latency through intelligent floor planning, using faster circuit topologies, and redesigning the address/data busing schemes.

For a latency-oriented DRAM design on the same chip as the processor, the memory latency for random addresses is potentially less than 30 ns; this is as fast as second-level caches. Recall that the memory latency on the AlphaServer 8400 is 253 ns.

These first two points—about bandwidth and latency—suggest that IRAM offers performance opportunities for two types of applications:

- Those with predictable memory accesses, such as matrix manipulations, may take advantage of the potential 50- to 100-fold increase in IRAM bandwidth.
- Those with unpredictable memory accesses and very large memory footprints, such as databases, may take advantage of the potential 5- to 10-fold decrease in IRAM latency.

Energy efficiency. Integrating a microprocessor and DRAM memory on the same die offers the potential for improving the memory system's energy consumption. As mentioned earlier, DRAM is much denser than SRAM, the traditional choice for on-chip memory. Therefore, an IRAM will have many fewer external memory accesses, which consume a great deal of energy in driving high-capacitance off-chip buses. Even on-chip accesses will be more energy efficient, since DRAM consumes less energy than SRAM. Finally, an IRAM has the potential for higher performance than a conventional approach. Because we can translate higher performance for some fixed energy consumption into equal performance at a lower amount of energy, we can translate IRAM's performance advantages into lower energy consumption.[11]

Memory size and width. Another advantage of IRAM over conventional designs is that it lets designers adjust both the size and width of the on-chip DRAM. Rather than being limited by powers of 2 in length or width, as with conventional DRAM, IRAM designers can specify exactly the number of words and their width. This flexibility can improve the cost of IRAM solutions versus memories made from conventional DRAMs.

Board space. Finally, IRAM may be attractive in applications where board area is precious—such as smart cellular phones or personal digital assistants—since it integrates several chips into one.

Potential disadvantages

Despite these potential advantages, we must address several topics before IRAM can succeed:

- *Area and power impact of increasing bandwidth to the DRAM core.* Standard DRAM cores are designed with a few highly multiplexed I/O lines to reduce area and power. To make effective use of a DRAM core's internal bandwidth, we will need to add more I/O lines. The area increase will affect IRAM's cost per bit.
- *Retention time of DRAM core when operating at high temperatures.* Giacalone[12] gave a rule of thumb of halving the retention rate for every increase of 10 degrees centigrade. Thus, refresh rates could rise dramatically for an IRAM run at the temperature of some of today's microprocessors.
- *Scaling a system beyond a single IRAM.* Even though a Gbit DRAM contains 128 Mbytes, there will certainly be systems that need more memory. Thus, a major architecture challenge is quantifying the pros and cons over several potential solutions to expandable memory.
- *Matching IRAM to the commodity focus of the DRAM industry.* Today's DRAMs are second-sourced commodities that are interchangeable, which allows vendors to manufacture them in high volumes. Unless the

industry were to adopt a single-processor architecture, adding a processor would stratify IRAMs and effectively reduce interchangeability.

- *Testing IRAM.* The cost of testing during manufacturing is significant for DRAMs. Adding a processor could significantly increase the test time on conventional DRAM testers.

Quantifying the potential advantages

The following sections examine three early attempts to quantify what might be done with IRAM technology.[3,11]

An IRAM Alpha

The fastest current microprocessor, the Alpha 21164, was described in sufficient detail[2] to allow us to estimate performance of an IRAM using a similar organization. The Alpha 21164 has three on-chip caches: an 8-Kbyte instruction cache, an 8-Kbyte data cache, and a 96-Kbyte level-two cache. The system we measured included a third-level 4-Mbyte cache off chip. (Table 1 describes the chip and system.) If we were designing an IRAM from scratch, we would surely make different decisions about the memory hierarchy, but a performance estimate of this conventional design will give us a rough idea of IRAM's potential.

Given the estimate of where time is spent from Table 2, the next step is to estimate the performance of each piece if this microprocessor were implemented as an IRAM. Table 4 shows the performance factors by which we multiplied the Alpha performance parameters to estimate the speed of the IRAM implementation.[3] Rather than pick a single number for each category, we picked optimistic and pessimistic factors for implementing in a standard DRAM process.

Guesses for the slowdown of logic in a standard DRAM process range from 1.3 to 2.0; we used these as our optimistic and pessimistic factors for the logic portion of the processor. A common mistake is to assume that everything on the microprocessor would slow down by that factor. As shown in Table 3, however, the vast majority of transistors in recent microprocessors are used for SRAM. Hence, we use a separate factor for the speed of SRAM in a DRAM process—1.1 to 1.3 times slower. Finally, in IRAM the time to main memory should be 5 to 10 times faster than the 253 ns of the Alpha system (Table 1). Hence, we multiply these times by 0.1 to 0.2.

Although some will argue with these values, our purpose is to suggest an approach to estimating performance so that others can supply their own values. We have not seen discussions of SRAM performance separated from logic performance, yet a combined performance estimate may lead to inaccurate predictions.

For our prorated estimate, we first multiply the processor time by the full logic slowdown, despite the possibility that the Alpha clock rate is limited by the speed of hits in the instruction and data caches. Next, since misses from the instruction and data caches go to the slower level-two cache, we multiply those misses by the SRAM factor. Misses from the level-two cache on an IRAM would go to the on-chip DRAM. As we can have a much wider interface on chip, and since we believe a latency-oriented DRAM could have much lower latency, we will assume the misses from the level-two cache in the IRAM will be about the same speed as those in the Alpha. Finally, we believe misses from the level-three cache would be much faster than the 253 ns of the Alpha, so we multiply them by the DRAM factor.

Table 5 shows the optimistic and pessimistic performance factors for an IRAM organized like an Alpha 21164. As expected, the SPEC92 benchmarks are the poorest performers in IRAM because they spend little time in the memory hierarchy. These programs are 1.2 to 1.8 times slower, depending on the IRAM time factors. Note that the database varies from a little slower to a little faster, and that Sparse Linpack varies from 1.2 to 1.8 times faster.

We based this study partly on two benchmarks that rarely miss in the caches. The SPEC92 programs, used in the original Alpha study,[2] are infamous for their light use of the memory hierarchy. Many real programs don't behave like SPEC92.

Although every computer designer wants the fastest hardware in which to design microprocessors, there is a range of performance that is acceptable for the end result. An IRAM Alpha falls within the acceptable range for the highest performance microprocessors. And if the memory gap continues to grow, as we expect, we can expect IRAMs to have even better performance relative to conventional designs.

If performance were the only potential advantage of IRAM, however, these results are *not* sufficient to justify the bold step of producing microprocessors in a DRAM process. Reasons for IRAM would have to include the cost savings of amortizing the fab cost using DRAM chips, lower power, less board space,

Table 5. Performance estimates for the IRAM Alpha.*

	Fraction of time spent in											
	Processor		I cache misses		D cache misses		L2 cache misses		L3 cache misses		Total**	
Program	Opt.	Pess.	Opt.	Pess.	Opt.	Pess.	Opt.	Pess.	Opt.	Pess.	Opt.	Pess.
SPECint92	1.02	1.57	0.04	0.05	0.14	0.17	0.05	0.05	0.00	0.00	1.25	1.83
SPECfp92	0.89	1.36	0.01	0.01	0.26	0.30	0.06	0.06	0.00	0.00	1.21	1.74
Database	0.30	0.46	0.18	0.21	0.15	0.18	0.20	0.20	0.03	0.05	0.85	1.10
Sparse	0.35	0.54	0.00	0.00	0.08	0.10	0.07	0.07	0.06	0.12	0.56	0.82

* Ratio of estimated IRAM times to Alpha times, using the optimistic (opt.) and pessimistic (pess.) parameters from Table 4.
** Overall time, ratio of IRAM to Alpha. A value greater than 1 means IRAM is slower.

Because an IRAM has low latency and is highly interleaved, it naturally matches the needs of a vector processor.

or lower cost due to getting the exact amount of memory on chip.

Keep in mind, however, that we based the performance estimate for this case study on using a conventional microprocessor organization as an IRAM and using a standard DRAM process. This is unlikely to be the best way to exploit a new technology. The next section explores an alternative model that better utilizes the IRAM potential.

An IRAM vector processor

High-speed microprocessors rely on instruction-level parallelism (ILP) in programs, which means the hardware has the potential to execute instructions in parallel. As mentioned earlier, these high-speed microprocessors rely on getting hits in the cache to supply instructions and operands at a sufficient rate to keep the processors busy.

An alternative model to exploiting ILP that does not rely on caches is vector processing. This is a well-established architecture and compiler model popularized by supercomputers, and it is considerably older than the superscalar model. Vector processors have high-level operations that work on linear arrays of numbers.

Advantages of vector computers and the vectorized programs that run on them include the following:

- Each result is independent of previous results, which enables deep pipelines and high clock rates.
- A single vector instruction does a great deal of work, which means fewer instruction fetches in general and fewer branch instructions, and thus fewer mispredicted branches.
- Vector instructions often access memory a block at a time, which allows memory latency to be amortized over, say, 64 elements.
- Vector instructions often access memory with regular (constant-stride) patterns, which allows multiple memory banks to simultaneously supply operands.

These last two advantages mean that vector processors do not rely on data caches for high performance. They rely instead on low-latency main memory, often made from SRAM, using up to 1,024 memory banks to get high memory bandwidth.

In addition to an interleaved, low-latency main memory, vector computers have large register sets, typically 8 to 32 "vector registers," each with about 32 to 128 sixty-four-bit elements. Thus, they have 32 to 256 Kbits of multiported, high-speed registers. Vector processors also depend on multiple, pipelined functional units, as do recent high-speed microprocessors. To match these high-speed functional units to the high-bandwidth memory, vector processors have multiple ports between the processor and memory.

Because an IRAM has low latency and is highly interleaved, it naturally matches the needs of a vector processor. As we head toward hundreds of millions of transistors on a chip, the large register set and multiple, pipelined functional units are also quite plausible. Thus, vectors appear to be one promising way to exploit IRAM.

Moreover, vectors easily allow a trade-off of more hardware and slower clock rate without sacrificing peak performance. For example, a vector processor with a clock rate of n that operates on two elements per clock cycle has the same peak performance as a vector processor with a clock rate of $2n$ that operates on one element per clock cycle.

To achieve multiple instructions per cycle, vector machines use multiple pipelines. For example, the Cray T90 uses two pipes, and the Fujitsu VP300 uses eight. Hence, even if IRAM logic were slower than conventional logic by a factor of two, a IRAM vector processor could have the same peak performance by consuming twice as many elements per clock cycle, trading transistors for clock rate. The observed performance would of course depend on application characteristics.

As shown in Figure 5, an IRAM vector microprocessor might include the following:

- Sixteen 1,024-bit-wide memory ports on the IRAM, offering a collective 100 Gbytes/s of memory bandwidth;
- Thirty-two 64-element vector registers; and
- Pipelined vector units for floating-point add, multiply, and divide; integer operations; load/store; and multimedia operations.

Rather than operating on one element at a time, a eight-pipe vector processor can operate on eight elements in a clock cycle at the cost of multiple vector units.

IRAM could have spectacular vector performance. In a 0.18-micron DRAM process with a 600-mm^2 chip, a high-performance vector accelerator might have eight add-multiply units running at 1,000 MHz and sixteen 1-Kbit buses running at 50 MHz. This combination offers 16 Gflops and 100 Gbytes/s, a balanced vector system. To put this performance in perspective, note that the fastest vector uniprocessor, the Cray T90, achieves a speed of 1.5 Gflops on 1,000×1,000 Linpack. Historically, Cray Research vector processors have doubled in performance every 36 months, so a 16-Gflops IRAM of the Gbit DRAM generation might be faster than a $1-million vector processor of the same era.

The question with vector processing is what fraction of the computation can be accelerated. Classically, scientific programs that deal with matrices have benefited from vector processing. New multimedia and DSP applications may lend themselves to vector processors as well. Indeed, one can characterize the new MMX extension of the 80x86 instruction set as a modest vector extension. Using vector terminology, in 8-bit mode the MMX has eight vector regis-

ters each with eight 8-bit elements, and its functional units each use eight pipes.

We do not intend this study to suggest that IRAM's fate depends on the popularity of vector processing. Rather, we intend to show that IRAM's performance features may lead to trade-offs very different from those in conventional microprocessor design, and that even these initial investigations show promise.

IRAM's energy efficiency

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. While many have examined processor efficiency, our goal is to examine the energy consumed by the memory system.

Power itself can be a deceiving metric, since it does not directly relate to battery life. As an extreme case, putting the processor in sleep mode wins if power is the only metric, yet this solution allows no work to be accomplished. Energy efficiency—expressed either as Joules per instruction or MIPS per Watt—better measures how a given machine best uses limited battery life.

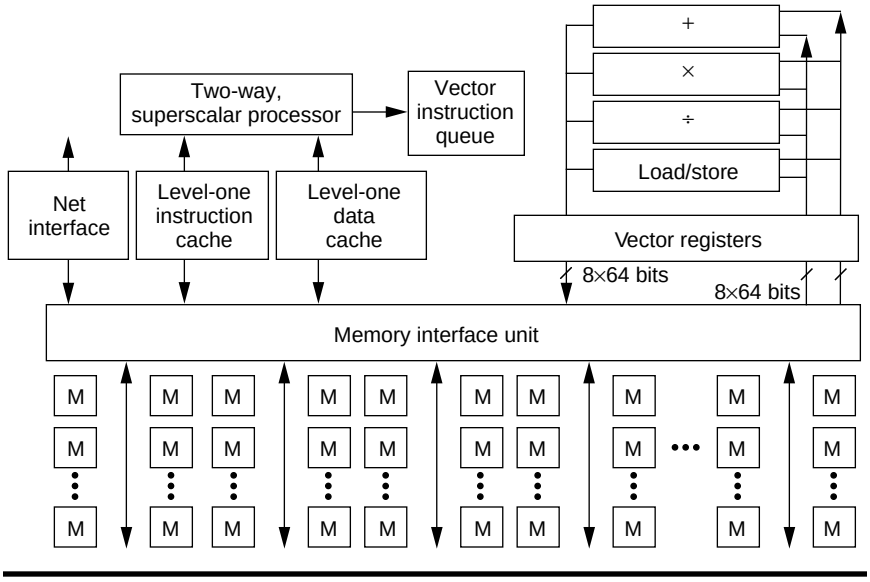

Figure 5. Organization of an IRAM vector processor design.

Table 6. Energy efficiency ratio, IRAM to StrongARM.

	perl	li	gcc	hyfsys	compress
IRAM, 16:1 memory	1.0	1.7	1.8	2.3	1.7
IRAM, 32:1 memory	1.5	2.0	2.5	2.5	4.5

Fromm et al. compared the energy efficiency of the Digital StrongARM memory system, including a 16-Kbyte instruction cache and a 16-Kbyte data cache on chip, to an IRAM memory system.[11] Looking at the number of bits per unit area of the StrongARM caches versus a DRAM of similar technology, we see a difference of almost 50:1. Taking a conservative tack, Fromm et al. compared the StrongARM to an IRAM with memory ratios of 16:1 and 32:1. Table 6 shows IRAM's energy efficiency advantage. Depending on the benchmark, the IRAM advantage is roughly a factor of 2 to 4.

Related work

IRAM may be timely, but it is not a new idea. We have found three categories useful for classifying related work.

Accelerators. Implementations in this category include some logic on chip to make a DRAM run well for a restricted application. Most of the efforts have targeted graphics, where logic is included with memory to be used as the frame buffer. The best known example is video DRAM. Other examples are Mitsubishi's 3D-RAM, which includes a portion of the *z*-buffer logic with 10 Mbits of DRAM to speed up 3D graphics,[13] and Neomagic's graphics accelerator for portable PCs. Nongraphics examples include a level-two cache that uses DRAM to increase size.[12]

Uniprocessors. This category combines a processor with on-chip DRAM. This part might be attractive because of high processor performance, good system power-performance or cost-performance, or combinations of all three.[14,15]

Multiprocessors. This area includes chips intended exclusively as building blocks in a multiprocessor, IRAMs that include a MIMD (multiple instruction, multiple data) multiprocessor within a single chip,[6,16,17] and IRAMs that include a SIMD (single instruction, multiple data) multiprocessor—or array processor—within a single chip.[18,19] This category is the most popular research area for IRAMs.

How they stack up. For uniprocessor and multiprocessor chips, Figure 6 (next page) shows the amount of hardware for memory on the *x*-axis versus the amount of hardware for processing on the *y*-axis. The units of the *x*-axis are bits of storage. It was more difficult to choose the units for the *y*-axis. How do you compare eight 16-bit simple processors each with one 16-bit ALU, one thousand twenty-four 1-bit processors each with a 1-bit ALU, and a 32-bit superscalar processor with two 32-bit ALUs and two 64-bit floating-point units?

Our solution was to multiply the number of each kind of arithmetic unit that can operate in parallel by its width and add these products. This yields the number of bits of parallel arithmetic units. Thus, for the three preceding examples, the number of bits of parallel arithmetic units are: $8 \times 16 = 128$; $1{,}024 \times 1 = 1{,}024$; and $(2 \times 32) + (2 \times 64) = 192$. Figure 6 shows several types of machines; the SIMD research machines have the most processor bits per chip. Note the substantial increase in memory for the Mitsubishi M 32 R/D

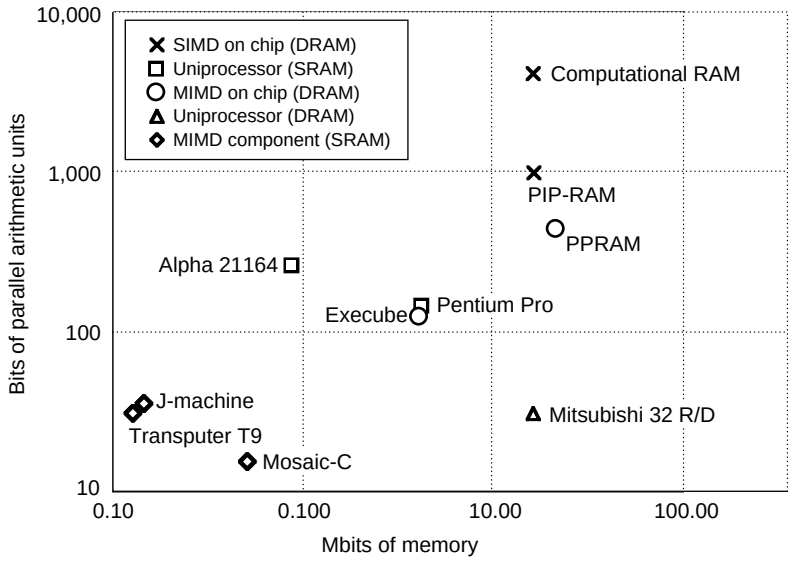

Figure 6. Ordering of IRAMs in a two-dimensional space: bits of arithmetic units versus Mbits of memory.

uniprocessor on a single die that uses DRAM.

This figure suggests two directions for IRAM. One path is up and to the right, using MIMD or SIMD on a chip with a modest amount of memory per arithmetic unit. The other path is gradually rising and to the right, going with a much larger ratio of memory to arithmetic units. This lower path follows the Amdahl rule of thumb, which suggests that for a balanced system, memory capacity increases linearly with processor speed. Although not shown on the graph, the same technology advance that increases memory density also makes logic faster.

In our view, due in part to a lack of sufficient on-chip memory, early IRAM researchers were lured away from making good uniprocessors by the promise of scalable multiprocessing.[5] More recent efforts have targeted multiple processors on a chip to get high peak performance, but have neglected the difficulty of programming such machines, especially when memory is limited.[6,17] New Gbit DRAMs provide sufficient on-chip memory to allow IRAMs with more balanced systems for fast uniprocessors; these are surely easier to program and hence are more widely applicable.

MERGING A MICROPROCESSOR AND DRAM on the same chip presents opportunities in performance, energy efficiency, and cost: a reduction in latency by a factor of 5 to 10, an increase in bandwidth by a factor of 50 to 100, an advantage in energy efficiency of a factor of 2 to 4, and an unquantified cost savings from removing superfluous memory and reducing board area. What is surprising is that these claims are *not* based on some exotic, unproven technology. They are based instead on tapping the potential of a technology in use for the last 20 years.

We believe the popularity of IRAM is limited by the amount of memory on chip, which should expand by about 60% per year. A best-case scenario would be for IRAM to expand its beachhead in graphics—which requires about 10 Mbits of storage—to games, embedded systems, and personal digital assistants, which require about 32 Mbits. Such high-volume applications could in turn justify the investment in a process more friendly to IRAM, with DRAM cells that are a little bigger than those in a DRAM fab but much more amenable to logic and SRAM.

Then, as IRAMs grow to provide 128 to 256 Mbits of storage, the network computer or portable computer markets could adopt this technology. Such a success could in turn entice either microprocessor manufacturers to include substantial DRAM on chip, or DRAM manufacturers to include processors on chip.

IRAM presents an opportunity for us to change the nature of the semiconductor industry. From the current division into logic and memory camps, a more homogeneous industry might emerge. Historical microprocessor manufacturers might ship substantial amounts of DRAM—just as they ship substantial amounts of SRAM today—or historical DRAM manufacturers might ship substantial numbers of microprocessors. It is not clear which camp would ship the most memory or the most microprocessors.

Before such a revolution can occur, however, the field needs more accurate answers to questions such as these:

- What are the speed, area, power, cost, and yield of logic in a DRAM process?
- What are the speed, area, power, and yield of cachelike memory in a DRAM process?
- How does DRAM change if it targets low latency?
- How does DRAM change if it targets large internal bandwidth?
- How do we balance DRAM's requirement for low power to keep refresh rates low with microprocessors' requirement for high power for high performance?
- Can the microprocessor portion of an IRAM have redundant components so as to achieve the same yields that DRAM achieves using redundancy?
- Can built-in-self-test bring down the potentially much higher costs of IRAM testing?
- What is the right way to connect up to 1,000 memory modules to a single CPU on a single-chip IRAM?
- What computer architectures and compiler optimizations turn the high bandwidth of IRAM into high performance?
- What is the right memory hierarchy for an IRAM, and how is that hierarchy managed?
- What is the architectural and operating system solution

for IRAM when applications need more memory than an IRAM provides on chip?
- Given the changes in technology and applications since the early 1980s, when RISC research was developed, is it time to investigate new instruction set architectures?

This list, combined with its potential impact on industry, makes IRAM an exciting research area. In the next decade, the answers to such questions will help determine whether IRAMs will be laboratory novelties or a major industry trend.

Acknowledgments

DARPA (DABT63-C-0056), the California State MICRO Program, and research grants from Intel and Sun Microsystems supported this research. For feedback on earlier versions of this article, we thank Krste Asanović and Steven Przybylski.

References

1. J.L. Hennessy and D.A. Patterson, *Computer Organization and Design,* 2nd ed., Morgan Kaufmann Publishers, San Francisco, 1997.
2. Z. Cvetanovic and D. Bhandarkar, "Performance Characterization of the Alpha 21164 Microprocessor Using TP and SPEC Workloads," *Proc. Second Int'l Symp. High-Performance Computer Architecture,* IEEE Computer Society Press, Los Alamitos, Calif., 1996. pp. 270-280.
3. D. Patterson et al., "Intelligent RAM (IRAM): Chips That Remember and Compute," *Dig. Technical Papers, 1997 IEEE Int'l Solid-State Circuits Conf.,* IEEE, Piscataway, N.J., 1997, pp. 224-225.
4. S.A. Przybylski, *New DRAM Technologies: A Comprehensive Analysis of the New Architectures,* MicroDesign Resources, Sebastopol, California, 1994.
5. I.M. Barron, "The Transputer," *The Microprocessor and Its Application,* D. Aspinall, ed., Cambridge University Press, London, 1978, pp. 343-357.
6. P.M. Kogge et al., "Combined DRAM and Logic Chip for Massively Parallel Systems," *Proc. 16th Conf. Advanced Research in VLSI,* IEEE CS Press, 1995, pp. 4-16.
7. M.D. Noakes, D.A. Wallach, and W.J. Dally, "The J-Machine Multicomputer: An Architectural Evaluation," *Proc. 20th Ann. Int'l Symp. Computer Architecture,* IEEE CS Press, 1993, pp. 224-235.
8. D. Burger, J.R. Goodman, and A. Kagi, "Memory Bandwidth Limitations of Future Microprocessors," *Proc. 23rd Ann. Int'l Symp. Computer Architecture,* IEEE CS Press, 1996, pp. 78-89.
9. S.E. Perl and R.L. Sites, "Studies of Windows NT Performance Using Dynamic Execution Traces," *Proc. Second Symp. Operating Systems Design and Implementation,* 1996, pp. 169-183.
10. W.A. Wulf and S.A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *Computer Architecture News,* Vol. 23, No. 1, Mar. 1995, pp. 20-24.
11. R. Fromm et al., "The Energy Efficiency of IRAM Architectures," submitted to ISCA 97: The 24th Ann. Int'l Symp. Computer Architecture, proceedings to be published by IEEE CS Press, 1997.
12. G. Giacalone et al., "A 1MB, 100MHz Integrated L2 Cache Memory and 128b Interface and ECC Protection," *Proc. Int'l Solid-State Circuits Conf.,* IEEE, 1996, pp. 370-371.
13. M.F. Deering, S.A. Schlapp, and M.G. Lavelle, "FBRAM: A New Form of Memory Optimized for 3D Graphics," *Proc. SIGGRAPH 94,* Assn. for Computing Machinery, New York, 1994, pp. 167-174.
14. T. Shimizu et al., "A Multimedia 32 b RISC Microprocessor with 16 Mb DRAM," *Dig. Technical Papers, 1996 IEEE Int'l Solid-State Circuits Conf.,* IEEE, 1996 pp. 216-217, 448.
15. A. Saulsbury, F. Pong, and A. Nowatzk, "Missing the Memory Wall: The Case for Processor/Memory Integration," *Int'l Symp. Computer Architecture,* IEEE CS Press, 1996, pp. 90-101.
16. M. Fillo et al., "The M-Machine Multicomputer," *Proc. MICRO 95: 28th Ann. IEEE/ACM Int'l Symp. Microarchitecture,* IEEE, 1995, pp. 146-156.
17. K. Murakami, S. Shirakawa, and H. Miyajima, "Parallel Processing RAM Chip with 256Mb DRAM and Quad Processor," *Dig. Technical Papers, 1997 IEEE Int'l Solid-State Circuits Conf.,* IEEE, 1997, pp. 228-229.
18. 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," *Dig. Technical Papers, 1996 IEEE Int'l Solid-State Circuits Conf.,* IEEE, 1996, pp. 372-373, 476.
19. D.G. Elliott, W.M. Snelgrove, and M. Stumm, "Computational RAM: A Memory-SIMD Hybrid and Its Application to DSP," *Proc. Custom Integrated Circuits Conf.,* IEEE, 1992, pp. 30.6.1-30.6.4.

David Patterson holds the Pardee Chair of Computer Science at the University of California, Berkeley, where he teaches computer architecture. He led the design and implementation of RISC I, likely the first VLSI reduced-instruction-set computer. This research became the foundation of the Sparc architecture, currently used by Fujitsu, Sun, and Texas Instruments. He was also a leader of the Redundant Arrays of Inexpensive Disks (RAID) project, which led to many companies' high-performance storage systems. These projects resulted in three distinguished dissertation awards from the Association for Computing Machinery (ACM).

Patterson has won teaching awards from his university, the ACM, and the IEEE. He is also a coauthor of five books and chair of the Computing Research Association. He is a fellow of both the ACM and the IEEE, and a member of the National Academy of Engineering.

Thomas Anderson, who is an associate professor in the Computer Science Division at UC Berkeley, is perhaps best known for his work on scheduler activations and operating system support for high-performance multiprocessing. His other projects include the lightweight remote procedure call (used in the Windows NT RPC system); Digital's AN2, with its novel high-speed network switch architecture (the basis for Digital's ATM products); language-independent software fault isolation (to be shipped next year

as part of Microsoft's Java VM to provide protected execution of non-Java code); xFS, the first serverless network file system; and the Berkeley NOW project.

Anderson has won the Diane S. McIntyre Award for excellence in teaching. He was a program committee member for the ACM ASPLOS Conference in 1996, and a member of NASA EOSDIS. He coauthored award papers at eight recent systems conferences, including the 1994 Hot Interconnects and 1996 Hot Chips symposia.

Neal Cardwell is a graduate student in computer science at UC Berkeley. His current research interests include operating systems, parallel and distributed computing, and compilers. He received a BS in computer science from the College of William and Mary.

Richard Fromm is a member of the IRAM project and is currently working toward a PhD in computer science at UC Berkeley. Previously, he was a member of the Semiconductor Engineering Group at Digital Equipment Corporation, where he contributed to the design of several Alpha microprocessors. His research interests are in computer architecture.

Fromm received a BSEE degree from Cornell University, Ithaca, N.Y., where he cooperated with the Digital Video Group at Bell Communications Research (Bellcore).

Kimberly Keeton is a PhD candidate in computer science at UC Berkeley. Her current research focuses on computer architecture and operating system support for large server applications. Her other research areas have included network performance analysis, multimedia storage servers, and mobile computing.

Keeton received an MS in computer science from UC Berkeley, and a BS in computer engineering from Carnegie Mellon University. She is a member of the IEEE, the ACM, and Usenix.

Christoforos Kozyrakis is currently pursuing a PhD in computer science at UC Berkeley. Previously, he was with ICS-FORTH, Greece, working on the design of single-chip high-speed switches. His research interests include computer architecture, VLSI systems design, and high-performance switching and routing.

Kozyrakis received a BSc degree in computer science from the University of Crete, Greece.

Randi Thomas, a PhD student in electrical engineering and computer science at UC Berkeley, is involved with vector processor research for the IRAM project and is interested in computer architecture. She recently worked in the compiler division of Cray Research in Eagan, Minn., on architectural design for two future SGI/Cray microprocessors.

Thomas received her bachelor's degree in mathematics at the University of California, Berkeley. Several years later she returned to Berkeley in the EECS Reentry Program and began her graduate work there as well. She is currently working on her master's thesis.

Katherine Yelick is a faculty member at UC Berkeley, where she has taught programming languages, parallel systems software, and data structures. Her research in parallel computing addresses irregular applications, data structures, compilers, and runtime systems. Her projects include equational unification algorithms, parallel symbolic applications, the Multipol distributed data structure library, the Split-C parallel language, the Titanium compiler for explicit parallelism, and compiler support for IRAM.

Yelick graduated with a PhD from MIT in 1991, where she worked on parallel programming methods and automatic theorem proving and won the George M. Sprowls Award for an outstanding PhD dissertation.

Address questions concerning this article to David Patterson, Computer Science Division/EECS Department, University of California, Berkeley, CA 94720-1776; patterson@cs.berkeley.edu.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 162 Medium 163 High 164