A Case for Intelligent RAM: IRAM

David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm,
Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas, and Katherine Yelick
Computer Science Division/EECS Department
University of California, Berkeley CA 94720-1776
2/8/97
email: patterson@cs.berkeley.edu

Abstract: Two trends call into question the current practice of microprocessors and DRAMs being fabricated as different chips on different fab lines: 1) the gap between processor and DRAM speed is growing at 50% per year; and 2) the size and organization of memory on a single DRAM chip is becoming awkward to use in a system, 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 as well as to allow more flexible selection of memory size and organization. In addition, IRAM promises savings in power and board area. This paper 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.

1. Introduction and Why there is a Problem

The division of the semiconductor industry into microprocessor and memory camps provides many advantages. First and foremost, a fabrication line can be tailored to the needs of the device. Microprocessor fab lines offer fast transistors to make fast logic and many metal layers to accelerate communication and simplify power distribution, while DRAM fabs 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, allowing microprocessors to use expensive packages that dissipate high power (5 to 50 watts) and provide hundreds of pins to make wide connections to external memory, while allowing DRAMs to use inexpensive packages which dissipate low power (1 watt) and use only a few dozen pins. Separate packages in turn mean computer designers can scale the number of memory chips independent of the number of processors: most desktop systems have 1 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 the Single In-line Memory Module (SIMM) or Dual In-line Memory Module (DIMM), which allows the end user to scale the amount of memory in a system.

Quantitative evidence of the success of the industry is its size: in 1995 DRAMs were a \$37B industry and microprocessors were a \$20B industry. In addition to financial success, the technologies of these industries have improved at unparalleled rates. DRAM capacity has quadrupled on average every 3 years since 1976, while microprocessor speed has done the same since 1986.

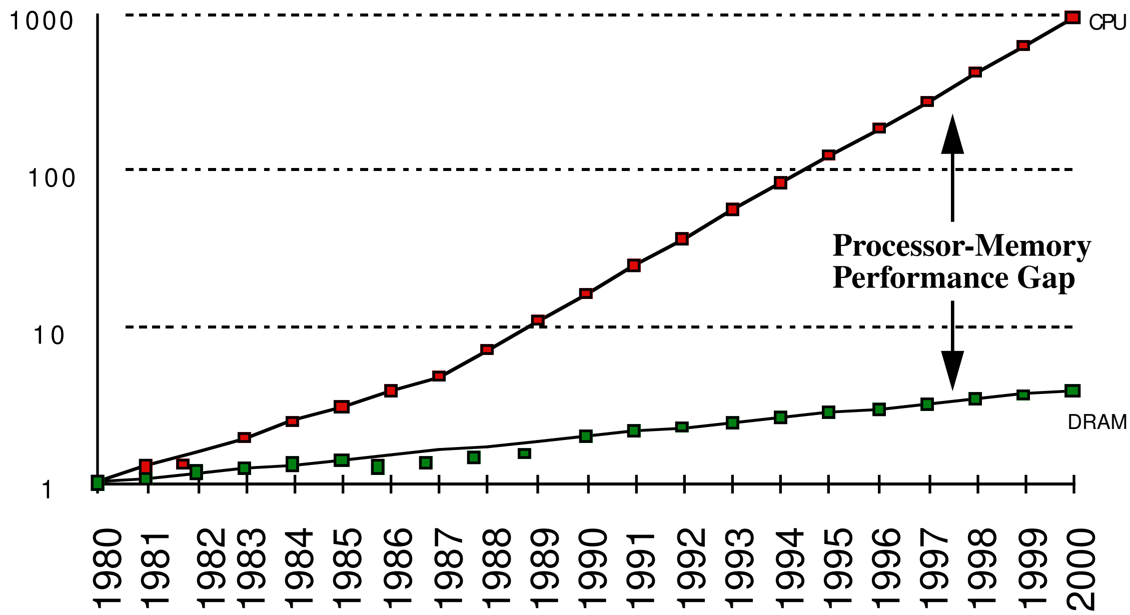

FIGURE 1. Processor-Memory Performance Gap.[Hen96].

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 the processor-memory performance gap by introducing deeper and deeper cache memory hierarchies; unfortunately, this makes the memory latency even longer in the worst case. For example, Table 1 shows CPU and memory performance in a recent high performance computer system. Note that the main memory latency in this system 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 microprocessor, the time to multiplex the addresses to the DRAM, the time to turn around the bidirectional data bus, the overhead of the memory controller, the latency of the SIMM connectors, and the time to drive the DRAM pins first with the address and then with the return data.

Table 1: The latency and bandwidth of the memory system of a high performance computer.

Processor	Alpha 21164	
Machine	AlphaServer 8200	
Clock Rate	300 MHz	
Memory Performance	Latency	Bandwidth
I Cache (8KB on chip)	6.7 ns (2 clocks)	4800 MB/sec
D Cache (8KB on chip)	6.7 ns (2 clocks)	4800 MB/sec
L2 Cache (96KB on chip)	20 ns (6 clocks)	4800 MB/sec
L3 Cache (4MB off chip)	26 ns (8 clocks)	960 MB/sec
Main Memory Subsystem	253 ns (76 clocks)	1200 MB/sec
Single DRAM component	≈60ns (18 clocks)	≈30–100 MB/sec

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

Category	SPECint92	SPECfp92	Database	Sparse
Clocks Per Instruction (CPI)	1.2	1.2	3.6	3.0
I cache misses per 1000 instructions	7	2	97	0
D cache misses per 1000 instructions	25	47	82	38
L2 cache misses per 1000 instructions	11	12	119	36
L3 cache misses per 1000 instructions	0	0	13	23
Fraction of time in processor	0.78	0.68	0.23	0.27
Fraction of time in I cache misses	0.03	0.01	0.16	0.00
Fraction of time in D cache misses	0.13	0.23	0.14	0.08
Fraction of time in L2 cache misses	0.05	0.06	0.20	0.07
Fraction of time in L3 cache misses	0.00	0.02	0.27	0.58

Despite huge on- and off-chip caches and very sophisticated processors with out-of-order, dynamically scheduled superscalar pipelines capable of executing multiple instructions per clock cycle, the long latency and limited bandwidth to main memory dominates performance for many 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 the SPEC92 integer CPU benchmarks, SPEC92 floating point CPU benchmarks, a data base program running a debit-credit benchmark, and a sparse matrix calculation called Sparse Linpack[Cve96]. The database and matrix computations spend about 75% of their time in the memory hierarchy. Although the 21164 is capable of executing 4 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 started shipping a 437 MHz version of the same processor with the same external memory system; with almost a 50% faster clock, an even larger fraction of application time will be spent waiting for main memory.

These extraordinary delays in the memory hierarchy occur despite tremendous resources being spent trying the bridge the processor-memory performance gap. We call the percent of die area and transistors dedicated to caches and other memory latency-hiding hardware the "*Memory Gap Penalty*". Table 3 quantifies the penalty; it has grown to 60% of the area and almost 90% of the transistors in several microprocessors[Pat97]. In fact, the Pentium Pro offers a package with two dies, with the larger die being the 512 KB second level cache.

While the Processor-Memory Performance Gap has widened to the point where it is dominating performance for many 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 shows that over time the number of DRAM chips required for a reasonably configured PC has been shrinking. The required minimum memory size, reflecting application and operating system memory usage, has been growing at only about half to three-quarters the rate of DRAM chip capacity. For example, consider a word processor that requires 8MB; if its memory needs had increased at the rate of DRAM chip capacity growth, that word processor would have had to fit in 80KB in 1986 and 800 bytes in 1976. The result of the prolonged rapid improvement in DRAM capacity is fewer DRAM chips needed per PC, to the point where soon many PC customers may require only a single DRAM chip.

However, at a minimum, a system must have enough DRAMs so that their collective width matches the width of the DRAM bus of the microprocessor. 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.

Table 3: Memory Gap Penalty for several microprocessors.

Year	Processor	On-Chip Cache Size	Memory Gap Penalty: % Die Area (not counting pad ring)	Memory Gap Penalty: Transistors	Die Area (mm^2)	Total Transistors
1994	Digital Alpha 21164	I: 8 KB, D: 8 KB, L2: 96 KB	37.4%	77.4%	298	9.3 M
1996	Digital Strong-Arm SA-110	I: 16 KB, D: 16 KB	60.8%	94.5%	50	2.1 M
1993	Intel Pentium	I: 8 KB, D: 8 KB	31.9%	≈32%	≈300	3.1 M
1995	Intel Pentium Pro	I: 8 KB, D: 8 KB, L2: 512 KB	P: 22.5% +L2: 100% (Total: 64.2%)	P: 18.2% +L2: 100% (Total: 87.5%)	P: 242 +L2:282	P: 5.5 M +L2: 31.0M

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, since testing time is a function of both chip width and chip capacity, and smaller die size, since the wider DRAMs have wider busses 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 busses. 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 like a SIMM module could use a new wide part for the data and an older narrow part for the check bits. The problem 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

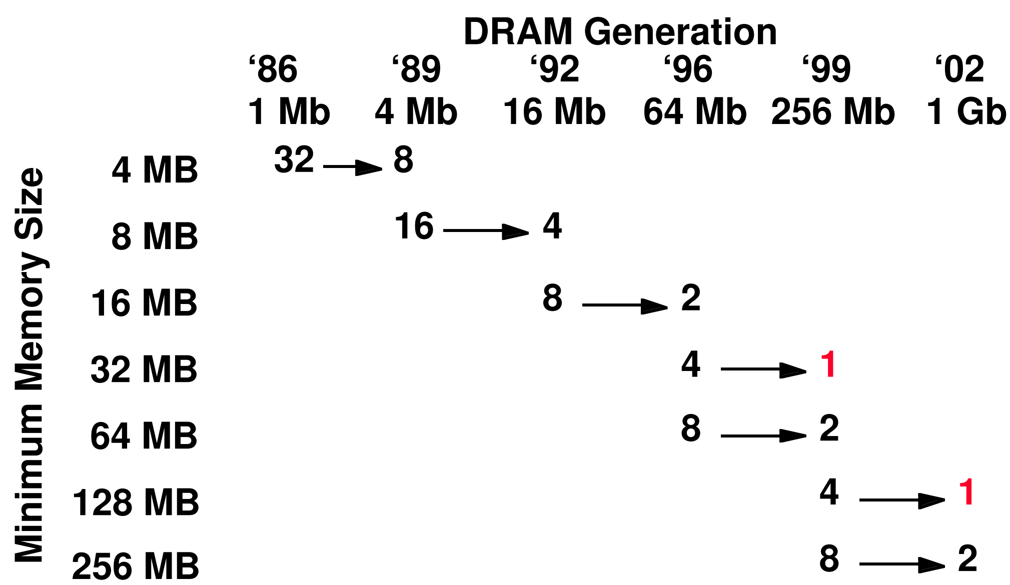

FIGURE 2. Number of DRAMs for a minimum memory size PC over time. Although the desirable minimum memory size is increasing, the capacity per DRAM chip is increasing more quickly.

a world, 8-bit wide DRAMs are much more efficient to use than 32-bit wide DRAMs, as unused memory bits increases the effective cost.

Figures 2 to 4 suggest that customers may no longer automatically switch to the larger capacity DRAM as soon as the next generation matches the same cost per bit in the same organization because 1) the minimum memory increment may be much larger than needed, 2) the larger capacity DRAM will need to be in a wider configuration that is more expensive per bit than the narrow version of the smaller DRAM, or 3) the wider capacity does not match the width needed for error checking and hence results in even higher costs. SIMM packaging isolates the customer from the DRAMs within the SIMM, so it is likely that customers will prefer the cheapest SIMM, which 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.

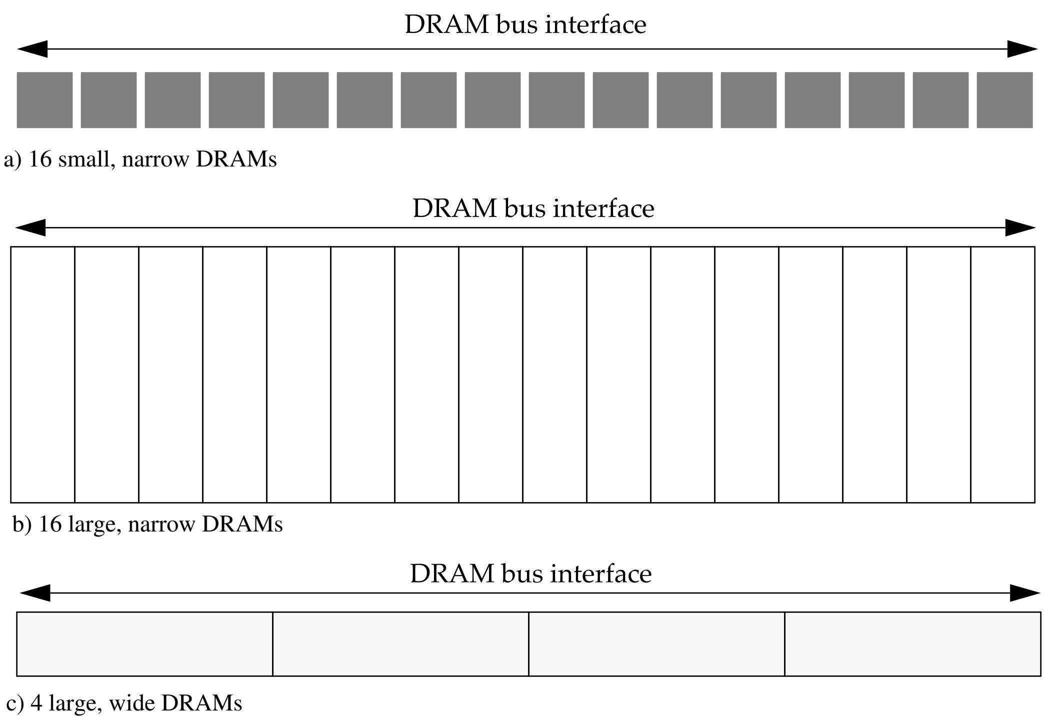

FIGURE 3. Relationship of DRAM bus width on microprocessor versus data width of DRAM chip and minimum number of DRAM chips and hence minimum memory capacity. Each rectangle represents a DRAM chip, with the area of the rectangle indicating the capacity and width indicating the number of data pins. For example, a 4-bit wide version of the 16 Mb DRAM requires 16 chips for a 64-bit bus or 32 MBytes ($64/4 \times 16/8$). Using a 4-bit wide version of the 64 Mb DRAM with the same bus is also 16 chips, yielding 128 MBytes ($64/4 \times 64/8$). Using the 16-bit wide version of the 64 Mb DRAM, the minimum memory increment returns to 32 MBytes($64/16 \times 64/8$), since we need just 4 chips for that bus.

2. IRAM and Why it 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. We call such a chip an “IRAM”, standing for Intelligent RAM, since most of 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 20 times denser than SRAM[Prz94].(The ratio is much larger than the transistor ratio because DRAMs use 3D structures to shrink cell size). Thus, IRAM enables a much larger amount of on-chip memory than is possible in a conventional architecutre.

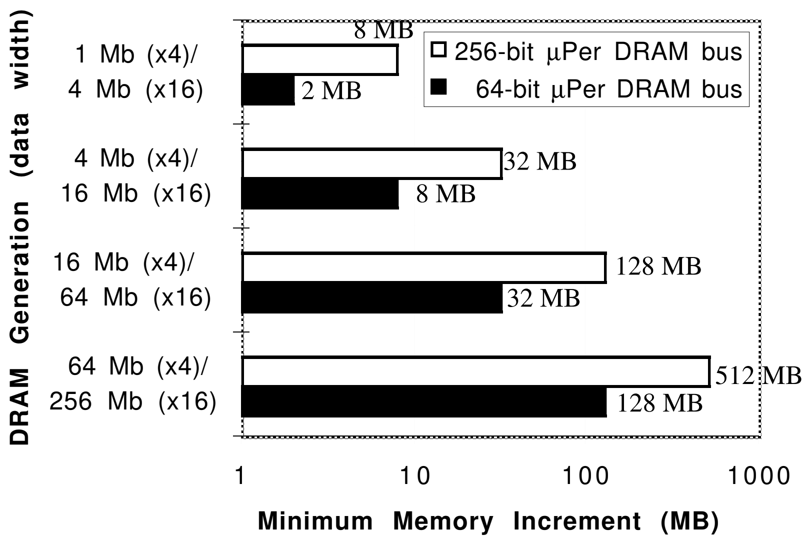

FIGURE 4. A minimum memory size for two DRAM bus widths as DRAM capacity changes. One bar shows the size for a 64-bit interface, used on the Intel Pentium, and the other is a 256-bit interface, used by several RISC chips.

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 that despite heroic efforts by architects, compiler writers, and applications developers, many more applications are limited by memory speed today than in the past. Second, since the actual processor occupies only about one-third of the die (Table 3), the upcoming gigabit DRAM has enough capacity that whole programs and data sets can fit on a single chip. In the past, so little memory could fit on-chip with the CPU that IRAMs were mainly considered as building blocks for multiprocessors [Bar78][Kog95][Noa93]. Third, DRAM dies have grown about 50% each generation; DRAMs are being made with more metal layers to accelerate the longer lines of these larger chips. Also, the high speed interface of synchronous DRAM will require fast transistors on the DRAM chip. These two DRAM trends should make logic on DRAM closer to the speed of logic on logic fabs than in the past.

Potential Advantages of IRAM

1) 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 can tap that bandwidth.The potential bandwidth of the gigabit DRAM is even greater than indicated by its logical organization. Since 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 amp. 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.

For example, two prototype 1 gigabit DRAMs were presented at ISSCC in 1996. As mentioned above, to cope with the long wires inherent in 600 mm^2 dies of the gigabit DRAMs, vendors are using more metal layers: 3 for Mitsubishi and 4 for Samsung. The total number of memory modules on chip is 512 2-Mbit modules and 1024 1-Mbit modules, respectively.

Thus a gigabit IRAM might have 1024 memory modules each 1K bits wide. Not only would there be tremendous bandwidth at the sense amps of each block, the extra metal layers enable more cross-chip bandwidth. Assuming a 1Kbit metal bus needs just 1mm, a 600 mm^2 IRAM might have 16 busses running at 50 to 100 MHz. Thus the internal IRAM bandwidth should be as high as 100-200 GBytes/sec. For comparison, the sustained memory bandwidth of the AlphaServer 8400—which includes a 75 MHz, 256-bit memory bus––is 1.2 Gbytes/sec. Recently, several computer architecture researchers have made the case that memory bandwidth will increasingly limit performance [Bur96] [Per96][Wul95].

2) Lower Latency. To reduce latency, the wire length should be kept as short as possible. This suggests the fewer bits per block the better. In addition, the DRAM cells furthest away 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 be aware when it is accessing “slow” or “fast” memory. Some additional reduction in latency can be obtained simply by not multiplexing the address as 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 does not need to be limited by the same constraints as a standard DRAM part. Much lower latency may be obtained by intelligent floorplanning, utilizing faster circuit topologies, and redesigning the address/data bussing schemes.

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

These first two points suggest IRAM offers performance opportunities for two types of applications:

1. Applications with predictable memory accesses, such as matrix manipulations, may take advantage of the potential 50X to 100X increase in IRAM bandwidth; and
2. Applications with unpredictable memory accesses and very large memory "footprints", such as data bases, may take advantage of the potential 5X to 10X decrease in IRAM latency.

3) Energy Efficiency. Integrating a microprocessor and DRAM memory on the same die offers the potential for improving 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 many 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 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 [Fro97].

4) Memory Size and Width. Another advantage of IRAM over conventional designs is the ability to adjust both the size and width of the on-chip DRAM. Rather than being limited by powers of 2 in length or width, as is 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.

5) Board Space. Finally, IRAM may be attractive in applications where board area is precious—such as cellular phones or portable computers—since it integrates several chips into one.

Potential Disadvantages of IRAM

In addition to these potential advantages, several questions must be answered for IRAM to succeed:

1) Area and speed of logic in a DRAM process: Discussions with experts in circuit design and process have suggested that the area cost might be 30% to 70%, and the speed cost today might be 30% to 100%.

2) Area and power impact of increasing bandwidth to DRAM core: Standard DRAM cores are designed with 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 the cost per bit of IRAM.

3) Retention time of DRAM core when operating at high temperatures: Giacalone [Gia96] gave a rule of thumb of halving the retention rate for every increase of 10 degrees centigrade; thus, refresh rates could rise dramatically if the IRAM is run at the temperature of some microprocessors.

4) Scaling a system beyond a single IRAM: Even though a gigabit DRAM contains 128 Mbytes, there will certainly be systems needing more memory. Thus a major architecture challenge is quantifying the pros and cons over several potential solutions.

5) Matching IRAM to the commodity focus of the DRAM industry: Today's DRAMs are second sourced commodities that are interchangeable, which allows them to be manufactured in high volumes. Unless a single processor architecture was adopted, adding a processor would stratify IRAMs and effectively reduce interchangeability.

6) Testing IRAM: The cost of testing during manufacturing is significant for DRAMs. Adding a processor would significantly increase the test time on conventional DRAM testers.

3. Quantifying the Potential Advantages of IRAM

This section looks at three early attempts to quantify what might be done with IRAM technology [Pat97] [Fro97].

Estimating Performance of an IRAM Alpha

The fastest current microprocessor, the Alpha 21164, was described in sufficient detail [Cve96] to allow us to estimate performance of an IRAM using a similar organization. The Alpha 21164 has three caches on-chip: an 8 KB instruction cache, an 8 KB data cache, and a 96 KB level 2 cache. The system measured included a third level 4 MB cache off chip. Table 1 on page 3 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 suggests the potential of IRAM. Given the estimate of where time is spent from Table 2 on page 3, the next step is to estimate the performance of each piece if this microprocessor was implemented as an IRAM. Table 4 shows the performance factor that we multiply the Alpha performance parameters to estimate the speed of an IRAM[Pat97].Rather than pick a single number of each category, we pick optimistic and pessimistic factors for each piece.

Table 4: Performance Factors for estimating IRAM performance.

Portion of microprocessor	Optimistic Time Factor	Pessimistic Time Factor
Logic	1.3	2.0
SRAM	1.1	1.3
DRAM	0.1	0.2

As mentioned above, guesses for the slowdown of logic in a DRAM process range from 1.3 to 2.0. We use these as our optimistic and pessimistic factors for the logic portion of the processor in Table 4. A common mistake is to assume that everything on the microprocessor would slowdown 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 of 1.1 to 1.3 times slower. Finally, the time to main memory should be 5 to 10 times faster in IRAM than the 253 ns of the Alpha system in Table 1. Hence we multiply the 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 separately from logic performance, yet a combined performance estimate may lead to inaccurate predictions.

Although an IRAM would likely use a different memory hierarchy, for purposes of estimating performance we will use the 21164 organization and adjust performance. We first multiply the processor time by the full logic slowdown, despite the possibility that the clock rate is limited by the speed of hits in the I and D caches. Next, since misses from I and D caches go to the slower L2 cache, we multiply those misses by the SRAM factor. Misses from L2 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 L2 cache will be about the same speed. Finally, we believe misses from L3 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, being 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*.

Table 5: Using optimistic and pessimistic parameters from Table 4 to estimate IRAM performance for four programs.

Category	SPECint92		SPECfp92		Database		Sparse	
	Opt.	Pes.	Opt.	Pes.	Opt.	Pes.	Opt.	Pes.
Fraction of time in processor	1.02	1.57	0.89	1.36	0.30	0.46	0.35	0.54
Fraction of time in I cache misses	0.04	0.05	0.01	0.01	0.18	0.21	0.00	0.00
Fraction of time in D cache misses	0.14	0.17	0.26	0.30	0.15	0.18	0.08	0.10
Fraction of time in L2 cache misses	0.05	0.05	0.06	0.06	0.20	0.20	0.07	0.07
Fraction of time in L3 cache misses	0.00	0.00	0.00	0.00	0.03	0.05	0.06	0.12
Total = ratio of time vs. alpha (>1 means IRAM slower)	1.25	1.83	1.21	1.74	0.85	1.10	0.56	0.82

This study was based in part on two benchmarks that rarely miss in the caches. The SPEC92 programs, used in the original Alpha study [Cve 96], are infamous for their light use of the memory hierarchy; this infamy is one reason they were retired and replaced by the SPEC95 benchmarks. Hence 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 range of acceptable performance for the highest performance microprocessors. And if the memory gap continues to grow, as we expect, then we can expect IRAMs to have even better performance relative to conventional designs.

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

amount of memory on chip.
Keep in mind, however, that the performance estimate for this case study is based on using a conventional microprocessor organization as an IRAM, which is unlikely to be the best way to exploit a new technology. The next section explores one alternative model that better utilizes the potential of IRAM.

Estimating Performance of an IRAM Vector Processor

High speed microprocessors rely on *instruction level parallelism* (ILP) in programs, which means the hardware has the potential short instruction sequences to execute in parallel. As mentioned above, 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*. It is a well established architecture and compiler model that was popularized by supercomputers, and it is considerably older than superscalar. 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:

1. Each result is independent of previous results, which enables deep pipelines and high clock rates.
2. A single vector instruction does a great deal of work, which means fewer instruction fetches in general and fewer branch instructions and so fewer mispredicted branches.
3. Vector instructions often access memory a block at a time, which allows memory latency to be amortized over, say, 64 elements.
4. 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 to have high performance. They rely instead on low latency main memory, often made from SRAM, using up to 1024 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 16 “vector registers” each with about 64 to 128 64-bit elements. Thus they have 32K bits to 128K bits 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.
An IRAM memory has low latency and is highly interleaved, hence an IRAM naturally matches the needs of a vector processor. As we head towards 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 2 elements per clock cycle has the same peak performance as a vector processor with a clock rate of 2N that operates on 1 element per clock cycle. Allowing for multiple instructions per cycle in a vector machine is dones with "multiple pipes." For example, the Cray T90 uses 2 pipes and the Fujitsu VP500 uses 8 pipes. Hence even if IRAM logic were slower by a factor of 2, 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 cousre depend on application characterisics.

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

- Sixteen 1024-bit wide memory ports on the IRAM, offering a collective 100 GB/sec of memory bandwidth.
- Sixteen 128 element vector registers.
- Pipelined vector units for floating point add, multiply, divide, integer operations, load-store, and multimedia operations
- Rather than operate 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 might have spectacular vector performance. In a 0.18 micron DRAM process with a 600 mm^2 chip, a high performance vector accelerator might have 8 Add-Multiply units running at 500 MHz and 16 1-Kbit busses running at 50 MHz. This combination offers 8 GFLOPS and 100 Gbytes/sec, which is a balanced vector system. Putting this performance in perspective, the fastest vector uniprocessor, the Cray T90, achieves a speed of 1.5 GFLOPS on 1000 x 1000 Linpack. Historically, Cray Research vector processors have doubled in performance every 36 months, so an 8 GFLOPS IRAM of the Gbit generation might be faster than $1M vector processor of that era.

The challenge with vector processing is what fraction of the computation can be accelerated by vector processing. Classically, scientific programs that deal with matrices have benefited from vector processing. New multimedia and DSP applications of computers may lend themselves to vector processors as well. Indeed, the new MMX extension of the 80x86 instruction set can be thought of as a modest vector extension. Using vector terminology, MMX has eight vector registers each with eight 8-bit elements and its functional units each use eight pipes.

This study is *not*t intended to suggest the fate of IRAM depends on the popularity of vector processing. It *is* intended to show that IRAM has performance features that may lead to trade-offs that are very different from conventional microprocessor design, and that even these initial investigations show promise.

Estimating Energy Efficiency of IRAM

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 efficiency of the processor, our goal is to

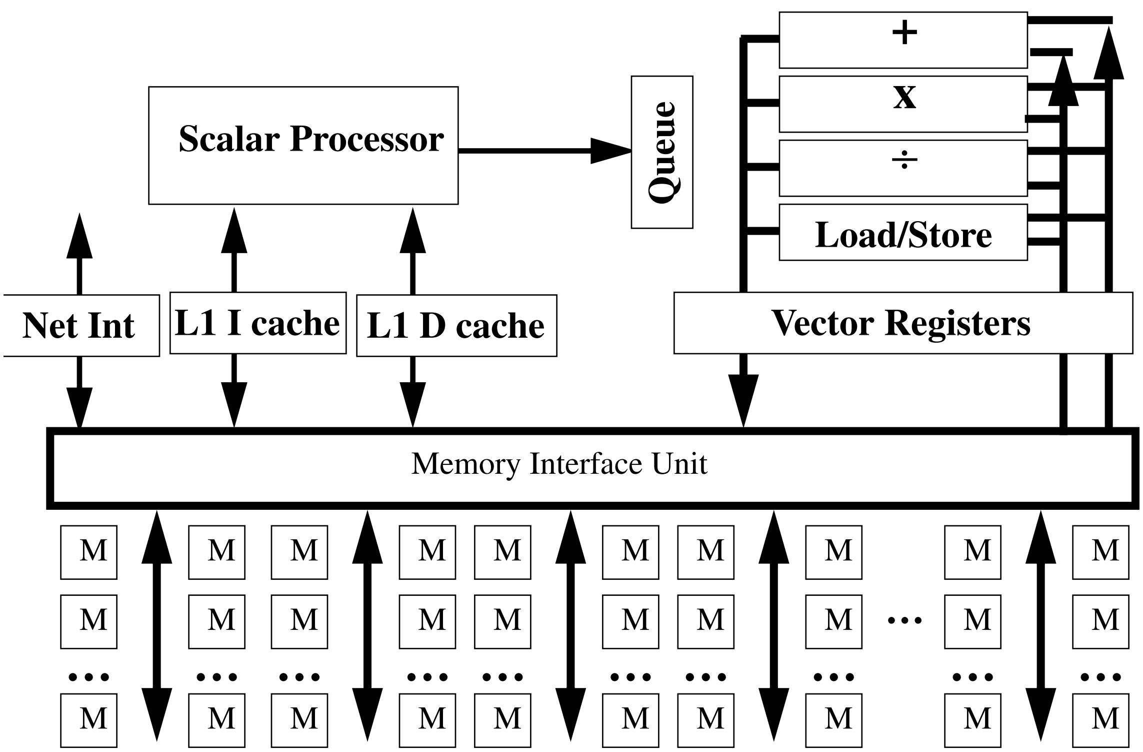

FIGURE 5. Organization of the IRAM design.

examine the energy consumed by the memory system.

Power 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. Hence energy efficiency, expressed either as Joules per Instruction or MIPS per Watt, better measures how a given machine best utilizes limited battery life.

Fromm et al compare the energy efficiency of the Digital StrongARM memory system, including a 16KB instruction cache and a 16 KB data cache on chip, to an IRAM memory system [Fro97]. Looking the number of bits per unit area of the StrongARM caches versus a DRAM of a similar technology we see a difference of almost 50:1, considerably larger than the conventional wisdom of 20:1 between an SRAM and DRAM cell. Taking a conservative tack, they compare the StrongARM to an IRAM with memory ratios of 16:1 and 32:1.The table below shows the energy efficiency advantage of IRAM. Depending on the benchmark, the IRAM advantage is roughly a factor of 2 to 4.

Benchmark	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

4. Related Work

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

1) Accelerators. This category includes some logic on chip to make a DRAM run well for a restricted application. Most of the efforts have been targeted at 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-DRAM, which includes a portion of the Z-buffer logic with 10 Mbits of DRAM to speedup 3D graphics [Dee94], and Neomagic's graphics accelerator for portable PCs. A non-graphics examples include a L2 cache that uses DRAM to increase size [Gia96].

2) Uniprocessors. This category combines a processor with on-chip DRAM. This part might be attractive because of high performance, good power-performance of the system, good cost-performance of the system, or combinations of all three [Shi96] [Sau96].

3) Multiprocessors. This category includes chips intended exclusively to be used as a building block in a multiprocessor, IRAMs that include a MIMD (Multiple Instruction streams, Multiple Data streams) multiprocessor within a single chip [Fil95][Kog95][Mur97], and IRAMs that include a SIMD (Single Instruction stream, Multiple Data streams) multiprocessor, or array processor, within a single chip [Aim96] [Ell92]. This category is the most popular research area for IRAMs.

Figure 6 places uniprocessor and multiprocessor chips on a chart showing the amount hardware for memory on the X-axis versus the amount of hardware for processing on the Y-axis. The units of the X-axis is bits of storage. The Y-axis is harder to choose; how do you compare eight 16-bit simple processors each with one 16-bit ALU, 1024 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 sum the products of each arithmetic unit that can operate in parallel, multiplied by its width, yielding the number of bits of parallel arithmetic units. Thus for the three examples above, the number of bits of parallel arithmetic units are: $8 \times 16 = 128$, $1024 \times 1 = 1024$, and $(2 \times 32) + (2 \times 64) = 196$. The types of machines are listed in the legend. The SIMD research machines have the most processor bits per chip. Note the substantial increase in memory for the Mitsubishi M 32 R/D uniprocessor on a single die which 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 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 memory capacity increases linearly with the *speed* of the processor for a balanced system. Although not shown on the graph, the same technology advance that increases memory density also makes logic faster.

In our view, early IRAM researchers were lured by the promise of scalable multiprocessing, due in part to the lack of sufficient on-chip memory, which distracted them from first making good uniprocessors [Bar78]. 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 [Kog95]. Gigabit DRAM has 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.

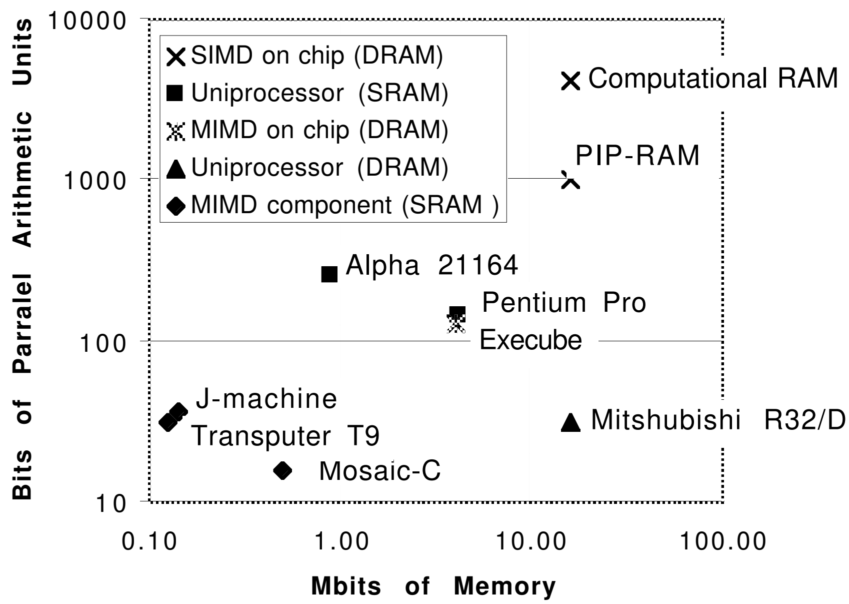

FIGURE 6. Ordering of IRAMs in a two-dimensional space of bits of arithmetic units versus Mbits of memory.

5. Conclusion

Merging a microprocessor and DRAM on the same chip presents opportunities in performance, energy efficiency, and cost: a factor of 5 to 10 reduction in latency, a factor of 50 to 100 increase in bandwidth, a factor of 2 to 4 advantage in energy efficiency, and an unquantified cost savings by removing superfluous memory and by reducing board area. What is surprising is that these claims are *not* based on some exotic, unproven technology; they 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, to the game, embedded, and personal digital assistant markets, which require about 32 Mbits of storage. Such high volume applications could in turn justify creation of a process that is more friendly to IRAM, with DRAM cells that are a little bigger than in a DRAM fab but much more amenable to logic and SRAM. As IRAM grows to 128 to 256 Mbits of storage, an IRAM might be adopted by the network computer or portable PC markets. Such a success could in turn entice either microprocessor manufacturers to include substantial DRAM on chip, or DRAM manufacturers to include processors on chip.

Hence IRAM presents an opportunity to change the nature of the semiconductor industry. From the current division into logic and memory camps, a more homogeneous industry might emerge with historical microprocessor manufacturers shipping substantial amounts of DRAM—just as they ship substantial amounts of SRAM today—or historical DRAM manufacturers shipping substantial numbers of microprocessors. Both scenarios might even occur, with one set of manufacturers oriented towards high performance and the other towards low cost.

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

- What is the speed, area, power, yield of logic in a DRAM process?
- What is the speed, area, power, yield of cache-like memory in a DRAM process?
- How does DRAM change if it is targeted for low latency?
- How does DRAM change if it is targeted for large internal bandwidth?
- How do we balance the desire of DRAM for low power to keep refresh rates low with the desire of microprocessors 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 1000 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 the hierarchy managed?
- What is the architectural and operating system solution for IRAM when applications need more memory than is found on-chip in an IRAM?
- Given the changes in technology and applications since the early 1980s when the RISC research was developed, is it time to investigate new instruction set architectures?

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

6. 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 would also like to thank the following people for giving feedback on earlier versions of this paper: Krste Asanovic and Steven Przybylski.

7. References

[Aim96] Aimoto, Y.; Kimura, T.; Yabe,Y; Heiuchi, H.; and others. “A 7.68 GIPS 3,84 GB/s 1W parallel image processing RAM integrating a 16 Mb DRAM and 128 processors”. *Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 8-10 Feb. 1996, p. 372-73, 476.

[Bar78] Barron, I.M. “The transputer.”*The microprocessor and its application*. Edited by: Aspinall, D. London, UK: Cambridge, 1978. p. 343-57.

[Bur96] Burger, D.; Goodman, J.R.; Kagi, A. ”Memory bandwidth limitations of future microprocessors.” ISCA '96: *The 23rd Annual International Conference on Computer Architecture*,Philadelphia, PA, USA, 22-24 May 1996:78-89.

[Cve96] Cvetanovic, Z.; Bhandarkar, D. “Performance characterization of the Alpha 21164 microprocessor using TP and SPEC workloads.” *Proceedings. Second International Symposium on High-Performance Computer Architecture*, San Jose, CA, USA, 3-7 Feb. 1996. p. 270-80.

[Dal88] Dally, W.J. “Fine-grain message-passing concurrent computers.” IN: *Third Conference on Hypercube Concurrent Computers and Applications*. Pasadena, CA, USA, 19-20 Jan. 1988). Edited by: Fox, G.

[Dee94] Deering, M.F.; Schlapp, S.A.; Lavelle, M.G. “FBRAM: a new form of memory optimized for 3D graphics,” *SIGGRAPH 94 Conference Proceedings*. Orlando, FL, USA, 24-29 July 1994). p. 167-74.

[Ell92] Elliott, D.G.; Snelgrove,W. M.; and Stumm, M. “Computational RAM: A Memory-SIMD Hybrid and its Application to DSP,” In *Custom Integrated Circuits Conference*, Boston, MA, May 1992, pages 30.6.1--30.6.4.

[Fil95] Fillo, M.; Keckler, S.W.; Dally, W.J.; Carter, N.P.; and others. “The M-Machine multicomputer”. *Proceedings of MICRO'95: 28th Annual IEEE/ACM International Symposium on Microarchitecture*, Ann Arbor, MI, USA, 29 Nov.-1 Dec. 1995). p. 146-56.

[Fro97] Fromm, R.; Cardwell, N.; McGaughy, B.; Perissakis, S.; and Patterson, D. “The Energy Efficiency of IRAM Architectures.” submitted to ISCA '97: *The 24th Annual International Conference on Computer Architecture*, Boulder, CO, USA, 1-3 June 1997.

[Gia96] Giacalone, G, *et al*; “A 1MB, 100MHz integrated L2 cache memory and 128b interface and ECC protection,” Proceedings of ISSCC, San Francisco, CA USA, Feb. 1996., p. 370-371.

[Hen96] Hennessy, J.L.; Patterson, D.A. *Computer Organization and Design*, 2nd ed. San Francisco: Morgan Kaufmann Publishers, 1997.

[Kog95] Kogge, P.M.; Sunaga, T.; Miyataka, H.; Kitamura, K.; and others. “Combined DRAM and logic chip for massively parallel systems.” *Proceedings. Sixteenth Conference on Advanced Research in VLSI*, Chapel Hill, NC, USA, 27-29 March 1995, p. 4-16.

[Mur96] Murakami, K. ; Shirakawa, S; and Miyajima, H. “Parallel Processing RAM Chip with 256Mb DRAM and Quad Processor”. *Digest of Technical Papers, 1997 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, Feb. 1997.

[Noa93] Noakes, M.D.; Wallach, D.A.; Dally, W.J. “The J-Machine multicomputer: an architectural evaluation.” *20th Annual International Symposium on Computer Architecture*, San Diego, CA, USA, 16-19 May 1993, p. 224-35.

[Pat97] Patterson, D.; Cardwell, N.; Anderson, T.; Cardwell, N.; Fromm, R.; Keeton, K.; Kozyrakis,K.; Thomas,R.; and Yelick,.K. “Intelligent RAM (IRAM): Chips that remember and compute”. *Digest of Technical Papers, 1997 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, Feb. 1997.

[Per96] Perl, S.E, and Sites, R.L. “Studies of Windows NT performance using dynamic execution traces,” *Second Symposium on Operating Systems Design and Implementation*, Seattle, WA, USA, 29 Oct. - 1 Nov., 1996.

[Prz94] Przybylski, Steven A. *New DRAM Technologies: A Comprehensive Analysis of the New Architectures*, MicroDesign Resources, Sebastopol, California, 1994.

[Sau96] Ashley Saulsbury, Fong Pong, Andreas Nowatzk, “Missing the Memory Wall: The Case for Processor/Memory Integration,” *International Symposium on Computer Architecture*, Philadelphia, PA USA, May 1996.

[Shi96] Shimizu, T.; Korematu, J.; Satou, M.; Kondo, H.; and others. “A multimedia 32 b RISC microprocessor with 16 Mb DRAM.”*Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference*, San Francisco, CA, USA, 8-10 Feb. 1996 p. 216-17, 448.

[Wul95] Wulf, W.A.; McKee, S.A. “Hitting the memory wall: implications of the obvious.” *Computer Architecture News*, March 1995, vol.23, (no.1):20-4.