Theme Feature

Scalable Processors in the Billion-Transistor Era: IRAM

Conventional architectures will not efficiently scale a hundredfold to effectively utilize billion-transistor chips. A more efficient way of using the huge amount of real estate available is to integrate a high-performance processor and the DRAM main memory on the same die, an architecture called intelligent RAM, or IRAM.

Christoforos E. Kozyrakis
Stylianos Perissakis
David Patterson
Thomas Anderson
Krste Asanović
Neal Cardwell
Richard Fromm
Jason Golbus
Benjamin Gribstad
Kimberly Keeton
Randi Thomas
Noah Treuhaft
Katherine Yelick
University of California, Berkeley

The importance of an efficient memory system is increasing as fabrication processes scale down, yielding faster processors and larger memories. This trend widens the processor-memory gap. Not long ago, off-chip main memory was able to supply the CPU with data at an adequate rate. Today, with processor performance increasing at a rate of about 60 percent per year and memory latency improving by just 7 percent per year,[1] it takes dozens of cycles for data to travel between the CPU and main memory.

Designers are investing vast amounts of chip resources to bridge this gap. An increasing fraction of the area budget within microprocessor chips is devoted to static RAM (SRAM) caches. For instance, almost half of the die area in the Digital Alpha 21164 is occupied by caches, used solely for hiding memory latency. This cache memory is just a redundant copy of information that would not be necessary if main memory had kept up with processor speed. Still, some applications show poor locality, resulting in low performance even with large caches.[2]

Other latency tolerance techniques include combining large caches with some form of out-of-order execution and speculation. Yet this is not an efficient solution for the future, as it requires a disproportionate increase in chip area and complexity. Consider, for example, the MIPS R5000 and R10000 processors. The first is a simple RISC processor, while the second is a complex, out-of-order, speculative one. The R10000 takes 3.43 times more area than the R5000, but its performance, as measured by its SPECint95 peak rating, is only 1.64 times higher.

Other architecture alternatives, like wide superscalar and VLIW (very long instruction word), suffer from drawbacks—implementation complexity, low utilization of resources, and immature compiler technology—or deliver only modest performance improvements. Moreover, they usually exacerbate the main memory bandwidth bottleneck.[3]

Beyond the uniprocessor, the possibility exists for the integration of multiple processors on a single die, but this integration would place even greater demands on the memory system. For any given die size, putting more processors on a die will result in less on-chip memory for each, thus increasing the number of slow, off-chip memory accesses. In general, increasing the computing resources without a corresponding increase in the on-chip memory will lead to an unbalanced system. Functional units will often be starved for data because of the high latency and limited bandwidth to and from off-chip memory.

Power dissipation is quickly becoming another major concern in processor architecture as current processors already require extremely high power budgets. The increasing importance of portable electronic systems dictates low-power processor and system designs. It is plausible that when billion-transistor chips become available, the desktop will not be the main focus of the computer industry.

IRAM APPROACH

The intelligent RAM (IRAM) approach to the billion-transistor microprocessor is to use the on-chip real-estate for dynamic RAM (DRAM) memory instead of SRAM caches. It is based on the fact that DRAM can accommodate 30 to 50 times more data than the same chip area devoted to caches.[4] This on-chip memory can be treated as main memory instead of a redundant copy, and in many cases the entire application will fit in the on-chip storage. Having the entire memory on the chip, coupled to the processor through a high bandwidth and low-latency interface, allows for processor designs that demand fast memory systems.

Advantages

IRAM sytems have several potential advantages.[5] The on-chip memory can support high bandwidth and

0018-9162/97/$10.00 © 1997 IEEE

The IRAM approach, combined with a vector architecture, leads to a scalable, high-performance system.

low latency by using a wide interface and eliminating the delay of pads and buses. Energy consumption in the memory system is decreased several times due to the reduction of off-chip accesses through high-capacitance buses.[4] Since the majority of pins in conventional microprocessors are devoted to wide memory interfaces, an IRAM can have a much more streamlined interface. Fewer pins result in a smaller package, and serial interfaces (like FibreChannel and Gigabit Ethernet) that are directly attached to the chip can provide ample I/O bandwidth without being limited by conventional slow I/O buses.[6]

The IRAM approach can be combined with most processor organizations because of the inherent cost advantages of system-level integration. Alas, the first impulse of many computer architects when offered a new technology is simply to build larger caches for conventional architectures. Such designs gain little performance from the on-chip main memory because they were developed with the implicit assumption of a slow memory system that is rarely accessed.[4] Using the IRAM approach creates the potential for superior performance on architectures that can effectively exploit the higher memory bandwidth and lower memory latency.

For any other architecture to be widely accepted, however, it has to be able to run a significant body of software. As the software model becomes more revolutionary, the cost-performance benefit of the architecture must increase for wide acceptance.[1] Given the rapid rate of processor performance improvement and the long time needed for software development, the amount of available code and the simplicity of the programming model are extremely important.

Vector IRAM architecture

An architecture that appears to be a natural match to IRAM because of its bandwidth demands and its well understood programming model is the vector processor. Figure 1 shows the floor plan of the Berkeley IRAM design. In this model the vector processor consists of a vector execution unit combined with a fast in-order scalar core.[7] The combination of a vector unit with a scalar processor creates a general-purpose architecture that can deliver high performance without the issue complexity of superscalar designs or the compiler complexity of VLIW.[8]

Although vector architectures are commonly associated with expensive supercomputers, V-IRAM is a cost-effective system, providing a scalar processor with a vector unit and the memory system on a single die. Vector computers today often use SRAM main memory for low latency and use exotic packaging to provide enough bandwidth to the processor, but a single-chip vector IRAM avoids these costs. The vector unit contains multiple parallel pipelines operating concurrently and vector registers striped across the pipelines, allowing multiple vector elements to be processed in a clock cycle. Increasing the number of pipelines provides a straightforward way to scale performance, as the capacity of integrated circuits increases, without requiring changes to the instruction issue logic or recompilation.

Vector processors have traditionally been used for scientific calculations,[9] but many other applications could benefit from a low-cost vector microprocessor. Emerging applications like multimedia (video, image, and audio processing) are inherently vectorizable: A vector instruction set is the natural way to express concurrent operations on arrays of data, like pixels or audio samples. For example, the Intel MMX extension can be considered a modest vector unit. Many database primitives, like sort, search, and hash-join, have been vectorized, and memory-intensive database applications like decision support and data mining could benefit from IRAM systems with a vector processor.

Even integer applications that are not commonly considered to be vectorizable can often achieve significant speedup through vectorization of their inner loops. For example, the SPECint95 benchmark m88ksim and data decompression achieve speedups of 42 percent and 36 percent respectively through vectorization.[7] In pretty good privacy (PGP) encryption, a vector microprocessor has been shown to significantly outperform an aggressive superscalar processor while occupying less than one-tenth of the die area. In addition to the vector processor, vector IRAM includes a fast scalar processor with small SRAM primary caches, so even nonvectorizable codes will benefit from the fast memory system.

Vector programming provides a simple way to exploit fine-grain data parallelism. Instruction and data dependencies can be efficiently expressed and passed to the hardware.[8] A large amount of research has been invested in vectorizing compilers and in programmer annotations to aid in vectorization, which have been in use by the community for years. In contrast, compilers for VLIW, multithreaded, and MIMD multiprocessors are much more experimental and typically require much more programmer intervention.

Although compiler researchers have looked at vector architectures, surprisingly the computer architecture research community has largely ignored vector architectures while advancing superscalar, VLIW, and multithreaded designs. Hence, innovation at the architecture-compiler interface may allow even more programs to vectorize, making vector IRAM even more attractive.

Because of the simplicity of their circuits, vector processors can operate at higher clock speeds than other architectural alternatives. Simpler logic, higher code density, and the ability to selectively activate the vector and scalar units when necessary also provide higher energy efficiency. Energy efficiency has increased importance in the IRAM context, where it is

necessary to keep the die temperature relatively low to keep the data retention rate at an acceptable level, since empirical data suggest that it has to be doubled for every 10-degree increase in die temperature. Finally, a vector unit with a wide interface to memory can operate as a parallel built-in self-test engine for the memory array, significantly reducing the DRAM testing time and the associated cost.

As the MIPS R5000 demonstrates, a simple scalar processor can have reasonable performance. Moreover, on-chip memory could reduce processor-memory latency by factors of 5 to 10 and increase memory bandwidth by factors of 50 to 200.[5] When such a processor is combined with a vector unit and low-latency, high-bandwidth DRAM memory, it becomes a general-purpose, high-performance, cost-effective, and scalable sytem.

BERKELEY V-IRAM SYSTEM

Figure 1 shows a possible floor plan of a gigabit-generation V-IRAM. A minimum feature size of 0.13 mm and a die of 400 mm^2 will be typical for first-generation production chips—assuming a full-size DRAM die with a quarter of the area dedicated to logic.

The vector unit consists of two load, one store, and two arithmetic units, each with eight 64-bit pipelines running at 1 GHz. Given that clock rates of 600 MHz have already been achieved for complex superscalar microprocessors, a 1-GHz clock rate is a realistic projection. Hence, the peak performance of the V-IRAM implementation is 16 GFLOPS (at 64 bits per operation) or 128 GOPS, when each pipeline is split into multiple 8-bit pipelines for multimedia operations.

The on-chip memory system has a total capacity of 96 Mbytes and is organized as 32 sections, each comprising 16 1.5-Mbit banks and an appropriate crossbar switch. Assuming a pipelined synchronous-DRAM-like interface with 20-ns latency and a 4-ns cycle, the memory system can meet the bandwidth demands of the vector unit at 192 Gbytes per second. The scalar core of V-IRAM is a dual-issue processor with first-level instruction and data caches.

This floor plan indicates another advantage of V-IRAM when trying to use a billion transistors: The design is highly regular with a few unique pieces used repeatedly across the die. Thus the development cost of V-IRAM can be much lower than it would be for conventional designs with a billion transistors.

CHALLENGES TO IRAM

For the IRAM approach to become a mainstream architecture, a number of critical issues need to be resolved. With respect to the fabrication process, the major concerns are the speed of transistors, noise, and overall yield. Current developments in the DRAM industry—such as merged DRAM-logic processes with dual gate oxide thickness and more than two layers of metal—suggest that future DRAM transistors will approach the performance of transistors in logic processes.[10] Noise introduced into the memory array by the fast switching logic can be addressed by using separate power lines and wells for the two system components and by adopting low-power design techniques and architectures. Redundancy in the processor—such as a spare vector pipeline in the case of V-IRAM—can be adopted to compensate for yield reduction due to the addition of logic in the DRAM chip.

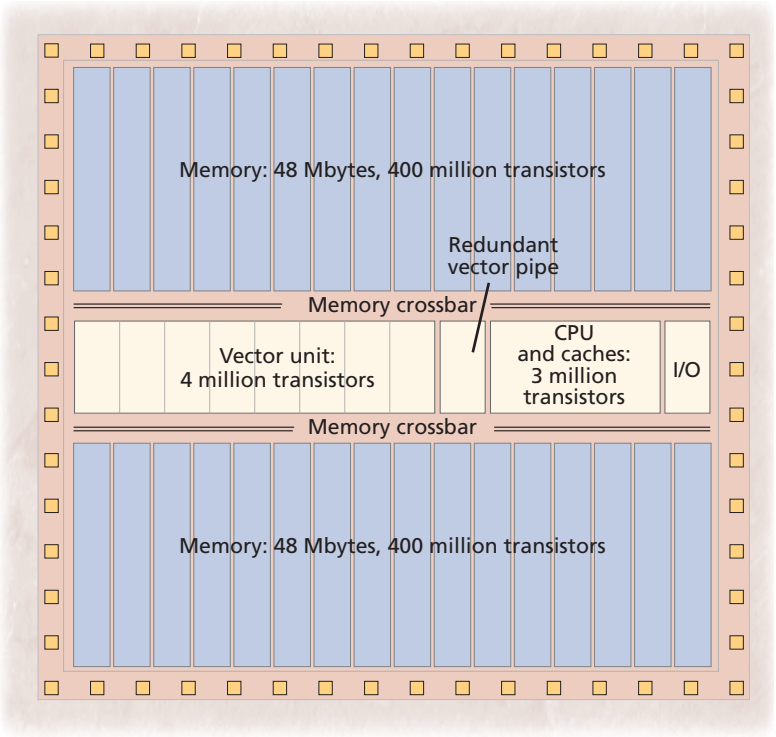

Figure 1. Potential floorplan of the Berkeley V-IRAM.

A more serious architectural consideration is the bounded amount of DRAM that can fit on a single IRAM chip. At the gigabit generation, 96 Mbytes may be sufficient for portable computers, but not for high-end workstations. A potential solution is to back up a single IRAM chip with commodity-external DRAM, using the off-chip memory as secondary storage with pages swapped between on-chip and off-chip memory. Alternatively, multiple IRAMs could be interconnected with a high-speed network to form a parallel computer. Ways to achieve this have already been proposed in the literature.[6,11,12] Fortunately, historical trends indicate that the end-user demand for memory will scale at a lower rate than the available capacity per chip. So, over time a single IRAM chip will be sufficient for increasingly larger systems, from portable and low-end PCs to workstations and even servers.

The vector architecture is not the only option for IRAM systems. If IRAM technology proves successful, there may be even more dramatic integration of processor and memory, distributing portions of processors closer to the individual memory banks. The challenge for such innovation is software migration. Fortunately, the synergy between IRAM and vector processing suggests a high-performance architecture with mature compiler technology, offering a simple and

low-cost solution with software migration for general-purpose systems in the billion-transistor era. ❖

Acknowledgments

This research is supported by the US Defense Advanced Research Projects Agency contract DABT63-C-0056, the California State MICRO Program, Intel, Silicon Graphics/Cray Research, and Sun Microsystems.

References

1. J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach,* 2nd ed., Morgan Kaufmann, San Mateo, Calif., 1996.
2. S.E. Perl and R.L. Sites, "Studies of Windows NT Performance Using Dynamic Execution Traces," *Proc. 2nd Symp. Operating Design and Implementation,* USENIX Assoc., Berkeley, Calif., 1996, pp. 169–183.
3. D. Burger, J.R. Goodman, and A. Kagi, "Memory Bandwidth Limitations of Future Microprocessors," *Proc. 23rd Ann. Int'l Symp. on Computer Architecture*, ACM Press, New York, 1996, pp. 78-89.
4. R. Fromm et al., "The Energy Efficiency of IRAM Architectures," *Proc. 24th Ann. Int'l Symp. Computer Architecture*, 1997, pp. 327–337.
5. D. Patterson et al., "A Case for Intelligent RAM," *IEEE Micro,* Mar./Apr. 1997, pp. 34–44.
6. K. Keeton, R. Arpaci-Dusseau, and D.A. Patterson, "IRAM and SmartSIMM: Overcoming the I/O Bus Bottleneck," http://iram.cs.berkeley.edu/isca97-workshop/ (current July 23, 1997).
7. K. Asanović, *Vector Microprocessors,* doctoral thesis, Univ. of California, Berkeley, Computer Science Division, 1997.
8. J.E. Smith, "The Best Way to Achieve Vector-Like Performance? Use Vectors," keynote talk at 21st Ann. Int'l Symp. Computer Architecture, Apr. 2–11, 1994, http://www.ece.wisc.edu/~jes/pitches/vector.ps (current July 23, 1997).
9. P.M. Johnson, "An Introduction to Vector Processing," *Computer Design,* Feb. 1978, pp. 89–97.
10. M. Nagy et al., "DRAM + Logic Integration: Which Architecture and Fabrication Process?" *IEEE Int'l Solid-State Circuits Conf. Digest of Technical Papers,* IEEE Press, Piscataway, N.J., 1997.
11. D.C. Burger, S. Kaxiras, and J.R. Goodman, "Data-Scalar Architectures," *Proc. 24th Ann. Int'l Symp. Computer Architecture*, ACM Press, May 1997, pp. 338–349.
12. S. Kaxiras, R. Sugumar, and J. Scharzmeier, "Distributed Vector Architecture: Beyond a Single-Vector IRAM," http://iram.cs.berkeley.edu/isca97-workshop/ (current July 23, 1997).

Christoforos E. Kozyrakis *is pursuing a PhD in computer science at the University of California, Berkeley. He received a BS in computer science from the University of Crete, Greece.*

Stylianos Perissakis *is pursuing a PhD in computer science at the University of California, Berkeley. He received a diploma in electrical and computer engineering from the National Technical University of Athens, Greece, and an MS in computer science from the University of California, Berkeley.*

David Patterson *holds the Pardee Chair of Computer Science at the University of California, Berkeley, where he teaches computer architecture. He received his PhD in computer science from the University of California, Los Angeles.*

Thomas Anderson *is an associate professor in the Computer Science Division at the University of California, Berkeley. He received an AB in philosophy from Harvard University and an MS and a PhD in computer science from the University of Washington, Seattle.*

Krste Asanović *holds a postdoctoral position in the Computer Science Division at the University of California, Berkeley. He received a BA in electrical and information sciences tripos from Cambridge University and a PhD from the University of California, Berkeley.*

Neal Cardwell *is pursuing a PhD in computer science at the University of California, Berkeley. He received a BS in computer science from the College of William and Mary.*

Richard Fromm *is pursuing a PhD in computer science at the University of California, Berkeley. He received a BS in electrical engineering from Cornell University.*

Jason Golbus *is pursing an MS in electrical engineering at the University of California, Berkeley. He received a BS in electrical engineering from Duke University.*

Benjamin Gribstad *is pursuing an MS in electrical engineering at the University of California, Berkeley. He received a BS in electrical engineering from the University of Minnesota.*

Kimberly Keeton *is pursing a PhD in computer science at the University of California, Berkeley. She received a BS in computer engineering from Carnegie Mellon and an MS in computer science from the University of California, Berkeley.*

Randi Thomas *is pursing a PhD in computer science at the University of California, Berkeley. She received a BS in mathematics from the University of California, Berkeley.*

Noah Treuhaft *is pursuing a PhD in the computer science division at the University of California, Berkeley. He received a BA in computer science and mathematics from Oberlin College.*

Katherine Yelick *is an associate professor at the University of California, Berkeley. She received a BS, an MS, and a PhD in computer science from MIT.*

For more information on the Berkeley IRAM project, see http://iram.cs.berkeley.edu/. Contact C.E. Kozyrakis at kozyraki@cs.berkeley.edu.