*Christoforos E. Kozyrakis*
*David A. Patterson*
University of California, Berkeley

# A New Direction for Computer Architecture Research

Current computer architecture research continues to have a bias for the past in that it focuses on desktop and server applications. In our view, a different computing domain—*personal mobile computing*—will play a significant role in driving technology in the next decade. This domain will pose a different set of requirements for microprocessors and could redirect the emphasis of computer architecture research.

dvances in integrated circuit technology will soon allow the integration of one billion transistors on a single chip. This is an exciting opportunity for computer architects and designers; their challenge will be to propose microprocessor designs that use this huge transistor budget efficiently and meet the requirements of future applications.

The real question is, just what are these future applications? We contend that current computer architecture research continues to have a bias for the past in that it focuses on desktop and server applications. In our view, a different computing domain—*personal mobile computing*—will play a significant role in driving technology in the next decade. In this paradigm, the basic personal computing and communicating devices will be portable and battery operated, and will support multimedia tasks like speech recognition. These devices will pose a different set of requirements for microprocessors and could redirect the emphasis of computer architecture research.

## BILLION-TRANSISTOR PROCESSORS

*Computer* recently produced a special issue on "Billion-Transistor Architectures."[1] The first three articles discussed problems and trends that will affect future processor design. Seven articles from academic research groups proposed microprocessor architectures and implementations for billion-transistor chips. These proposals covered a wide architecture space, ranging from out-of-order designs to reconfigurable systems. At about the same time, Intel and Hewlett-Packard presented the basic characteristics of their next-generation IA-64 architecture, which is expected to dominate the high-performance processor market within a few years.[2]


0018-9162/98/$10.00 © 1998 IEEE

It is no surprise that most of these proposals focus on the computing domains that have shaped processor architecture for the past decade:

- The uniprocessor desktop running technical and scientific applications, and
- the multiprocessor server used for transaction processing and file-system workloads.

Table 1 summarizes the basic features of the proposed architectures presented in the *Computer* special issue. We also include two other processor implementations not in the special issue, the Trace and IA-64. Developers of these processors have not presented implementation details. Hence we assume that they will have billion-transistor implementations and speculate on the number of transistors they devote to on-chip memory or caches. (For descriptions of these processors, see the "Using a Billion Transistors" sidebar.)

Table 1 reports the number of transistors used for caches and main memory in each billion-transistor processor. The amount varies from almost half the transistor budget to 90 percent. Interestingly, only one design, Raw, uses that budget as part of the main system memory. The majority use 50 to 90 percent of their transistor budget on caches, which help mitigate the high latency and low bandwidth of external memory.

In other words, the conventional vision of future computers spends most of the billion-transistor budget on redundant, local copies of data normally found elsewhere in the system. For applications of the future, is this really our best use of a half-billion transistors?

**Table 1. Number of memory transistors in the billion-transistor microprocessors.**

| Architecture | Key idea | No. of memory transistors (millions) |
|---|---|---|
| Advanced superscalar | Wide-issue superscalar processor with speculative execution; multilevel on-chip caches | 910 |
| Superspeculative | Wide-issue superscalar processor with aggressive data and control speculation; multilevel, on-chip caches | 820 |
| Trace | Multiple distinct cores that speculatively execute program traces; multilevel on-chip caches | 600 |
| Simultaneous multithreading | Wide superscalar with support for aggressive sharing among multiple threads; multilevel on-chip caches | 810 |
| Chip multiprocessor | Symmetric multiprocessor; system shared second-level cache | 450 |
| IA-64 | VLIW architecture with support for predicated execution and long-instruction bundling | 600 |
| Raw | Multiple processing tiles with reconfigurable logic and memory interconnected through a reconfigurable network | 670 |

## Using a Billion Transistors

The first two architectures in *Computer*'s survey—the advanced superscalar and superspeculative—have very similar characteristics. The basic idea is a wide superscalar organization with multiple execution units or functional cores. These architectures use multilevel caching and aggressive prediction of data, control, and even sequences of instructions (traces) to use all the available instruction level parallelism (ILP). Due to their similarity, we group them together and call them *wide superscalar* processors.

The trace processor consists of multiple superscalar processing cores, each executing a trace issued by a shared instruction issue unit. It also employs trace and data prediction, and shared caches.

The simultaneous multithreading (SMT) processor uses multithreading at the granularity of instruction issue slot to maximize the use of a wide-issue, out-of-order superscalar processor. It does so at the cost of additional complexity in the issue and control logic.

The chip multiprocessor (CMP) uses the transistor budget by placing a symmetric multiprocessor on a single die. There will be eight uniprocessors on the chip, all similar to current out-of-order processors. Each uniprocessor will have separate first-level caches but share a large second-level cache and the main memory interface.

IA-64 can be considered a recent commercial reincarnation of architectures based on the very long instruction word (VLIW), now renamed *explicitly parallel instruction computing*. Based on the information announced thus far, its major innovations are the instruction dependence information attached to each long instruction and the support for bundling multiple long instructions. These changes attack the problem of scaling and low code density that often accompanied older VLIW machines. IA-64 also includes hardware checks for hazards and interlocks, which helps to maintain binary compatibility across chip generations. Finally, it supports predicated execution through general-purpose predication registers, which reduces control hazards.

The Raw machine is probably the most revolutionary architecture proposed, as it incorporates reconfigurable logic for general-purpose computing. The processor consists of 128 tiles, each of which has a processing core, small first-level caches backed by a larger amount of dynamic memory (128 Kbytes) used as main memory, and a reconfigurable functional unit. The tiles interconnect in a matrix fashion via a reconfigurable network. This design emphasizes the software infrastructure, compiler, and dynamic event support. This infrastructure handles the partitioning and mapping of programs on the tiles, as well as the configuration selection, data routing, and scheduling.

**Table 2. Evaluation of billion-transistor processors for the desktop/server domain. "Wide superscalar" includes the advanced superscalar and superspeculative processors.**

| Characteristic | Wide superscalar | Trace | Simultaneous multithreading | Chip multiprocessor | IA-64 | Raw |
|---|---|---|---|---|---|---|
| SPECint04 performance | + | + | + | 0 | +/0 | 0 |
| SPECfp04 performance | + | + | + | + | + | 0 |
| TPC-F performance | 0 | 0 | + | + | 0 | – |
| Software effort | + | + | 0 | 0 | 0 | – |
| Physical-design complexity | – | 0 | – | 0 | 0 | + |

## THE DESKTOP/SERVER DOMAIN

In optimizing processors and computer systems for the desktop and server domain, architects often use the popular SPECint95, SPECfp95, and TPC-C/D benchmarks. Since this computing domain will likely remain significant as billion-transistor chips become available, architects will continue to use similar benchmark suites. We playfully call such future benchmark suites "SPECint04" and "SPECfp04" for technical/scientific applications, and "TPC-F" for database workloads.

Table 2 presents our prediction of how these processors will perform on these benchmarks. We use a grading system of + for strength, 0 for neutrality, and – for weakness.

### Desktop

For the desktop environment, the wide superscalar, trace, and simultaneous multithreading processors should deliver the highest performance on SPECint04. These architectures use out-of-order and advanced prediction techniques to exploit most of the available instruction level parallelism (ILP) in a single sequential program. IA-64 will perform slightly worse because very long instruction word (VLIW) compilers are not mature enough to outperform the most advanced hardware ILP techniques—those which exploit runtime information. The chip multiprocessor (CMP) and Raw will have inferior performance since research has shown that desktop applications are not highly parallelizable. CMP will still benefit from the out-of-order features of its cores.

For floating-point applications, on the other hand, parallelism and high memory bandwidth are more important than out-of-order execution; hence the simultaneous multithreading (SMT) processor and CMP will have some additional advantage.

### Server

In the server domain, CMP and SMT will provide the best performance, due to their ability to use coarse-grained parallelism even with a single chip. Wide superscalar, trace, or IA-64 systems will perform worse, because current evidence indicates that out-of-order execution provides only a small benefit to online transaction processing (OLTP) applications.[3] For the Raw architecture, it is difficult to predict any potential success of its software to map the parallelism of databases on reconfigurable logic and software-controlled caches.

### Software effort

For any new architecture to gain wide acceptance, it must run a significant body of software.[4] Thus the effort needed to port existing software or develop new software is very important. In this regard, the wide superscalar, trace, and SMT processors have the edge, since they can run existing executables. The same holds for CMP, but this architecture can deliver the highest performance only if applications are rewritten in a multithreaded or parallel fashion. As the past decade has taught us, parallel programming for high performance is neither easy nor automated.

In the same vein, IA-64 will supposedly run existing executables, but significant performance increases will require enhanced VLIW compilers. The Raw machine relies on the most challenging software development. Apart from the requirements for sophisticated routing, mapping, and runtime-scheduling tools, the Raw processor will need new compilers or libraries to make this reconfigurable design usable.

### Complexity

One last issue is physical design complexity, which includes the effort devoted to the design, verification, and testing of an integrated circuit.

Currently, advanced microprocessor development takes almost four years and a few hundred engineers.[1,5] Testing complexity as well as functional and electrical verification efforts have grown steadily, so that these tasks now account for the majority of the processor development effort.[5] Wide superscalar and multithreading processors exacerbate both problems by using complex techniques—like aggressive data/control prediction, out-of-order execution, and multithreading—and nonmodular designs (that is, they use many individually designed multiple blocks).

With the IA-64 architecture, the basic challenge is the design and verification of the forwarding logic among the multiple functional units on the chip.

Designs for the CMP, trace processor, and Raw machine alleviate the problems of physical-design complexity by being more modular. CMP includes multiple copies of the same uniprocessor. Yet, it carries on the complexity of current out-of-order designs with support for cache coherency and multiprocessor communication. The trace processor uses replicated processing elements to reduce complexity. Still, trace prediction and issue involve intratrace dependence checking, register remapping, and intraelement forwarding. Such features account for a significant portion of the complexity in wide superscalar designs.

Similarly, the Raw design requires the design and replication of a single processing tile and network switch. Verification of a reconfigurable organization is trivial in terms of the circuits, but verification of the mapping software is also required, which is often not trivial.

The conclusion we can draw from Table 2 is that the proposed billion-transistor processors have been optimized for such a computing environment, and that most of them promise impressive performance. The only concern for the future is the design complexity of these architectures.

## A NEW TARGET: PERSONAL MOBILE COMPUTING

In the past few years, technology drivers changed significantly. High-end systems alone used to direct the evolution of computing. Now, low-end systems drive technology, due to their large volume and attendant profits. Within this environment, two important trends have evolved that could change the shape of computing.

The first new trend is that of multimedia applications. Recent improvements in circuit technology and innovations in software development have enabled the use of real-time data types like video, speech, animation, and music. These dynamic data types greatly improve the usability, quality, productivity, and enjoyment of PCs.[6] Functions like 3D graphics, video, and visual imaging are already included in the most popular applications, and it is common knowledge that their influence on computing will only increase:

- 90 percent of desktop cycles will be spent on "media" applications by 2000;[7]
- multimedia workloads will continue to increase in importance;[1] and
- image, handwriting, and speech recognition will pose other major challenges.[5]

The second trend is the growing popularity of portable computing and communication devices.


*Figure 1. Future personal mobile-computing devices will incorporate the functionality of several current portable devices.*

Inexpensive gadgets that are small enough to fit in a pocket—personal digital assistants (PDA), palmtop computers, Web phones, and digital cameras—are joining the ranks of notebook computers, cellular phones, pagers, and video games.[8] Such devices now support a constantly expanding range of functions, and multiple devices are converging into a single unit.

This naturally leads to greater demand for computing power, but at the same time, the size, weight, and power consumption of these devices must remain constant. For example, a typical PDA is five to eight inches by three inches, weighs six to twelve ounces, has two to eight Mbytes of memory (ROM/RAM), and is expected to run on the same set of batteries for a few days to a few weeks.[8] Developers have provided a large software, operating system, and networking infrastructure (wireless modems, infrared communications, and so forth) for such devices. Windows CE and the PalmPilot development environment are prime examples.[8]

### A new application domain

These two trends—multimedia applications and portable electronics—will lead to a new application domain and market in the near future.[9] In the *personal mobile-computing* environment, there will be a single personal computation and communication device, small enough to carry all the time. This device will incorporate the functions of the pager, cellular phone, laptop computer, PDA, digital camera, video game, calculator, and remote shown in Figure 1.

Its most important feature will be the interface and interaction with the user: Voice and image input and

**Most of the billion-transistor architectures offer only limited support for personal mobile computing.**

output (speech and pattern recognition) will be key functions. Consumers will use these devices to take notes, scan documents, and check the surroundings for specific objects.[9] A wireless infrastructure for sporadic connectivity will support services like networking (to the Web and for e-mail), telephony, and global positioning system (GPS) information. The device will be fully functional even in the absence of network connectivity.

Potentially, such devices would be all a person needs to perform tasks ranging from note taking to making an online presentation, and from Web browsing to VCR programming. The numerous uses of such devices and the potentially large volume lead many to expect that this computing domain will soon become at least as significant as desktop computing is today.

## A different type of microprocessor

The microprocessor needed for these personal mobile-computing devices is actually a merged general-purpose processor and digital-signal processor (DSP) with the power budget of the latter. Such microprocessors must meet four major requirements:

- high performance for multimedia functions,
- energy and power efficiency,
- small size, and
- low design complexity.

The basic characteristics of media-centric applications that a processor needs to support were specified by Keith Diefendorff and Pradeep Dubey:[6]

- *Real-time response*. Instead of maximum peak performance, processors must provide worst case guaranteed performance that is sufficient for real-time qualitative perception in applications like video.
- *Continuous-media data types*. Media functions typically involve processing a continuous stream of input (which is discarded once it is too old) and continuously sending the results to a display or speaker. Thus, *temporal locality* in data memory accesses—the assumption behind 15 years of innovation in conventional memory systems—no longer holds. Remarkably, data caches may well be an obstacle to high performance for continuous-media data types. This data is typically narrow (pixel images and sound samples are 8 to 16 bits wide, rather than the 32- or 64-bit data of desktop machines). The capability to perform multiple operations on such data types in a single wide data path is desirable.
- *Fine-grained parallelism*. Functions like image, voice, and signal processing require performing the same operation across sequences of data in a vector or SIMD (single-instruction, multiple-data) fashion.
- *Coarse-grained parallelism*. In many media applications, a pipeline of functions process a single stream of data to produce the end result.
- *High instruction reference locality*. Media functions usually have small kernels or loops that dominate the processing time and demonstrate high temporal and spatial locality for instructions.
- *High memory bandwidth*. Applications like 3D graphics require huge memory bandwidth for large data sets that have limited locality.
- *High network bandwidth*. Streaming data—like video or images from external sources—requires high network and I/O bandwidth.

With a budget of much less than two watts for the whole device, the processor must be designed with a power target of less than one watt. Yet, it must still provide high performance for functions like speech recognition. Power budgets close to those of current high-performance microprocessors (tens of watts) are unacceptable for portable, battery-operated devices. Such devices should be able to execute functions at the minimum possible energy cost.

After energy efficiency and multimedia support, the third main requirement for personal mobile computers is small size and weight. The desktop assumption of several chips for external cache and many more for main memory is infeasible for PDAs; integrated solutions that reduce chip count are highly desirable. A related matter is code size, because PDAs will have limited memory to minimize cost and size. The size of executables is therefore important.

A final concern is design complexity—a concern in the desktop domain as well—and scalability. An architecture should scale efficiently not only in terms of performance but also in terms of physical design. Many designers consider long interconnects for on-chip communication to be a limiting factor for future processors. Long interconnects should therefore be avoided, since only a small region of the chip will be accessible in a single clock cycle.[10]

## Processor evaluation

Table 3 summarizes our evaluation of the billion-transistor architectures with respect to personal mobile computing.

As the table shows, most of these architectures offer only limited support for personal mobile computing.

**Real-time response.** Because they use out-of-order techniques and caches, these processors deliver quite unpredictable performance, which makes it difficult to guarantee real-time response.

**Table 3. Evaluation of billion-transistor processors for the personal mobile-computing domain. "Wide superscalar" includes the advanced superscalar and superspeculative processors.**

| Characteristic | Wide superscalar | Trace | Simultaneous multithreading | Chip multiprocessor | IA-64 | Raw | Comments |
|---|---|---|---|---|---|---|---|
| Real-time response | – | – | 0 | 0 | 0 | 0 | Out-of-order execution, branch prediction, and/or caching techniques make execution unpredictable. |
| Continuous data types | 0 | 0 | 0 | 0 | 0 | 0 | Caches do not efficiently support data streams with little locality. |
| Fine-grained parallelism | 0 | 0 | 0 | 0 | 0 | + | MMX-like extensions are less efficient than full vector support. Reconfigurable logic can use fine-grained parallelism. |
| Coarse-grained parallelism | 0 | 0 | + | + | 0 | + | |
| Code size | 0 | 0 | 0 | 0 | – | 0 | For the first four architectures, the use of loop unrolling and software pipelining may increase code size. IA-64's VLIW instructions and Raw's hardware configuration bits lead to larger code sizes. |
| Memory bandwidth | 0 | 0 | 0 | 0 | 0 | 0 | Cache-based designs interfere with high memory bandwidth for streaming data. |
| Energy/power efficiency | – | – | – | 0 | 0 | – | Out-of-order execution schemes, complex issue logic, forwarding, and reconfigurable logic have power penalties. |
| Physical-design complexity | – | 0 | – | 0 | 0 | + | |
| Design scalability | – | 0 | – | 0 | 0 | 0 | Long wires—for forwarding data or for reconfigurable interconnects—limit scalability. |

**Continuous-media data types.** Hardware-controlled caches complicate support for continuous-media data types.

**Parallelism.** The billion-transistor architectures exploit fine-grained parallelism by using MMX-like multimedia extensions or reconfigurable execution units. Multimedia extensions expose data alignment issues to the software and restrict the number of vector or SIMD elements operated on by each instruction. These two factors limit the usability and scalability of multimedia extensions. Coarse-grained parallelism, on the other hand, is best on the simultaneous multithreading, CMP, and Raw architectures.

**Code size.** Instruction reference locality has traditionally been exploited through large instruction caches. Yet designers of portable systems would prefer reduced code size, as suggested by the 16-bit instruction sets of Mips and ARM. Code size is a weakness for IA-64 and any other architecture that relies heavily on loop unrolling for performance, as such code will surely be larger than that of 32-bit RISC machines. Raw may also have code size problems, as programmers must "program" the reconfigurable portion of each data path. The code size penalty of the other designs will likely depend on how much they exploit loop unrolling and in-line procedures to achieve high performance.

**Memory bandwidth.** Cache-based architectures have limited memory bandwidth. Limited bandwidth becomes a more acute problem in the presence of multi-

**Table 4. Evaluation of vector IRAM for two computing domains.**

| Domain | Characteristics | Rating |
|---|---|---|
| Desktop/server computing | SPECint04 | – |
| | SPECfp04 | + |
| | TPC-F | 0 |
| | Software effort | 0 |
| | Physical-design complexity | 0 |
| Personal mobile computing | Real-time response | + |
| | Continuous data types | + |
| | Fine-grained parallelism | + |
| | Coarse-grained parallelism | 0 |
| | Code size | + |
| | Memory bandwidth | + |
| | Energy/power efficiency | + |
| | Design scalability | 0 |

ple data sequences (with little locality) streaming through a system—exactly the case with most multimedia data.

The potential use of streaming buffers and cache bypassing would help sequential bandwidth, but such techniques fail to address the bandwidth requirements of indexed or random accesses. In addition, it would be embarrassing to rely on cache bypassing for performance when a design dedicates 50 to 90 percent of the transistor budget to caches.

**Energy/power efficiency.** Despite the importance to both portable and desktop domains,[1] most designs do not address the energy/power efficiency issue. Several characteristics of the billion-transistor processors increase the energy consumption of a single task and the power the processor requires.

These characteristics include redundant computation for out-of-order models, complex issue and dependence analysis logic, fetching a large number of instructions for a single loop, forwarding across long wires, and the use of typically power-hungry reconfigurable logic.

**Design scalability.** As for physical design scalability, forwarding results across large chips or communication among multiple cores or tiles is the main problem of most billion-transistor designs. Such communication already requires multiple cycles in several high-performance, out-of-order designs. Simple pipelining of long interconnects is not a sufficient solution as it exposes the timing of forwarding or communication to the scheduling logic or software. It also increases complexity.

The conclusion to draw from Table 3 is that the proposed processors fail to meet many of the requirements of the new computing model. Which begs the question, what design will?

## VECTOR IRAM

Vector IRAM,[1] the architecture proposed by our research group, is our first attempt to design an architecture and implementation that match the requirements of the mobile personal environment.

We base vector IRAM on two main ideas: vector processing and the integration of logic and DRAM on a single chip. The former addresses many demands of multimedia processing, and the latter addresses the energy efficiency, size, and weight demands of portable devices. Vector IRAM is not the last word on computer architecture research for mobile multimedia applications, but we hope it proves a promising initial step.

The vector IRAM processor consists of an in-order, dual-issue superscalar processor with first-level caches, tightly integrated with a vector execution unit that contains eight pipelines. Each pipeline can support parallel operations on multiple media types and DSP functions. The memory system consists of 96 Mbytes of DRAM used as main memory. It is organized in a hierarchical fashion with 16 banks and eight subbanks per bank, connected to the scalar and vector unit through a crossbar. This memory system provides sufficient sequential and random bandwidth even for demanding applications.

External I/O is brought directly to the on-chip memory through high-speed serial lines (instead of parallel buses), operating in the Gbit/s range.

From a programming point of view, vector IRAM is comparable to a vector or SIMD microprocessor.

### Comparison with billion-transistor architectures

We asked reviewers—a group that included most of the architects of the billion-transistor architectures—to grade vector IRAM. Table 4 presents the median grades they gave vector IRAM for the two computing domains—desktop/server and mobile personal computing. We requested reviews, comments, and grades from all the architects of the processors in *Computer*'s September 1997 special issue, and although some were too busy, most were kind enough to respond.

Obviously, vector IRAM is not competitive within the desktop/server domain. Indeed, this weakness in the conventional computing domain is probably the main reason some are skeptical of the importance of merged logic-DRAM technology. As measured by SPECint04, we do not expect vector processing to benefit integer applications. Floating-point intensive applications, on the other hand, are highly vectorizable. All applications would still benefit from the low memory latency and high memory bandwidth.

In the server domain, we expect vector IRAM to perform poorly due to its limited on-chip memory. A potentially different evaluation for the server domain could arise if we examine workloads for decision support

instead of online transaction processing.[11] In decision support, small code loops with highly data-parallel operations dominate execution time. Architectures like vector IRAM and Raw should thus perform significantly better on decision support than on OLTP workloads.

In terms of software effort, vectorizing compilers have been developed and used in commercial environments for decades. Additional work is required to tune such compilers for multimedia workloads and make DSP features and data types accessible through high-level languages. In this case, compiler-delivered MIPS/ watt is the proper figure of merit.

As for design complexity, vector IRAM is a highly modular design. The necessary building blocks are the in-order scalar core, the vector pipeline (which is replicated eight times), and the basic memory array tile. The lack of dependencies within a vector instruction and the in-order paradigm should also reduce the verification effort for vector IRAM.

The open questions are what complications arise from merging high-speed logic with DRAM, and what is the resulting impact on cost, yield, and testing. Many DRAM companies are investing in merged logic-DRAM fabrication lines, and many companies are exploring products in this area. Also, our project sent a nine-million-transistor test chip to fabrication this fall. It will contain several key circuits of vector IRAM in a merged logic-DRAM process. We expect the answer to these questions to be clearer in the next year. Unlike the other billion-transistor proposals, vector IRAM's challenge is the implementation technology rather than the microarchitectural design.

### Support for personal mobile computing

As mentioned earlier, vector IRAM is a good match to the personal mobile-computing model. The design is in-order and does not rely on data caches, making the delivered performance highly predictable—a key quality in supporting real-time applications. The vector model is superior to limited, MMX-like, SIMD extensions for several reasons:

- It provides explicit control of the number of elements each instruction operates on.
- It allows scaling of the number of elements each instruction operates on without changing the instruction set architecture.
- It does not expose data packing and alignment to software, thereby reducing code size and increasing performance.
- A single design database can produce chips of varying cost-performance ratios.

Since most media-processing functions are based on algorithms working on vectors of pixels or samples, it's not surprising that a vector unit can deliver the highest performance. The presence of short vectors in some applications does not pose a performance problem if the start-up time of each vector instruction is pipelined and chaining (the equivalent of forwarding in vector processors) is supported, as we intend.

**Vector IRAM's challenge is in the implementation technology rather than the microarchitectural design.**

The code size of programs written for vector processors is small compared to that of other architectures. This compactness is possible because a single vector instruction can specify whole loops.

Memory bandwidth, both sequential and random, is available from the on-chip hierarchical DRAM.

Vector IRAM includes a number of critical DSP features like high-speed multiply accumulates, which it achieves through instruction chaining. Vector IRAM also provides auto-increment addressing (a special addressing mode often found in DSP chips) through strided vector memory accesses.

We designed vector IRAM to be programmed in high-level languages, unlike most DSP architectures. We did this by avoiding features like circular or bit-reversed addressing and complex instructions that make DSP processors poor compiler targets.[12] Like all general-purpose processors, vector IRAM provides virtual memory support, which DSP processors do not offer.

We expect vector IRAM to have high energy efficiency as well. Each vector instruction specifies a large number of independent operations. Hence, no energy needs to be wasted for fetching and decoding multiple instructions or checking dependencies and making various predictions. In addition, the execution model is strictly in order. Vector processors need only limited forwarding within each pipeline (for chaining) and do not require chaining to occur within a single clock cycle. Hence, designers can keep the control logic simple and power efficient, and eliminate most long wires for forwarding.

Performance comes from multiple vector pipelines working in parallel on the same vector operation, as well as from high-frequency operation. This allows the same performance at lower clock rates—and lower voltages—as long as we add more functional units. In CMOS logic, energy increases with the square of the voltage, so such trade-offs can dramatically improve energy efficiency. DRAM has been traditionally optimized for low power, and the hierarchical structure provides the ability to activate just the sub-banks containing the necessary data.

As for physical-design scalability, the processor-memory crossbar is the only place were vector IRAM uses long wires. Still, the vector model can tolerate latency if sufficient fine-grained parallelism is available. So deep pipelining is a viable solution without any hardware or software complications in this environment.

For almost two decades, architecture research has focused on desktop or server machines. As a result, today's microprocessors are 1,000 times faster for desktop applications. Nevertheless, in doing so, we are designing processors of the future with a heavy bias toward the past. To design successful processor architectures for the future, we first need to explore future applications and match their requirements in a scalable, cost-effective way.

Personal mobile computing offers a vision of the future with a much richer and more exciting set of architecture research challenges than extrapolations of the current desktop architectures and benchmarks. Vector IRAM is an initial approach in this direction.

Put another way, which problem would you rather work on: improving performance of PCs running FPPPP—a 1982 Fortran benchmark used in SPECfp95—or making speech input practical for PDAs? It is time for some of us in the very successful computer architecture community to investigate architectures with a heavy bias for the future. ❖


### Acknowledgments

The ideas and opinions presented here are from discussions within the IRAM group at UC Berkeley. In addition, we thank the following for their useful feedback, comments, and criticism on earlier drafts, as well as the grades for vector IRAM: Anant Agarwal, Jean-Loup Baer, Gordon Bell, Pradeep Dubey, Lance Hammond, Wang Wen-Hann, John Hennessy, Mark Hill, John Kubiatowicz, Corinna Lee, Henry Levy, Doug Matzke, Kunle Olukotun, Jim Smith, and Gurindar Sohi.

This research is supported by DARPA (DABT63-C-0056), the California State MICRO program, NSF (CDA-9401156), and by research grants from LG Semicon, Hitachi, Intel, Microsoft, Sandcraft, SGI/Cray, Sun Microsystems, Texas Instruments, and TSMC.


### References


1. D. Burger and J. Goodman, "Billion-Transistor Architectures," *Computer*, Sept. 1997, pp. 46-47.
2. J. Crawford and J. Huck, "Motivations and Design Approach for the IA-64 64-bit Instruction Set Architecture," *Microprocessor Forum*, Micro Design Resources, 1997.
3. K. Keeton et al., "Performance Characterization of the Quad Pentium Pro SMP Using OLTP Workloads," *Proc. 1998 Int'l Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 15-26.
4. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach, 2nd ed.*, Morgan Kaufmann, San Francisco, 1996.
5. J. Wilson et al., "Challenges and Trends in Processor Design," *Computer*, Jan. 1998, pp. 39-50.
6. K. Diefendorff and P. Dubey, "How Multimedia Workloads Will Change Processor Design," *Computer*, Sept. 1997, pp. 43-45.
7. W. Dally, "Tomorrow's Computing Engines," keynote speech, *Fourth Int'l Symp. High-Performance Computer Architecture*, Feb. 1998.
8. T. Lewis, "Information Appliances: Gadget Netopia," *Computer*, Jan. 1998, pp. 59-66.
9. G. Bell and J. Gray, *Beyond Calculation: The Next 50 Years of Computing*, Springer-Verlag, Feb. 1997.
10. D. Matzke, "Will Physical Scalability Sabotage Performance Gains?" *Computer*, Sept. 1997, pp. 37-39.
11. P. Trancoso et al., "The Memory Performance of DSS Commercial Workloads in Shared-Memory Multiprocessors," *Proc. Third Int'l Symp. High-Performance Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 250-260.
12. J. Eyre and J. Bier, "DSP Processors Hit the Mainstream," *Computer*, Aug. 1998, pp. 51-59.



*Christoforos E. Kozyrakis is currently pursuing a PhD degree in computer science at the University of California, Berkeley. Prior to that, he was with ICS-FORTH, Greece, working on the design of single-chip high-speed network routers. His research interests include microprocessor architecture and design, memory hierarchies, and digital VLSI systems. Kozyrakis received a BSc degree in computer science from the University of Crete, Greece. He is a member of the IEEE and the ACM.*

*David A. Patterson teaches computer architecture at the University of California, Berkeley, and holds the Pardee Chair of Computer Science. At Berkeley, he led the design and implementation of RISC I, likely the first VLSI reduced instruction set computer. He was also a leader of the Redundant Arrays of Inexpensive Disks (RAID) project, which led to the high-performance storage systems produced by many companies. Patterson received a PhD in computer science from the University of California, Los Angeles. He is a fellow of the IEEE Computer Society and the ACM, and is a member of the National Academy of Engineering.*

*Contact the authors at the Computer Science Division, University of California, Berkeley, CA 94720-1776; {kozyraki, patterson}@cs.berkeley.edu.*