Architects Look to Processors of Future

Applications, Instruction Sets, Memory Bandwidth Are Key Issues

For this special issue, we asked several processor architects how, based on 25 years of history, they see the microprocessor continuing to evolve in the future. Their responses discuss several technical barriers to success and how they might be overcome. Equally important is an often overlooked issue: what will people do with all this performance?

Gordon Bell

Many New Applications Will Emerge

In 1947, the big idea (perhaps of all time) was the stored program computer that was soon to operate. In the same year, the transistor, a second and equally big idea, was invented. By the mid 1960s, a way of fabricating and interconnecting transistors on silicon substrates was invented and in use.

Photo of Gordon Bell

The development of the microprocessor in 1971 ensured the evolution of computing would continue in a very focused fashion. The next 15–25 years look equally bright. The only form of intelligence more easily, cheaply, and rapidly fabricated is the human brain, estimated to have a processing power of around 1,000 million million ops/s (one petaops), with a memory of 10 terabytes [Cochrane, 1996].

For five decades, hardware has stimulated the evolution of computer platforms of various performance, size, cost, form, and applications, from watches and pacemakers to mainframes. It is safe to predict computers in 2047 will be at least 100,000 times more powerful. If hardware continues to evolve at the annual factor of 1.6 we know as Moore's Law [Moore, 1996], computers that are 10 billion times more powerful will exist! Magnetic-storage density and fiber-optic data transmission rates have evolved at the 60% rate (a doubling every 18 months, or 100 times per decade), too.

It is also likely that, since improvements in algorithms and methods often occur at the same rate as in hardware, any future goal is likely to be reached in half the time one would predict based on hardware alone. I don't believe the homely computer, built as a simple processor/memory structure, will take on a very different look, but rather will continue on an evolutionary path of only slightly more parallelism of instruction execution. For the past decade, real application performance (RAP) of microprocessors has diverged from the peak announced performance (PAP) that follows Moore's Law. This trend will continue!

Figure 1 shows past hardware evolution and a 50-year forecast of the future. The next 15 years, based on semiconductor progress, are most likely to follow this trend. After that time, the figure shows a diverging range of possibilities.

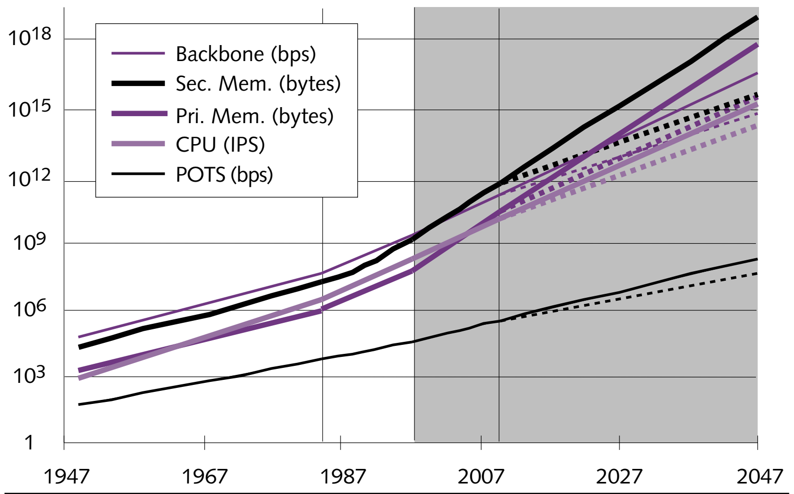

Figure 1. Aggressive projections (solid lines) show processing power, memory, and bandwidth increasing rapidly throughout the next 50 years. Even in the conservative case (dashed lines), there will be enormous improvements in the future. (Source: Gordon Bell, 1996)

What Forms Will the Future Computer Take?

All intellectual property and everything bitable will be in cyberspace. With cyberspace, the speed limit is our ability to find new places. Bitability comes from the hardware and software interfaces (I/O) that the computer has acquired, created, or evolved to allow it to communicate with people and the physical world. We eventually expect speech, video, and gesture interfaces, followed by having computers that anticipate. Surely, we can expect a "do what I say" metaphor within a decade, since it has been a dream for so long.

Direct body interfaces are increasingly important, including touch, direct nerve stimulus, and artificial organs, eyes, ears, and limbs. I don't expect computers will interface by taste and smell. For achieving the mobility and navigation in the physical world that would enable useful robots, the big inventions already exist today as demonstrations, with video recognition, global-positioning systems (GPS), laser sensing, single-chip phased-array radar, and sonar. They have to evolve to low-cost components and become fast enough. By 2047, I expect homes, commercial areas, and factories will have useful robots that do not require extensive training.

New computer classes based on price will continue to be determined by applications and their resulting markets together with three factors: hardware platform technology (e.g., semiconductors, magnetics, and displays); hardware/software interfaces to connect with the physical world, including people; and network infrastructures (e.g., the Internet and eventually home and body area networks).

My theory of computer class formation, based solely on using lower-cost components and different forms of use to stimulate new structures, accounted for the emerging of minicomputers (1970s), workstations and personal computers (1980s), and personal organizers. The World Wide Web has stimulated other computer classes to emerge, including network computers, telecomputers, and television computers that are combined with phones and television sets, respectively. As Table 1 shows, this basic theory also accounts for the emergence of embedded and low-cost game computers using worldwide consumer distribution networks. Mobility via a radio network opens up more future possibilities that are not just adaptations of cellular phones.

Generation	Platform (logic, memories, O/S)	User Interface	Network Infrastructure
The beginning (direct and batch use)	Vacuum tube, transistor, core, drum and mag tape	Card, paper tape	None originally; computer was self-contained
Interactive timesharing via commands	Integrated circuit, disk, multiprogramming	Glass, teletype and keypunch, command language	POTS using modem; proprietary nets using WAN
Distributed PCs and workstations	The microprocessor. PCs and workstations, floppy, disk, distributed O/S	WIMP (windows, icons, mouse, pull-down menus)	WAN, LAN
World Wide Web	Evolutionary PCs and workstations, servers everywhere, Web O/S	Browser	Fiber optics backbone, WWW, HTTP
SNAP (Scalable Network and Platforms)	PC uni- or multiprocessor commodity platform	Server provisioning	SAN (System Area Network) for clusters
One dial tone: phone, videophone, TV and data	Network computer, telecomputer, TV computer	Telephone, videophone, television	xDSL for POTS, cable, fiber (longer term); home area nets
Do what I say	Embedded in PCs, hand-held devices, phone, PDA	Speech, common sense	Body area nets. IR and radio LANs for network access
Anticipatory by observing user needs	Room monitoring, gesture	Vision, gesture control, common sense	Home area nets
Robots	No special	Radar, sonar, vision, mobility, arms, hands	IR and radio LAN
Ubiquity embedded	$1–$100 devices that interoperate	Computer-to-computer control	Home area and body area networks

Table 1. New computer classes and their enabling components. (Source: Gordon Bell)

Within a few years, scalable computing, using an arbitrary number of commodity-priced computers and commodity high-speed networks to operate as one, is likely to replace traditional computers, i.e., servers of all types! We call this approach to computing SNAP, for scalable network and platforms [Gray, 1996]. The underlying parallelism is a challenge that has escaped computer science for decades.

As communication instruments, computers enable the substitution of time and place of work, creating a flat, equal-access world [CNRI, 1996]. After nearly 30 years of the Internet, people-to people communication via e-mail and chat remains the top application. Is telepresence for work, learning, and entertainment the long-term "killer app"?

Can these systems be built in this short time? Will computers interface with humans biologically, rather than in the superficial, mechanical way they do now? More likely, nearly-zero-cost communicating computers will be everywhere, embedded in everything from phones and light switches to all-seeing, all-changing pictures. They'll be the eyes and ears for the blind and deaf, and they will eventually drive vehicles.

We will need to be fully connected anywhere at all times. The big idea is fiber-optic cable that evolves to carry more bits per second each year at 1.6× per year. Perhaps an equally big idea is in the making: the high-speed digital subscriber link, a.k.a. "the last mile," that permits high-speed data to go to the home via the world's trillion dollars of installed copper connections. In parallel, radio links will enable "anywhere computing." Body and home area networks are parts of the network story that need to be invented.

As VP of R&D at Digital for 23 years, Gordon Bell led the development of the PDP and VAX minicomputers and other products. He is now a senior researcher at Microsoft and can be reached at gbell@microsoft.com.

References

CNRI. "Vision of the NII: Ten Scenarios," Reston, Virginia, 1996. See also *www.cnri.reston.va.us.*

Cochrane, Peter. Many papers on the future of computers; *www.labs.bt.com/people/cochrap.*

Gray, J. "Scalable Servers"; *www.research.com/research/barc.*

Moore, Gordon. "Gigabits and Gigabucks," University Video Corp. Distinguished Lecture, 1996; *www.uvc.com.*

Richard Sites

It's the Memory, Stupid!

When we started the Alpha architecture design in 1988, we estimated a 25-year lifetime and a relatively modest 32% per year compounded performance improvement of implementations over that lifetime (1,000× total). We guestimated about 10× would come from CPU clock improvement, 10× from multiple instruction issue, and 10× from multiple processors.

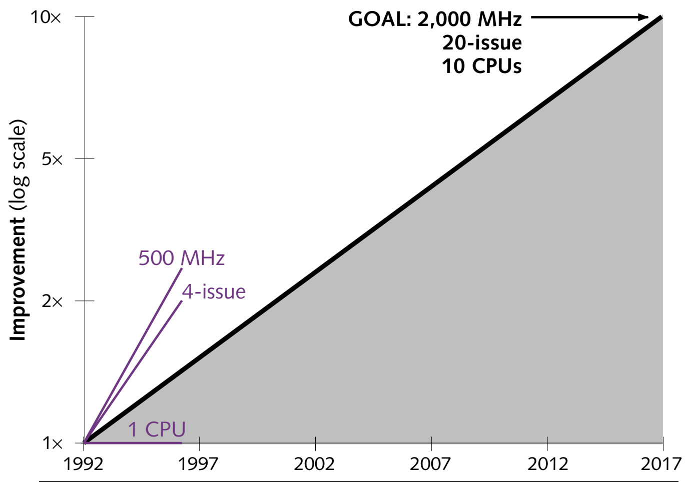

Figure 2. Starting with a 200-MHz two-issue Alpha processor with a single CPU on the chip, Digital planned to achieve a 1,000× performance increase by scaling each of the three parameters by 10×. In 1996, however, clock frequency and issue rate are scaling faster than expected while the number of CPUs per chip has not increased beyond one. (Source: Digital, Sites)

The first Alpha chip came out in 1992 at 200 MHz with one dual-issue CPU per die. Figure 2 shows the expected progress.

So what has happened in the subsequent four years? If all three dimensions changed equally, each should have improved by about 1.45× over the four years. In fact, Alpha CPU clock speed is ahead of our guestimate with a 2.5× improvement, issue width is a bit ahead at 2×, and on-chip multiprocessing is behind at 1×, as the figure shows.

Richard Sites

Increasing CPU clock speed is entirely self-contained; it depends only on chip designers and the fabrication process. In contrast, increasing the effective instruction-issue width requires both chip designers to create the more complex issue logic and compiler writers to fill the extra issue slots with useful instructions. Effective use of multiple processors is even more software intensive—it depends heavily on multithreaded operating systems and application software, which are just becoming widely available in the volume market.

The Alpha experience mirrors that of the rest of the industry: improvements in the three dimensions are not uniform but are proceeding at different rates based on their coupling with industry software trends.

What will happen across the industry in the next 4–5 years? CPU clock rates will certainly continue to improve. Today's optimizing compilers are good at filling two issue slots per cycle, and I expect them to improve over the next five years to effectively support something beyond six-issue—perhaps 8 or 10 instructions per cycle. I expect to see matching wide-issue hardware somewhere in the industry.

With the advent of Windows NT and with small multiprocessor systems becoming mainstream servers, there is now a growing emphasis in the software industry on multithreaded programs. In the coming years, small numbers of CPUs may also become common in desktop machines. The delays of network and Web access, and heavier use of multiple windows on large screens, will encourage further deployment of multithreaded software. Perhaps near the end of five years, it will be economical to place multiple processors (or more precisely, multiple sets of PC and registers sharing cache and function units) on a single chip. The maturity of the software industry may allow substantial growth in instruction issue and in the number of CPUs per chip.

How will this change microprocessor design? Across the industry, today's chips are largely able to execute code faster than we can feed them with instructions and data. There are no longer performance bottlenecks in the floating-point multiplier or in having only a single integer unit. The real design action is in memory subsystems—caches, buses, bandwidth, and latency.

Processor Speed Has Outstripped Memory

An anecdote: in a recent database benchmark study using TPC-C, both 200-MHz Pentium Pro and 400-MHz 21164 Alpha systems were measured at 4.2–4.5 CPU cycles per instruction retired. In other words, three out of every four CPU cycles retired zero instructions; most were spent waiting for memory. Another anecdote: software MPEG decoding may turn out to be limited by memory accesses, not by pixel arithmetic. Processor speed has seriously outstripped memory speed.

Increasing the width of instruction issue and increasing the number of simultaneous instruction streams only makes the memory bottleneck worse. If a CPU chip today needs to move 2 Gbytes/s (say, 16 bytes every 8 ns) across the pins to keep itself busy, imagine a chip in the foreseeable future with twice the clock rate, twice the issue width, and two instruction streams. All these factors multiply together to require about 16 Gbytes/s of pin bandwidth to keep this chip busy. It is not clear whether pin bandwidth can keep up—32 bytes every 2 ns?

I expect that over the coming decade memory subsystem design will be the *only* important design issue for microprocessors.

Richard L. Sites is a senior architect at Digital Equipment Corp., where he co-led the design of the Alpha architecture. He can be reached at sites@pa.dec.com.

William Dally

The End of Instruction Sets

What will a microprocessor look like in 10 years? Extrapolating current trends suggests the end of instruction sets as we know them. Microprocessors in 2006 will have explicitly parallel instruction sets. A variety of architectures will run standard parallel binaries via emulation. Cost and performance of these processors will be determined largely by the bandwidth and latency of the memory interface. To achieve adequate performance over this interface, processors will be integrated on the same chip as memory, and these integrated processor/memory chips will communicate among themselves over high-bandwidth point-to-point links.

Parallelism is required to exploit advancing technology. The number of devices and wiring per chip increases by 60% per year while clock rates increase at only 15% per year. Maintaining the historical 50%-per-year increase in sustained processor performance thus requires a 30%-per-year increase in parallelism. Today, processors that issue four instructions per clock are at the limits of the instruction-level parallelism (ILP) that can be extracted from legacy instruction sets. These processors also pay a high price in complexity because they perform this parallelization in hardware at run time. A 2006 processor with a peak issue rate of 32 instructions per clock must encode its instructions in an explicitly parallel manner.

The format in which binary application programs are exchanged, the ABI, must change as today's legacy instruction sets are abandoned. It is logical to switch the ABI not to a new hardware instruction set but rather to an explicitly parallel abstract instruction set that is easy to translate and emulate. Binary-to-binary translation or on-the-fly emulation software then converts the application binary to native code. Systems today translate legacy codes, including awkward memory-management structures and self-modifying code, onto existing architectures with reasonable performance. Translating a clean ABI to an architecture designed as a good target is a much easier task that should approach the efficiency of code directly compiled for the hardware instruction set.

Switching to an abstract or "soft" ABI makes economic sense for both software and hardware vendors. A soft ABI supports a broad range of hardware architectures and operating systems without the testing and distribution costs associated with multiple binaries. Soft ABIs also obviate the need to release recompiled binaries to optimize performance on a new hardware implementation. Soft ABIs benefit hardware vendors by reducing compatibility constraints and verification effort. Perhaps the greatest advantage of a soft ABI is longevity. By remaining independent of factors such as number of registers and issue slots, a soft ABI is likely to outlive a hard ABI.

Achieving Multiple Levels of Parallelism

To express the parallelism required by one or more 32-issue processors, a successful soft ABI and its hardware instruction sets must explicitly express parallelism across a spectrum of granularity. (The granularity of a computation is the number of operations that are scheduled as a unit.) One approach, implemented in the MIT M-Machine, involves using very long instruction words for the finest granularity, microthreads for an intermediate level of granularity, and conventional threads for the coarsest level of granularity.

At the finest level, the compiler (or translator) statically schedules operations into instructions to exploit ILP. But data-dependent branches and variable-latency operations, like memory loads, are difficult to handle via static scheduling. Such run-time variations are better handled by gang scheduling microthreads, sequences of long instruction words that run simultaneously and interact via processor registers. Microthreads make the instruction word longer in a manner that allows limited slip between segments of the instruction stream. This technique localizes the effect of a cache miss or a conditional branch to the segment containing the unpredictable instruction.

It is inefficient, however, to have a microthread wait on a very long latency operation, such as an I/O operation, since it consumes scarce hardware resources: thread slots and register names. Conventional threads are required to handle events at this level of granularity. By multiplying the parallelism available at each level of granularity, such an approach extracts significantly more parallelism from a given program than methods that operate at a single grain size.

Memory-Centric Computers

Computers in 2006 will be memory, not processor, dominated. Cost will be driven by memory capacity, and performance by memory latency and bandwidth. Processors will have so much instruction and arithmetic bandwidth that the memory system will almost always be the bottleneck, as Dick Sites points out. Placing the processor on the DRAM chip is the easiest way to achieve adequate bandwidth and latency to the first-level memory. High-pin-count chips using current-mode point-to-point signaling will enable high bandwidth to global memory. Chips with 256-bit buses operating at 1 GHz (32 Gbytes/s) should be economically achievable in this time frame.

William Dally

In a memory-dominated computer, the incremental cost of adding a processor is small. A typical memory chip in 2006 will store 4 Gbits of data on a silicon area sufficient to hold 1,024 1-GOPS single-issue processors. On such a chip, adding issue slots is advantageous as long as the incremental speedup of the last slot is greater than 0.1%. A 32-issue processor (with a few spare slots for yield) on such a chip would use about 4% of the area of a combined processor/DRAM chip. The 32nd issue slot would be worthwhile even if it only increases the average issue rate from 9.9 to 10, since this 1% increase is greater than the 0.1% cost.

Machines that require more than 4 Gbits (512 Mbytes) of memory will add memory capacity with processor/DRAM chips rather than DRAM-only chips for the same reason. The incremental cost is small compared with the expected gain. These processor/DRAM chips will be the jellybean chips of the future. In these memory-centric machines, the main advantage of CPU speed is in reducing the amount of time the memory (which will account for more than 90% of the cost of the machine) is tied up running a problem.

When you open the case of a 2006 PC, you are likely to find a number of integrated processor/DRAM chips connected by wide high-bandwidth links. The processor ensemble will have an explicitly parallel instruction set that extracts parallelism across the granularity spectrum. Applications will be distributed in a soft binary format that is mapped by software to the native instruction set. The machine will have a high ratio of operations to bytes, making efficient use of the expensive memory. Of course, this will all seem commonplace compared with the virtual-reality user interface.

William J. Dally is a professor of Electrical Engineering and Computer Science at MIT, where he directs a group that builds experimental microprocessors and parallel computers; check the Web at www.ai.mit.edu/people/billd/billd.html.

David Ditzel

x86 Becomes CPU Battleground

Things were exciting for microprocessors in the 1980s and early '90s, as RISC technology enabled a new instruction set almost every year. RISC designers focused on making better technical tradeoffs between hardware and software than in older, more complex mainframes and minicomputers. Tremendous energy was spent arguing about which instruction set was slightly better than another, as if how one encoded a 32-bit instruction might actually determine who would dominate the future of microprocessors.

Photo of Dave Ditzel

When RISC failed to deliver significant enough performance advantages, the market decided the winner would be the processor with the most software, awarding the prize to Intel's x86 architecture. Looking forward, no new instruction set seems likely to displace the x86 in the next decade. Does that mean the end to exciting new changes in microprocessor design? Not at all. If anything, the pace of change will accelerate, but the battles will be in different places.

Radical changes are coming to microprocessor design. In the past 20 years, microprocessor designers largely copied and extended design techniques used in mainframes and supercomputers. As VLSI allowed more transistors per chip, designers incorporated standard techniques such as pipelining, caches, superscalar instruction issue, and out-of-order execution. Now these standard techniques are nearly exhausted, and only small performance gains are yet to be extracted. This will force designers to try more radical techniques—some good, and some bad.

Upcoming Technical Battles

DRAM vs. Microprocessors. As processor-to-memory bandwidth becomes increasingly critical, there will be a merging of processors and DRAM. A single next-generation DRAM will hold 32 Mbytes. Half of that DRAM chip, 16 Mbytes, is more memory than in most of today's PCs; the other half could be used for a microprocessor. Who will get there first? Intel or a DRAM maker? Given that Intel is not currently in the DRAM business, there would seem to be a huge opportunity looming for semiconductor companies with a merged logic/DRAM CMOS process.

Multimedia Accelerators vs. Microprocessors. There is a growing gap in the bandwidth needed between the CPU and the graphics/multimedia accelerator. Standalone accelerators can get more powerful only if they successfully take over more of the processing that is today done in the CPU. On the other hand, future microprocessors will attempt to incorporate multimedia acceleration on the same die as the CPU. In the long run, this conflict would seem to be a losing battle for the graphics- and multimedia-accelerator makers.

New Instruction Sets vs. Compatibility. Despite excellent Pentium Pro performance, the x86 instruction set is severely flawed. Although RISC instruction sets made improvements, they too became bogged down with backward compatibility. New instruction extensions *(see* 101003.PDF*)*, seen in VLIW research, show significant performance gains are possible. So far, however, the computer industry has been unable to convert the existing software base to take advantage of new instruction sets.

Single-Chip Multiprocessors vs. Uniprocessors. Multiprocessors have earned a place in high-end servers. The question is whether they have a place on a single piece of silicon. Advances in uniprocessors have kept a fast enough pace that desktop users have not been willing to pay for multiprocessor solutions, nor have software vendors taken up the challenge of providing multiprocessor applications. A sufficient software base is nowhere in sight.

32-Bit vs. 64-Bit Addressing. While 64-bit addressing is technically attractive, any transition will take several years. The transition from 16 to 32 bits began only last year, with the introduction of Windows 95. Until a large fraction of the installed base of computers is 64-bit ready, software vendors will not have much incentive to produce programs that will not run on older 32-bit machines. Current 32-bit systems can do 64-bit arithmetic and access huge databases without the need for more address bits. The eventual transition will happen a few years after Microsoft forces it, which doesn't look to be anytime soon.

Low Power vs. High Performance. Low power used to mean low performance, but this will change as new techniques are developed to reduce power without sacrificing performance.

Wires vs. Gates. The past 30 years have been spent minimizing the number of gates per pipeline stage. As we move to deep submicron CMOS, wires become the limiting factor to performance, and new processor architectures are needed to take advantage of this change.

Moderating Factors

The following factors will moderate technical directions in future microprocessors.

x86 Dominates. Mainstream computing will be dominated by the x86 architecture. Yes, there will be niches for high-performance RISC servers, and perhaps Java processors for consumer applications, but most of the profits in the computer industry will be made on x86 microprocessors.

Avoid Recompiled Benchmarks. Performance metrics need to be refocused on real binaries, not benchmarks recompiled from source code. In the real world, programs are rarely recompiled for each new processor to get more performance. As microprocessors employ more radical architectural techniques and highly optimized compilers (that are rarely used outside of benchmarking), there is a growing disparity between the performance a user receives and what is advertised. Real binaries tend to be compiled for older processors and with very little compiler optimization. Making these binaries run well is what matters.

ILP Limits. Instruction-level parallelism has now reached practical limits, and the problem is exacerbated by real (i.e., unrecompiled) binaries. Superscalar techniques have gone to four instruction issues per clock. There simply is not much more parallelism to extract from existing binaries; don't expect to see well-designed uniprocessors launching 8 or 16 instructions, as there is minimal incremental performance available. Research into nonstandard techniques may provide interesting alternatives, such as multithreaded processors.

CMOS Technology Dominates. Because single-thread ILP is limited, and we can't just recompile all applications whenever we want a new microarchitecture, general performance improvements over several years can be reasonably predicted by simply following CMOS scaling rules. Each new generation of CMOS technology provides for a shrink in die size and faster speed. One can usually expect the clock speed of a given chip to increase by 25–50% with each new generation of technology.

Top System Vendors Poised to Fall

The most radical change to the computer industry will be an upset in the top computer companies. As the x86 becomes a more entrenched standard, Intel will be positioned to shift from being a microprocessor company to the world's largest full-fledged computer company. A new set of top players could emerge from those companies that are cross-licensed with Intel or those that find other ways to sell x86 processors. Expect to see huge battles as today's large computer companies realize that if they do not make their own x86-compatible processors, they will be relegated to being low-margin distributors of Intel systems. The stakes are for billions of dollars in revenue and for survival as profitable entities, so we should expect some exciting new solutions.

David R. Ditzel is president and CEO of Transmeta Corporation (Santa Clara, Calif.). He was a key participant in the development of the CRISP and SPARC architectures and coauthor of The Case for RISC. *He can be reached at* dave@transmeta.com.

Yale Patt

First, Let's Get the Uniprocessor Right

A billion transistors on a single chip will present a challenge very different from 10 million or 100 million transistors on a chip. The design point is always performance or cost. If the design point is cost, as it often is, the answer is easy: continue to integrate more and more system functionality on the chip. Long before one billion transistors, the entire system can reside on a single chip.

Photo of Yale Patt

But if the design point is performance, we have a serious partitioning problem. Is it better to partition the billion transistors into smaller units and implement a multiprocessor on a chip, or to build a very wide VLIW uniprocessor, or to use the billion transistors in support of a processor with a single instruction stream, interconnecting multiple chips to obtain a multiprocessor system? A fourth paradigm, the coarse-grained multithreaded uniprocessor, right now is in my view just an MP in uniprocessor clothing, since the critical design requirement is identical.

At the level of 100 million transistors, the answer is very clear to me: use the transistors in a uniprocessor. Recent studies show that, on real-world code, even a four-wide uniprocessor spends more than 90% of its time stalled due to memory pin bandwidth; these studies argue that additional processors on the chip would only exacerbate the problem.

VLIW makes no sense for general-purpose processing, as was proved years ago by the Cydrome and Multiflow experiments on numeric code. The recent flurry of interest in general-purpose VLIW was mainly due to the rumor that Intel and Hewlett-Packard would produce a VLIW P7. That rumor has since been pretty strongly disavowed. VLIW still has value, particularly for special-purpose structures such as graphics accelerators and DSP processors. But those situations are well confined, and the VLIW solution bears strong resemblance to microcode of an earlier era.

The final option (for 100 million transistors) seems the correct one: use the transistors in a powerful uniprocessor. Both deeper pipelines and wider issue widths both provide greater opportunity for parallel execution. But they also provide a substantially greater penalty for branch misprediction.

Therefore, a goodly number of transistors can be effectively used to improve the branch predictor. Wider issue makes sense only if the data path supports it. That means more function units, more bypass paths, and more logic for correctly routing and synchronizing the results of processing.

At 100 million transistors, I don't believe we have at all run out of steam in things we can add in support of a single instruction stream. At one billion transistors, the question is less easy to answer cavalierly.

Still, I think I would opt for a yet more powerful uniprocessor. My argument continues to hinge on the partitioning problem. A successful design partitions the system so that costly communications are close together, and communications that can tolerate long latencies are far apart. At one billion transistors, one can put on the chip multiple levels of cache, with an L3 cache sufficiently large that the processor should not stall due to lack of memory bandwidth. An augmented instruction set could manage (using prefetch and poststore) the movement of code and data so each is present in the single-cycle, very fast L1 cache exactly when needed.

Issue widths will be wider, cycle times faster, and pipelines correspondingly deeper. This will require more transistors devoted to the branch predictor and more transistors devoted to the data path (additional function units), chewing up the available billion-transistor budget.

Of course, the above makes sense only if algorithms can be redesigned to take advantage of such a powerful dynamically scheduled engine. But if we forbid lazy microarchitects and lazy compiler people, we must insist on equal attention from the algorithm people.

So I, like everyone else, expect to see MP activity increase over the next several years. But I think the individual component of that MP, even when each chip boasts one billion transistors, will be a very powerful single-chip uniprocessor, speculatively executing very wide issue instruction streams with very nearly 100% branch-prediction accuracy, at very short cycle times made possible by very deep execution pipelines.

Yale Patt is a professor of Electrical Engineering and Computer Science at the University of Michigan, where he directs research in high-performance computer implementations. He can be reached at patt@eecs.umich.edu.