Microprocessor Design Issues: Thoughts on the Road Ahead

As power becomes increasingly important, the cubic trade-off between time and power (T^3P = constant) forces designers to use relatively cheap area to increase performance rather than the expensive power required by higher clock rates. As designers focus on power, the question of optimal power-oriented architectures is apparent.

Michael J. Flynn
Patrick Hung
Stanford University

With the scaling of technology promising increases in chip frequency and especially transistor density, system designers must make trade-offs for a rapidly moving target. They must constantly deal with area, time, power, reliability, and technology design trade-offs as well as enormous design complexity at the same time. Here, we look at the technology roadmap and what it means to computer architects, updating our views of six years ago.[1]

SIA's roadmap

For the past 30 years, process technology has mainly driven microprocessor designs. The Semiconductor Industry Association (SIA) regularly projects the progress of process technology; these projections have become the basis of and assumption for new chip designs.[2] Published officially as the *International Technology Roadmap for Semiconductors*, these projections are commonly called the SIA roadmap. Although the SIA roadmap changes—becoming more or less ambitious over the years—the technological advances have been, and will likely continue to be, formidable.

Table 1 is the SIA process technology projection for the next 13 years. The table focuses on lithography or process generations as defined by the size of the smallest transistor gate that manufacturers can produce with the cited process. Currently, each process generation is in terms of nanometers. Up until recently, the industry referred to process generations in terms of microns with earlier generations being 0.13, 0.18, 0.25, 0.35 µm, and so on.

Table 2 shows the SIA roadmap for high-performance microprocessors. According to the 2003 roadmap, experts expected their on-chip clock frequency to increase by more than 10 times to 53.2 GHz by the end of 2018; at the

This paper was based on the plenary talk by Michael J. Flynn at the 2003 Federated Computing Research Conference in San Diego, with some updated information on processor development in the past year.

Published by the IEEE Computer Society

Table 1. SIA roadmap for memories and microprocessors.

Year of introduction	Feature size (nm)	Wafer size (cm)	Microprocessor physical gate length (nm)	Defect density (defect/cm^2)* Memory	No. of transistors (millions)
2004	90	30	37	0.28	553
2007	65	30	25	0.32	1,106
2013	32	45	13	0.37	4,424
2018	18	45	7	0.35	8.848

* For the same years, SIA predicts that microprocessor defect density will remain at 0.14 defects/cm^2. Die size, 3.1 cm^2, also remains consistent.

Table 2. SIA roadmap for high-performance microprocessors.*

Year of introduction	No. of transistors (millions)	Clock frequency (GHz) On chip	Clock frequency (GHz) Off chip	Maximum no. of wiring levels
2004	553	4.2	2.5	10 to 14
2007	1,106	9.3	4.9	11 to 15
2013	4,424	23.0	18.6	12 to 16
2018	14,045	53.2	56.8	14 to 18

*Projections show chip size remaining a constant 310 mm^2.

same time, the number of transistors on a single die would increase nearly 30 times to 14 billion. Recent developments probably portend a lowering of the clock frequency projections.

Table 3 shows the SIA roadmap for high-volume, cost-performance microprocessors. The number of transistors is about 30 to 50 percent less than for high-performance microprocessors, but the general trends are similar to those of high-performance microprocessors.

Table 3. SIA roadmap for cost-performance microprocessors.

Year of introduction	Transistors density (millions/cm^2)	Transistor count Introduction*	Transistor count Production*
2004	138	4.2	2.5
2007	276	9.3	4.9
2013	1,104	23.0	18.6
2018	3,506	53.2	56.8

* Chip size at introduction is 280 mm^2; at production, 140 mm^2.

Table 4 shows the expected power consumption for three types of microprocessors:

- high-performance processors, typically for server and workstation applications;
- high-volume cost-performance processors for consumer products; and
- low-power processors for mobile applications.

In all three categories, experts expect the power consumption to rise gradually in the coming decades.

Table 4. SIA power roadmap for three types of microprocessors.

Year of introduction	Power (W) High-performance	Power (W) Cost-performance	Power (W) Portable
2004	158	84	2.2
2007	189	104	2.5
2013	251	138	3.0
2018	300	168	3.0

The SIA roadmap focuses attention on transistors (area), their speed (with reduced switching time), and power dissipation. But larger issues remain: the design time for enormously complex designs and the integrity of the design (in terms of reliability, testability, and security). Following the SIA roadmap's focus, we first look at the area-time-power issues, but any commercial implementation would carefully consider these larger issues at the outset. We will address design time and

Table 5. Some state-of-the-art general-purpose computational processors.

Processor type and model	No. of cores	No. of threads	Cache	Cache size (Mbytes)
Server				
Intel Montecito	2	2 × 2	L3	24
Sun Niagara	8	8 × 4	L2	3
IBM Power5	8	8 × 2	L3	36
Workstation				
Intel Pentium 4 Prescott	1	1 × 2	L2	1
AMD Athlon 64	1	1 × 1	L2	1
Laptop				
Intel Dothan	1	1 × 1	L2	2

computational integrity later in this article.

Before examining any specific design issues, it's helpful to consider the current processor market.

Processor market

There are two main classes of microprocessors: general-purpose computational microprocessors for server and PC applications, and specialized microprocessors for embedded and system-on-chip (SoC) applications. The former contains one or multiple on-chip homogeneous processors with off-chip main memory; the latter combines one or multiple heterogeneous processor cores with on-chip memory.

General-purpose computational processors

Historically, performance drives these designs, especially in server processors. For laptop processors, performance and power consumption also drive the design; laptops have a fairly modest power budget because of battery limitations. Generally speaking, server processors have a power envelope between 100 W and 130 W, workstation processors have a power envelope between 60 W and 90 W, and laptop processors have a power envelope between 20 W and 30 W.[3]

To satisfy the fast processor requirements and remain within memory limitations, all computational processors use a large cache complex that occupies most of the die area. As a result, the processor logic designs are not particularly area limited and can be highly elaborated, such as those required for very-long-instruction-word (VLIW) designs or a superscalar design with multithreading capabilities.

Table 5 shows some state-of-the-art general-purpose computational processors.[4-9] All have large on-chip caches. In this table, the number of cores refers to the number of on-die processors, and number of threads refers to the degree of multithreading on each processor. For instance, 2 × 2 means two processor cores, each having two simultaneously executing threads.

Unlike the other processors in Table 5, IBM's Power5 is a 95 mm × 95 mm multichip module (MCM), consisting of four processor die (eight processor cores) and four level-3 (L3) cache chips. The eight cores located on the module form a tightly coupled symmetrical multiprocessing (SMP) group.

Historically, clock speed drove computational processor performance. In the last 20 years, the processor clock frequency doubled approximately every 18 months (Moore's law), but this trend slowed down significantly in the last couple of years. For example, the clock frequency of Intel's Pentium 4 processor only increased from 3.06 to 3.2 GHz between 2002 and 2003. This represents a less than 5 percent gain in two-thirds of a Moore's law interval.

Although the 2003 SIA roadmap projected that the on-chip clock frequency of high-end microprocessor would reach 4.2 GHz in 2004, this did not happen. In May 2004, Intel announced that the company had hit the "power wall" with traditional clock speed scaling. It canceled two single-core processors (Tejas and Jayhawk) and accelerated a strategy to bring dual-core processors to the mainstream PC markets.[10] Indeed, many believe that the 31-stage superpipelined Tejas was extremely power hungry but could not deliver performance that justified the power consumption. Other processor vendors, such as IBM, Sun, and AMD, also announced that multicore processors would be the norm in their future high-performance lines.[7,11,12]

This signals an end to the gigahertz-clock-speed race in the computational processor market. Now, processor vendors are competing in terms of number of processor cores and processor threads. In the near future, we project a gradual increase in clock frequency but a significant growth in the number of on-chip processor cores. Performance gain will come primarily from instruction- and thread-level parallelism rather than pure clock speed.

Table 6. Some state-of-the-art embedded and SoC processors.

Processor type	Model	Architectural overview
Low-power embedded	Infineon TriCore 2	Tricore 2 follows a similar architecture as in the original Tricore with three pipes dedicated to execution, load-store activity, and looping. It supports dual threading to lower energy-expensive external-bus accesses. It has embedded SRAM to support hard real-time applications and reduce power consumption. At 250 MHz, it consumes about 75 mW in UMC's 130-nm process technology.
Application	Nomadik STn8800	The STn8800 uses the AHB bus to combine an ARM926EJ RISC control processor core with a Java accelerator, an MMDSP+ VLIW DSP core, and hardware video and audio accelerators. When decoding MPEG-4 QCIF (Quarter Common Intermediate Format) at 15 fps with MP3 audio, it requires about 20 mW in a 130-nm process technology.
Media	Trimedia TM5250	This VLIW architecture has 29 functional units. The integer operations use an 11-stage pipeline, whereas floating-point operations use a 16-stage pipeline. Each VLIW instruction bundle can contain up to five operations, including three branches. To reduce the branch penalty, it uses dynamic branch prediction. Power consumption is about 1 W at 500 MHz in a 130-nm process.
Extremely specialized, high performance	Xelerator X10q	The X10q is the first 40 Gbyte/s network processor. It consists of 200 VLIW processor cores pipelined together with interfaces to memory and functional units. There are about 1,040 pipeline stages, and the aggregate data flow in the pipeline at 200 MHz is 26.2 Pbytes/s. The power consumption is about 9.5 W in TSMC's 130-nm process technology.
Soft IP core	Silicon Hive Avispa+	Avispa+ is a VLIW machine with 768-bit instruction words. The instruction words are divided into 60 individual instructions dispatched in parallel. At 150 MHz, it can execute up to 9 billion instructions per second, consuming 150 mW. Avispa+ is user-configurable at design time to allow different mixes of execution resources for different applications.

Embedded processors and SoCs

There are many types of embedded and system-on-chip (SoC) processors: Some are very specialized and have extremely high performance (such as network processors), and some are very cheap and find extensive use in everyday electronic appliances. Table 6 shows some of today's prominent embedded and SoC processors.

Although most research focuses on general-purpose computational processors, they represent only a small percentage of the processors actually produced in any year. In fact, specialized processors for embedded applications are the dominant processor types and represent the bulk of processor implementations.[13]

Figure 1 illustrates that combinations (amalgamations) of heterogeneous processors managed by more sophisticated microprocessor units represent a major growth area for processor design. In this figure, the label SoCs represent both SoC[15] and system-on-package (SoP)[16] applications.

SoC designs often use many different types of processors, depending on what suits the application. Often, companies purchase generic processors as intellectual property (IP) and integrate them into a design. For example, the ARM 7 TDMA[17] is a popular, licensed 32-bit RISC processor core. Its success is in part due to its compact logic design and small memory footprint, and in part due to the many third-party IP components available with its AMBA bus. Even in a specialized SoC design, processor cores are likely to have been designed by others. Generally, the industry has gravitated toward several methodologies for the design and license of processor cores; Table 7 summarizes typical scenarios.

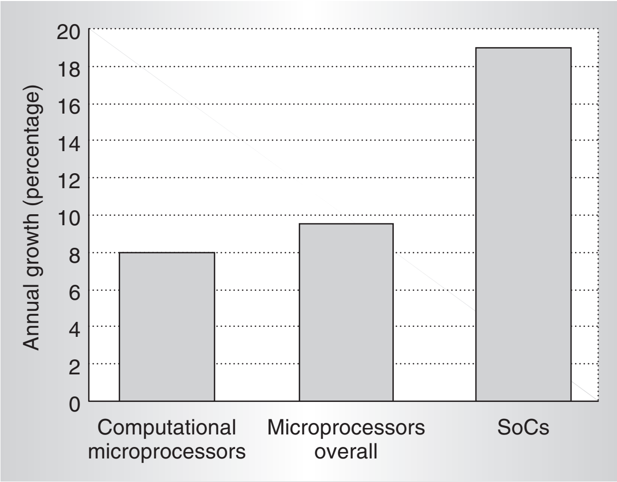

Figure 1. Annual growth in the MPU marketplace.[13,14]

Hard IP components are physical-level designs that use all the features available in a process technology, including circuit design and physical layout. Vendors distribute many analog and mixed-signal IP components (such as SRAMs and phase-locked loops) in this format to ensure optimal timing and other design characteristics.

Firm IP components are gate-level designs that include device sizing but are generally applicable to many fab facilities with different process technologies. *Soft IP* components are logic-level designs in synthesizable format and are directly applicable to standard-cell technologies. One key advantage of this approach is it lets users adapt the source code to fit their design. Clearly, the more-optimized design is less flexible but has better cost-performance trade-offs. For example, designers might use soft IP in a pilot production of a field-programmable gate array (FPGA) implementation and hard IP in the mass production of an application-specific IC (ASIC) implementation. Recently, there has been a growing trend toward using a structured ASIC, effectively merging hard and soft IP in the same design.

SoC implementations have many advantages over a general-purpose design. For many applications, the system specifications are well defined and include real-time delay constraints. In addition, the application permits processors to specialize, handling a particular function. Doing so permits a reduction in clock frequency (and power) because the design can regain performance by employing straightforward concurrency in the architecture (for example, the use of a simple VLIW for digital signal-processing applications).

In many SoC market segments, the extensive custom optimization of high-performance processors is difficult to justify, and they thus commonly employ off-the-shelf processor core designs. Because designers often know the storage size for programs and data at design time, they can include specific storage structures on chip. Today, designs use static RAM instead of dynamic RAM (DRAM) because DRAM traditionally requires a process technology that minimizes leakage current but reduces logic speed. This approach is changing, as embedded DRAM using silicon-on-insulator technology is becoming more popular.

With multiple storage units, multiple processors (some specialized, some generic), and specialized controllers, one important SoC challenge is to design a robust bus hierarchy that can ensure timely communications and enable complex system integration and verification.

Table 7. IP design methodologies.

Type	Abstraction level	Description
Customized hard IP	Physical	Designs can use IP only in the designated ASIC foundry and having a fixed process technology. These require no customization. Performance and die area are not further optimizable.
Synthesized firm IP	Gate	This class of IP will work with multiple ASIC foundries. Die area is somewhat fixed but designers can slightly optimize performance during physical layout.
Synthesizable soft IP	Register transfer	Working with any ASIC foundry, this IP uses any process technology. Designers can more fully optimize performance and area during logic synthesis and physical layout.

Area-time-power trade-offs

To analyze the future microprocessor architecture, we must first understand the trade-offs among area (A), time (T), power (P), and technology (S). As Figure 2 shows, a theoretical 3D surface can represent the optimal design space at a fixed process technology. For example, higher performance (smaller T) inevitably requires more power (P) and area (A).

The driving force in design innovation is the rapid advance in technology. As technology advances and feature size shrinks, the three other design considerations benefit from one process generation to another, resulting in higher speed, smaller area, and reduced power consumption.

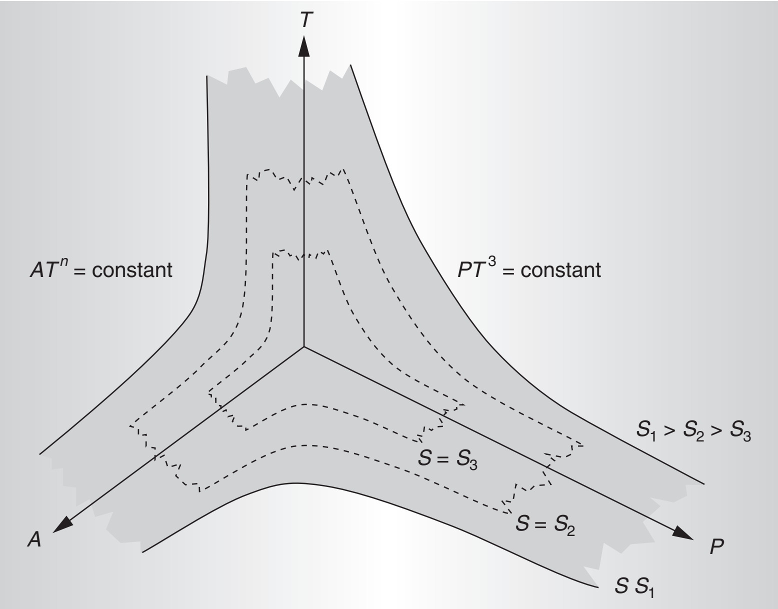

Figure 2. Trade-offs among area (A), time (T), power (P), and technology (S).

Power consideration

For most applications, power is the real price of performance. Processor power dissipation is the sum of dynamic switching power and static leakage power:

$$\begin{aligned} Power_{\text{total}} &= \text{switching power} \\ &+ \text{leakage power} \\ &= [(1/2) \times C \times V \times \Delta V \times f_{\text{clk}}] \\ &+ (I_{\text{leakage}} \times V) \end{aligned}$$

In most applications today, switching loss is much more significant than static leakage loss. As threshold voltage (V_T) continues to drop, leakage loss increases exponentially, creating a major future problem. Fortunately, techniques such as multiple V_T design and adaptive body bias (ABB) have proved effective in alleviating static loss.[6,18]

Performance-oriented designs take advantage of the decreased capacitance that the process technology provides to improve switching time accordingly:

$$\Delta t = (C \times \Delta V)/I$$

Large switching current I significantly improves switching times. If V_T is small compared with the supply voltage, the switching current is roughly proportional to the square of the supply voltage. Thus, the switching frequency is roughly proportional to the supply voltage, which implies that there is a cubic relationship between T and power consumption (P) at the same feature size:

$$P \times T^3 = \text{constant}$$

This cubic relationship is actually an approximation; Zyuban and Strenski provide a more in-depth analysis.[19]

Based on this cubic rule, it requires roughly eight times more power to reduce processing time by half. Using the same argument, it is apparent why today's high-end processors face a formidable power wall: As the on-chip clock frequency reaches 4 GHz, it becomes difficult to dissipate the switching and leakage power losses. In addition, merely increasing clock speed can no longer reduce processing time significantly because of memory latency and other pipeline bottlenecks. Thus, there are very few incentives to running at 4 GHz using the current process technology.

In fact, microarchitects have faced a similar power wall in the past. Up until the early 1990s, the bipolar technology of emitter-coupled logic (ECL) dominated high-performance applications, such as mainframes and supercomputers. At power densities of 80 W/cm^2, these chips required a module package having some form of liquid cooling, like the chip in Figure 3.[20]

As Figure 4 shows, CMOS circuit performance approached that of bipolar circuits in around 1993. The extraordinary cost of cool-

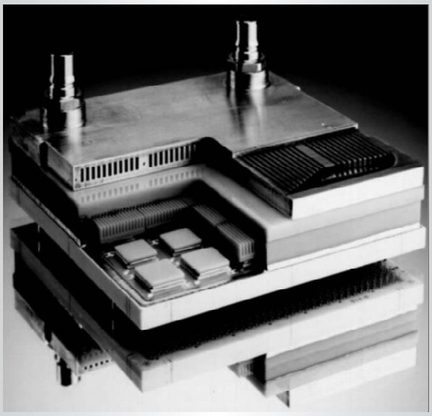

Figure 3. Hitachi M-880 ECL (bipolar) 500-MHz processor module (circa 1991). The 10 × 10 cm module dissipated 800 watts. Cooling this die meant sealing it in helium and pumping chilled water across the water jacket at the top.[20]

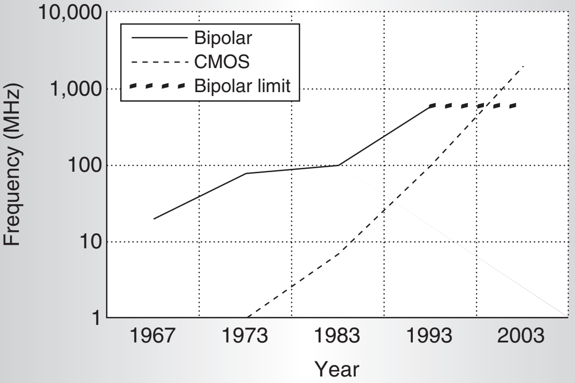

Figure 4. Processor frequency for bipolar and CMOS over time (http://www.macinfo.de).

ing bipolar circuits became unjustifiable, and the bipolar era ended. Now, as CMOS designs approach the same power densities, architects might have to consider similar cooling techniques for high-end applications.

Performance (time) consideration

As with the power-time (*P*-*T*) trade-off, time (*T*) and die area (*A*) have similar design trade-offs. Invariably, a faster functional unit (such as an arithmetic logic unit or memory interface) is larger and more expensive than a slower functional unit, or no reasonable implementation would use a slower functional unit. Ullman shows that an implementation is theoretically bounded by AT if volume (I/O, power dissipation, and so on) limits it; or AT^2 in implementations that are communications (internal bisection) limited.[21] So depending on the type of functional unit and circuit implementation, doubling processing speed could increase die size by 2 to 4 times.

$$A \times T^n = \text{constant}$$

where n is typically between 1 and 2.

The trade-off of cost, performance, and power consumption is fundamental to any system design. Although there are numerous design possibilities in this area-time-power continuum, the application always drives the optimum design point. Specifically, for high-end applications, performance is the most important design consideration. Here, architects face three major performance bottlenecks or walls:

- *Memory wall.* The access time to memory depends on wire delay that is relatively constant with scaling. As processor clock rate increases, so do cache miss times, creating a memory wall on performance. So far, significant processor hardware support (such as increased cache size and branch tables) and an increased emphasis on memory performance has helped designers manage cache misses and branches, reducing access to memory requirements and alleviating the impacts of the memory wall. Nevertheless, the memory wall is still a serious issue, especially for database applications.
- *Frequency wall.* The maximum pipeline segment size (the number of logic gates in one pipeline stage) determines clock frequency. Reductions in segment size beyond a certain point are difficult, creating a frequency wall. Here, the question is how to improve performance without reducing the segment size.
- *Power wall.* Higher frequency implies greater power density. As clock speed increases, the cost of removing the resulting heat limits a design and its effectiveness at some point.

Although these three walls arise from physical constraints and application program behaviors, new processor and memory paradigms can certainly improve memory access and access predictability. For example, area concurrency—not clock frequency—can help improve performance. That's why processor vendors are now using the extra area to implement multiple elaborate processor cores with large memory and sophisticated bus organization.

Performance is the obvious driver for high clock rates. But a very high clock rate is expensive in power terms, and it might not produce extra performance. In the past, the aggressive high clock rates did not come mainly from process technology scaling. As late as five years ago, the SIA roadmap predicted achievement of a 1-GHz clock only late in this decade. There have been three basic contributing factors enabling the extremely high clock rates:

- increasing the number of segments (or stages) in the pipelines (superpipelining);
- better clock generation and distribution, reducing clock skew (via elaborate clock tree structure); and
- better circuit design (for example, skew tolerant domino logic[22]) minimizing clock overheads.

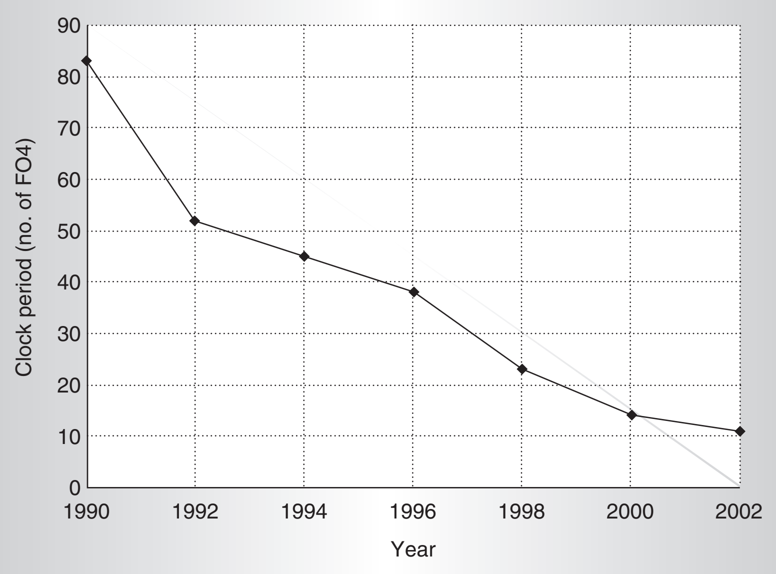

Figure 5. Number of FO4 gate delays in a cycle for the last seven generations of Intel processors. The *x*-axis represents the years of introduction for these processors.[24]

Figure 5 shows that the length (in gate delays) of a pipeline segment has decreased significantly, probably by more than five times, measured in units of fan-out of four (FO4) inverter gate delays.[23] On the other hand, recent research indicates that the pipeline segment shorter than eight FO4 delays for integer operations and 6 FO4 delays for floating-point operations not only increases power consumption but also lowers overall performance.[24]

Simply increasing the pipeline rate by increasing the number of pipeline segments does not necessarily increase performance. Indeed, performance decreases if there are excessive pipeline breaks and segment clocking overhead.

For the best performance, the optimum number of pipeline segments, S_{opt}, is a function of the probability of a pipeline disruption (b) and clock overhead (C). For an unsegmented instruction execution time, T_e, a simple model using a linear single-issue pipeline[25] shows that

$$S_{opt} = [(1 - b)\, T_e / bC]^{1/2}.$$

Low clock overhead (very low C) enables increased pipeline segmentation, but performance does not correspondingly improve unless the probability of pipeline disruption, b, is also low. So processors also employ large branch table buffers and branch vector prediction tables, significantly decreasing average delays because of lower branch probability.[1]

Disruptions can also come from cache misses, which require another strategy, such as very large and multilevel on-chip caches. With higher clock rates and relatively constant DRAM latency, the cache-miss time can exceed 300 processor clock cycles. So as the processor speed increases, the cache miss penalty (in cycles) inevitably increases. To execute the same number of instructions per cycle (IPC), the faster processor must use an even larger cache to reduce its target miss rate.

So, how big should the cache be? For a wide range of cache configurations and workload conditions, the cache size (in number of bytes) must at least double to cut the target miss rate in half.[26] Assuming the delay per miss is constant, the cache size should be at least proportional to the processor clock speed to compensate and maintain the same IPC.

Figure 6 shows the cache sizes for implementations of Intel's Itanium. As proces-

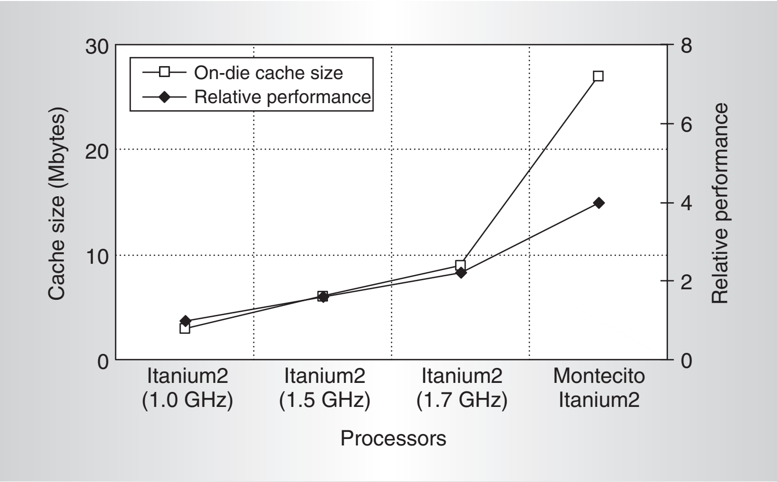

Figure 6. Cache size and processor performance for Itanium processors.

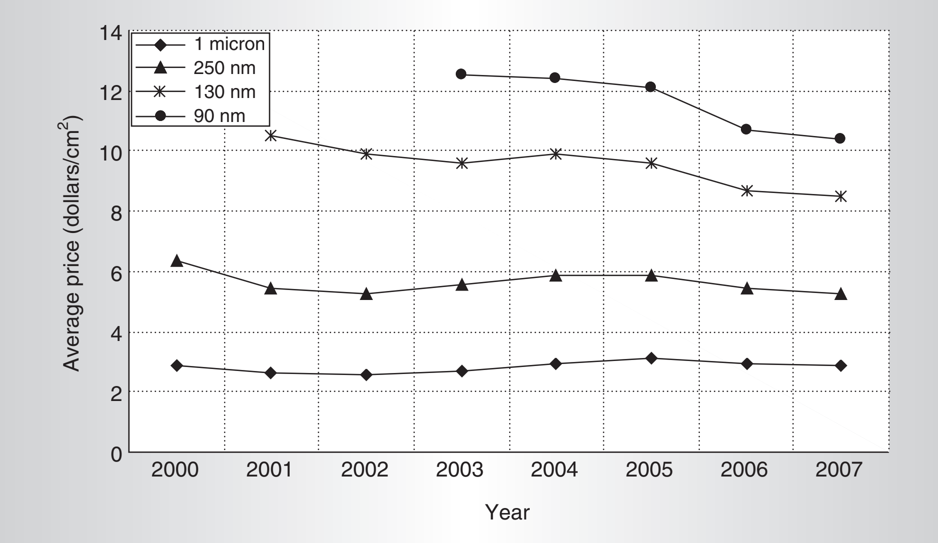

Figure 7. Average foundry price trend, 2000 to 2007.[28]

sor performance increases, so does cache size. Figure 6 shows performance data based on reported SPECmarks (http://www.spec.org) with an estimate for the dual-core Montecito based on an assumed 80 percent effectiveness for the second processor core. Intel's Montecito has a 24-Mbyte, L3 cache and a total cache size of 27 Mbytes.[6] Recent Itanium implementations show caches occupying over 80 percent of the die area, approaching 90 percent. Because interconnect delay will continue to limit memory performance, the trend toward cache dominance of die area will continue.

At clock rates above 1 GHz, one-cycle access time to the level-one (L1) cache is a problem for cache sizes greater than 32 Kbytes.[27] Hence, most general-purpose computational processors use multilevel caches and, in some cases, multicycle (usually two-cycle) pipelined L1 caches.

Cost (area) consideration

From the economic point of view, a new foundry with the latest technologies takes $2 to $5 billion to build. Needless to say, the wafer and mask costs are significantly higher with newer, smaller feature sizes. As an example, the wafer cost at 130 nm is currently more than double than the wafer cost at 250 nm.

At 130 nm, a 1 cm^2 die costs about $10 while dies that are several generations older cost about $2; Figure 7 shows this trend. It is always expensive at the leading edge, but eventually costs decline. More significantly, as processes move beyond 130-nm lithography, mask costs have skyrocketed. The average mask set costs $500,000 and a more complicated mask can cost as much as $1 million. The primary reason for the rising mask costs is that the lithography requires the adoption of the expensive reticle enhancement technologies (RETs), which allow the replication of features on wafer that are significantly smaller than the wavelength of the light used for the exposure.

Despite the increasing nonrecurring design costs, the marginal production cost per die is indeed low, and the cost per transistor is becoming lower. Consider the following example:

- A single wafer, 30 cm in diameter, produces about 700 die, each 1 cm^2. A die in a 90-nm process has 100 to 200 million transistors per cm^2.
- Yield (in terms of percentage of good die) is $e^{-\rho A}$ where ρ is the defect density, and A is the die area. At $\rho = 0.2$ and $A = 1$ cm^2, $\rho A = 0.2$, so the yield is 80 percent.
- A state-of-the-art fab for 30-cm wafers (with $\rho = 0.2$) might cost $3 billion and require $5 billion per year in sales to be

profitable. This means that it must have at least 5 million wafer starts and produce almost 5 billion 1-cm^2 die per year.
- At this order—$2,000 to $10,000 per wafer—that's upwards of $2 per die or $2 per 100 million transistors. So, how do implementations use these transistors at such a low cost?

Die area, as a measure of production cost, is only a consideration in midrange-server applications. At the high end, other system components—such as the package, memory, cooling, and storage devices—largely determine costs. Surprisingly, in small-die (low-cost) systems, die size is also not much of a cost factor because packaging and testing determine most of the chip costs. The problem is that with increasing transistor density, the smallest cost-effective die might soon contain at least 10 million transistors.

Looking ahead, how do you use a die containing 200 million to more than 2 billion transistors? Many argued that transistors will be "too cheap to meter," and die size will no longer be important.

However, we believe that die size will still be an important consideration even in SoC design. Designers will need many more transistors to effectively reduce power and to implement a wide variety of system functions. Indeed, you can use area to significantly broaden the design range with imaginative new system applications.

Beyond area-time-power trade-offs

Besides area, time and power, successful strategies must address the issue of computational integrity and productivity.

Computational integrity

Computational integrity involves multiple design considerations, such as testability, reliability, serviceability, recoverability, fail-safe computation, and security. The genesis of computational integrity goes back to the era of mainframe computers, but today's large, gigahertz-clock-frequency designs require techniques developed then and much more.

Today, design for testability (DFT) means adding accessibility paths, such as scan paths, which enable an external tester to verify a chip's functionalities over a broad range of state combinations.[29] Based on the specific design, automatic test pattern generation (ATPG) tools generate efficient vectors that can guarantee a very high coverage. Unfortunately, as transistor count continues to rise and clock speeds go beyond the gigahertz range, boundary scan paths are becoming ineffective for several reasons:

- With a billion transistors on a single die, it takes too much time to sufficiently verify all the state machines and logic blocks in a design. In essence, there is no way to guarantee system integrity unless every chip can verify its functionalities on its own.
- As the cost of transistors continues to drop, it makes economic sense to use extra logic to implement built-in self-test (BIST) in each logic block.[30]
- Boundary scan is becoming infeasible due to restricted timing budgets. When the system clock runs at gigahertz speed, the fault spectrum goes beyond simple wire opens and shorts. A fault is more likely to arise from limited tolerance to transmission line loss, impedance discontinuities, crosstalk, and nonlinear driver effects.

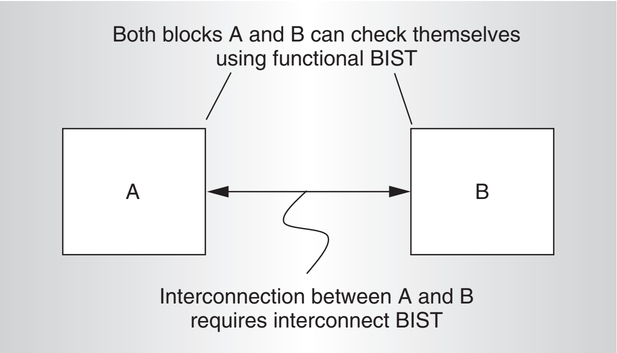
Both blocks A and B can check themselves using functional BIST
A
B
Interconnection between A and B requires interconnect BIST

Figure 8. Functional and interconnect BIST.

Generally speaking, there are two types of BIST: functional and interconnect (or IBIST). Functional BIST verifies memory and logic blocks, whereas IBIST verifies high-speed interconnections. Today, designers primarily use IBIST between two chips, but this technique will also be useful on-chip when the clock frequency reaches tens of gigahertz. Figure 8 shows the BIST architecture of a high-speed chip with two internal functional blocks connected by a

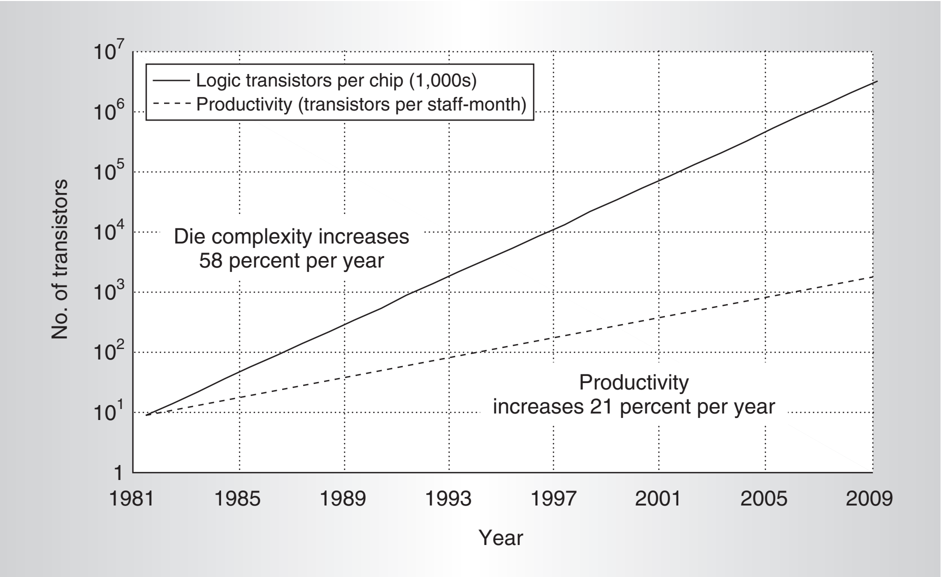

Figure 9. Productivity gap (http://www.sematech.org).

high-speed on-chip serial link. In this example, the chip only needs one IBIST engine because there is only one high-speed serial link.

Reliability, availability, and serviceability (RAS) are parts of a very important aspect of computational integrity. RAS involves error detection and correction, error diagnosis, failover operation, and fault recovery.

Higher clock frequencies increase electrical noise and noise sensitivity, and smaller feature sizes lead to more failures over time from electrostatic overstressing and electromigration. Although transistors and circuits might fail, this need not lead to demonstrable faults in the entire system. Decades ago, when mainframe machines cost millions of dollars, computers error checked, diagnosed, check pointed, and retried computations. With the advent of consumer-oriented computing, the priorities had shifted to cost and performance. Now, as transistor costs continue to drop, reliability is becoming available for all applications. Specifically, redundant and fault-tolerant systems will likely become a norm in future computer architectures.

The techniques supporting fault-tolerant systems are largely known, but architects must extend and integrate them into emerging processor architectures. For example, Intel's dual-core 64-bit Montecito processor uses multiple error protection schemes (such as Pellston technology).[6]

System security is also an emerging aspect of computational integrity. Thanks to the Internet and wireless communication networks, security has become a major issue. Computer viruses and worms attack millions of machines. This problem is becoming more serious as Web services become more popular, especially in business applications. To this end, a number of industry initiatives, including the Trusted Computing Group (https://www.trustedcomputinggroup.org) and the Next-Generation Secure Computing Base (NGSCB, http://www.microsoft.com/resources/ngscb/default.mspx), aim to provide fail-safe operation. Although this area still requires much research, the best solution will involve innovations from both software and hardware.

Productivity gap

Over the past 20 years, die complexity (transistors per die) has grown at 58 percent per year, but designer productivity (through the use of CAD tools, for example) has only increased by 21 percent per year. This design gap, shown in Figure 9, is formidable and has served as the basis for many design ideas.

The most effective way to reduce the productivity gap is to adopt a higher level of design abstraction, as Figure 10 shows. This happened many times in the past, such as when physical transistor design was replaced by schematic diagrams and gate-level design, which in turn was replaced by register-transfer level (RTL) design. It is likely that RTL will eventually migrate to the electronic-system level, which supports progressive refinement from the untimed functional model to a timed functional model, then to a transaction-level model and a cycle-accurate RTL model. To maximize design reuse and to increase productivity, it would be helpful to perform design and verification in a single, unified language such as SystemVerilog, for example (http://www.systemverilog.org).

Another effective way to shorten the design time is to reuse existing IP. Future hardware designers are more likely to deal with IP (controllers, small processors, floating-point units, digital signal processors, and memory) than ICs. This opens up a whole new field for specialized-hardware design services. At the system-level, dealing with designs from multiple sources requires acquiring these designs (the IP) and amalgamating them into a system. This requires the implementation, scaling (physical mapping), validation, and testing of each element to occur before interconnecting them by a bus or switch.

The efforts of the VSI Alliance has been particularly helpful in developing a standards framework for module interfaces and buses (http://www.vsi.org). A growing design industry now operates by selling designs rather than hardware. Indeed, hardware design (outside of the few large-scale vendors) is a tale of two types of designs: the IP designer who designs a series of processor or arithmetic units, and the system designer who assembles these designs.

Future technology directions

With obvious frequency and power limitations, the main design challenge is to use area to maximum advantage in both high-performance and very low-power designs. Here, we consider just those two design endpoints as targets in themselves. However, these endpoints can also provide insights into other intermediate designs.

High-performance architecture

Multiple cores, multiple threads per core, and large multilevel caches define today's high-speed commodity processors. Without exception, all instruction sets have SIMD support for multimedia applications. There is a convergence of computing and communication technologies, thanks to the Internet and the advent of various wireless broadband technologies.

As a result, on-die support for wired and wireless networking, and other security functions will be increasingly important. Because processor clock frequency is unlikely to increase dramatically, at least in the short term, designers will focus on improving the overall system performance with faster buses (such as PCI Express, HyperTransport, and Infiniband) and faster memory interfaces—dual data rate (DDR2 and DDR3) and fully buffered dual inline memory modules (FBDIMM).

Going beyond evolutionary progress requires the effective use of large degrees of concurrency or parallelism. Achieving success in these areas has not been easy. Although high-performance computing has always attracted attention, success is usually based on a few implementations targeted at special applications. Extending performance beyond these applications requires extensive software support, and the development of this support has proven difficult and time consuming. Software support for parallelism moves at a glacial rate when compared to the silicon roadmap.

Massively parallel processing (MPP) implementations illustrate this point. In the 1990s, many thought MPP would be the key to high-performance, cost-efficient computing. MPP was based on hundreds or thousands of interconnected processor and memory nodes. The interconnection was either direct (such as a grid or hypercube topology) or indirect (such as omega topology).

MPP implementations were, however, costly and had notable but limited success. The world's fastest computer—IBM's Blue-Gene prototype—has 8,192 dual core processors and achieves 36 Tflops (Linpack) at the expense of more than 100 kilowatts. The final version will be 8× larger with 64k processors and 32TB of memory and consume 84SkW.[31] General use of MPP failed because of cost and the limited parallelism it offered. Indeed, parallel applications, such as simulations and transaction processing, found cheaper solutions, such as grid computing. Moreover, applications that require the MPP

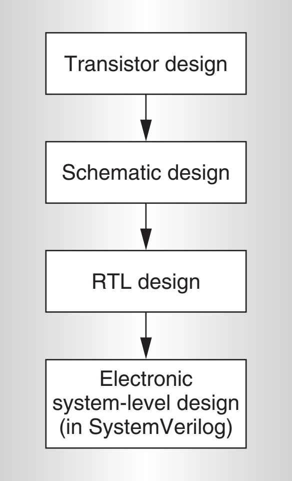

Figure 10. Productivity increases with higher abstraction levels.

Table 8. Energy capacity for each battery application.

Type	Energy capacity (mA-hr)	Lifetime (duty cycle)	At power (mW)
Rechargeable	10,000	50 hours (10 to 20% duty)	400 to 4
Two AAs (or AAA)	4,000	0.5 years (10 to 20% duty)	1 to 10
Button	40	5 years (always on)	0.001

communications bandwidth in their interconnection network are usually difficult to parallelize.

It's clear that high-performance system success is based more on software breakthroughs than simply on hardware. Restricting attention to smaller implementations, there are some promising architectural possibilities.

FPGAs. Field-programmable gate arrays seem to contradict the goals of area-time-power-optimized technology. Designers optimize FPGAs for implementation efficiency and reconfigurability; interconnects consume most of the die area. However, the regularity and programmability of these devices can support systolic or tessellated computation. When the application permits, it's possible to configure the computation into cells (clusters of combinational logic blocks) and then unroll and pipeline them across the 2D cellular array. Stream compilers, such as A Stream Compiler (ASC),[32] are now available for use with FPGAs. Researchers report excellent computational bandwidth using this technology across several diverse applications.

Grape. At the outer edge of performance is the Grape series of computers for solving the *N*-body problem in the simulation of galaxy formation.[33] Realizing almost 30 Tflops, Grape uses older technology in a desk-sized implementation. The unrolled and pipelined computation uses low-precision, pipelined, arithmetic units as cells. The computation, although straightforward and without branches, does require each cell to access every other cell (to determine gravitational force) in each time step. It's interesting to speculate whether it was possible to sue a stream compiler approach with larger arithmetic cells to realize significant performance across a useful body of applications.

Extremely low-power architecture (ELPA)

Ubiquitous computing is sometimes thought of as the third wave in computing, after mainframes and PCs. Ubiquitous computing seeks to connect together wearable computing devices, consumer electronics, PCs, and servers in wired and wireless networks.

Inevitably, wearable computing devices and most consumer electronics have to run on batteries. Table 8 shows the energy capacity for each battery application. For wearable computing applications, button batteries with a small amount of energy are the most attractive option.

To realize ubiquitous computing, the industry needs a new class of extremely low-power architectures (ELPAs), operating at microwatts (not watts or milliwatts).

So how do you design a processor with reasonable performance that consumes $O(\mu W)$? There are six obvious power-saving schemes:

- Assuming the clock rate remains constant, scaling alone lowers power by reducing parasitic wire and gate capacitance.
- Lowering frequency lowers power by a cubic relation, so a 10× reduced frequency should lower power by 1,000×. To regain lost performance, we might need 10× more logic and functional units, reducing the gain to 100×.
- Slower processors are better matched to memory speed, obviating the need for large caches and buffers. This results in a smaller core and improves the approach in the preceding description.
- Asynchronous clocking and logic eliminate clock power and reduce the number of state transitions. Indeed, unused logic paths and units dissipate no active power. Such economies could reduce active power by up to 10×, depending on the logic's configuration.
- Circuit techniques, such as adiabatic switching, adaptive body biasing, multiple V_T, can also reduce switching and leakage losses.[34,35]
- Designers should use an integrated power management solution—from application to operating system to hardware—to save power on all levels.[6,36]

Consider a StrongARM processor that operates at 160 MHz in a 0.35-µm process technology.[37] It consumes about 450 mW at 1.6 V supply voltage. Table 9 shows the relative power consumption in each functional unit.

Following the *ITRS*, if the same StrongARM processor operates at 0.5 V in a 22-nm process technology, the die size is more than 250 times smaller, but the parasitic wire and gate capacitance is only roughly 15× less. Assuming that the switching loss dominates the total power consumption, a scaled version of the StrongARM processor consumes roughly 2.8 mW.

However, unless we use a different process, the lower supply voltage and the resulting higher leakage imply relatively high static power, perhaps equal to the active power. This makes further significant power reductions impossible.

Assuming the availability of a power-optimized process technology (with relatively high V_T), we might be able to significantly reduce static power. Applying a 10× frequency reduction brings the active power close to our targeted power range, 2 to 5 μW. Eliminating the power and area associated with the clock, reducing the sizes and number of the caches, and reducing the number of memory management units also helps reduce the overall power. Sadly, all of this is very optimistic. As with the assumptions on static power, sources of power dissipation that were negligible in a current design would dominate microwatt designs.

Consider just one challenge to ELPA processors: I/O communications. Assume the entire computer system (processor plus memory) is implemented on a single chip; the chip still needs to communicate with the outside world (such as the Internet). A 2-pF I/O with a 0.5-V voltage swing at 20 MHz already consumes 10 μW. The ELPA challenges are formidable but not well addressed. Among the obvious problems are the following:

- The process technology must evolve in two directions: performance optimized (allowing higher static power) and power optimized with very low static power.
- In the long run, new communication and packaging technologies might be necessary to solving the power problem. These technologies include on-chip optical and wireless communication technologies, and SoP technologies that work with emerging nanoscale components.[15,38-40]
- Devising effective ELPA processor architectures will require a better understanding of designs that run at a very low speed (less than 20 MHz) to save power. Such designs would use a large number of data paths and functional units (area concurrency) to regain performance.

Table 9. Functional unit power dissipation in a StrongARM processor.

Functional unit	Power consumption (percentage)
Instruction cache	27
Instruction unit	18
Data cache	16
Clock	10
Instruction memory management unit	9
Execution unit	8
Data memory management unit	8
Miscellaneous	4

The current *ITRS* promises 100× circuit density and 10× clock speed in the next 15 years. As in the past, this roadmap is more accurate in predicting circuit density than clock speed, because it is much harder to foretell the future in microarchitecture. Nevertheless, it is very unlikely that clock speed will exceed 10× because of the power wall. Assuming architects can fully use the extra area without any major communication or other performance overheads, the increases in circuit density and clock speed represent up to 1,000× improvement in processor performance or a change of 10^{-6} in power consumption. How architects use these enabling technologies depends on the target application.

Processors for high-performance server should continue to be relatively cost insensitive. Because of the power and memory walls, these designs will achieve performance gains mainly by implementing multiple base processor cores with a large cache. Each base processor core should resemble current architectures that use a microarchitecture based on instruction- and thread-level parallelisms and that have SIMD extensions, where appropriate. An average server can easily contain many processors, forming a multiple-instruction multiple-data system ensemble with a configurable network similar to today's supercomputers. Such a server system can efficiently run thousands of threads, but each thread can only be a few times faster than today's server thread. To be useful to users, such a situation demands a new software paradigm. One big improvement should be the increased attention to very high integrity computational requirements.

If they can overcome power and design difficulties, SoCs promise to usher in the ubiquitous computing era. Designers could integrate SoCs into appliances, communications devices, notebooks, and watch-type wearable devices. Such systems would have to remain on at all times; be secure and available; use very low power; and work with wireless networks having communication bandwidths from 1 Kbps to 100 Mbps. SoC designs have many frequency-power design points, from 100 mW with a 1-ns cycle time to 1 mW with a 100-ns cycle time. Using several base or core processor designs coupled to a variety of signal processors and RAM or ROM, the SoC presents the ultimate challenge in system architecture and design. Such a design paradigm would require new system products and concepts; new interconnect technologies; and new IP management and design tools. They might indeed lead to many design and product opportunities on the road ahead. MICRO

References

1. M.J. Flynn, P. Hung, and K.W. Rudd, "Deep-Submicron Microprocessor Design Issues," *IEEE Micro*, vol. 19, no. 4, July-Aug. 1999, pp. 11-22.
2. *International Technology Roadmap for Semiconductors*, Semiconductor Industry Assoc., 2003; http://public.itrs.net/.
3. M. Baron, "MPU EDA to face Complexity," *Microprocessor Report*, 12 July 2004.
4. P. Glaskowsky, "Putting Prescott in Perspective," *Microprocessor Report*, 17 Feb. 2004.
5. R. Kalla et al., "IBM Power5 Chip: A Dual-Core Multithreaded Processor," *IEEE Micro*, vol. 24, no. 2, Mar.-Apr. 2004, pp. 40-47.
6. C. McNairy and R. Bhatia, "Montecito: The Next Product in the Itanium Processor Family," *Hot Chips 16 Conf. Record*; http://www.hotchips.org/archives/hc16/.
7. P. Kongetira, "A 32-way Multithreaded SPARC Processor," *Hot Chips 16 Conf. Record*; http://www.hotchips.org/archives/hc16/.
8. P. Glaskowsky, "AMD Athlon 64: The Game's On," *Microprocessor Report*, 14 Oct. 2003.
9. K. Krewell, "Dothan by the Numbers," *Microprocessor Report*, 24 May 2004.
10. K. Krewell, "Intel's PC Roadmap Sees Double," *Microprocessor Report*, 1 June 2004.
11. K. Krewell, "AMD vs. Intel in Dual-Core Duel," *Microprocessor Report*, 6 July 2004.
12. P.N. Glaskowsky, "Servers Take Giant Steps," *Microprocessor Report*, 9 Feb. 2004.
13. T. Starnes, *Programmable Microcomponent Forecast through 2006*, Gartner, Feb. 2003.
14. B. Lewis, *Microprocessor Cores Shape Future SLI/SOC Market*, Gartner, 2002.
15. Henry Chang et al., *Surviving the SOC Revolution*, Kluwer Academic Publishers, 1999.
16. R. Tummala, "SOP: Microelectronic Systems Packaging Technology for the 21st Century," *Advanced Microelectronics*, vol. 26, no. 3, May-June 1999, pp. 29-37.
17. S. Furber, *ARM: System-on-Chip Architecture*, 2nd ed., Addison-Wesley, 2000.
18. M. Evers, "Low Power AMD Athlon 64 and AMD Opteron Processors," *Hot Chips 16 Conf. Record*; http://www.hotchips.org/archives/hc16/.
19. V. Zyuban and P. Strenski, "Unified Methodology for Resolving Power-Performance Tradeoffs at the Microarchitectural and Circuit Levels," *Proc. Int'l Symp. Low-Power Electronic Design* (ISLPED 02), IEEE Press, 2002, pp. 166-171.
20. F. Kobayashi et al., "Hardware Technology for Hitachi M-880 Processor Group," *41st Electronic Components and Technologies Conf.*, IEEE Press, 1991, pp. 693-703.
21. J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1984.
22. D. Harris and M.A. Horowitz, "Skew-Tolerant Domino Circuits," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, Nov. 1997, pp. 1702-1711.
23. V. Agarwal et al., "Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures," *27th Ann. Int'l Symp. Computer Architecture* (ISCA 00), IEEE CS Press, 2000, pp. 248-259.
24. M.S. Hrishikesh et al., "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4

Inverter Delays," *Proc. 29th Ann. Int'l Symp Computer Architecture* (ISCA 02), IEEE CS Press, 2002, pp. 14-24.

25. P.K. Dubey and M.J. Flynn, "Optimal Pipelining," *J. Parallel and Distributed Computing*, vol. 8, no. 1, Jan. 1990, pp. 10-19.
26. M.J. Flynn, *Computer Architecture: Pipelined and Parallel Processor Design*, Jones and Bartlett Publishers, 1995.
27. G. McFarland, "CMOS Technology Scaling and Its Impact on Cache Delay," PhD thesis, Stanford Univ., June 1997.
28. J. Hines, *Midyear 2003 Semiconductor Manufacturing Market: Wafer Foundry*, Gartner, Aug. 2003.
29. IEEE Std. 1149.1-1990, *IEEE Standard Test Access Port and Boundary-Scan Architecture*, 1990.
30. A. L. Crouch, *Design-For-Test for Digital ICs and Embedded Core Systems*, Prentice-Hall, 1999.
31. N.R. Adiga et al. (The BlueGene/L Team), "An Overview of the BlueGene/L Supercomputer," *Proc. 2002 ACM/IEEE Conf. on Supercomputing* (SC 02), IEEE CS Press, 2002, pp. 1-22.
32. O. Mencer et al., "Design Space Exploration with A Stream Compiler," *IEEE Int'l Conf. on Field Programmable Technology* (FPT 03), IEEE Press, 2003, pp. 270-277.
33. J. Kakino et al., "A 29.5 Tflops Simulation of Planetesimals in Uranus-Neptune Region on GRAPE-6," *Proc. 2002 ACM/IEEE Conf. on Supercomputing* (SC 02), IEEE Press, 2002, pp. 1-14.
34. W. Athas et al., "A Low-Power Digital Systems Based on Adiabatic Switching Principles," *IEEE J. Solid-State Circuits*, vol. 2, no. 12, Dec. 1994, pp. 398-407.
35. J. Tschanz et al., "Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variation on Microprocessor Frequency and Leakage," *Digest of Tech. Papers 2002 IEEE Int'l Solid-State Circuits Conf.* (ISSCC 02), IEEE Press, 2002, pp. 412-413.
36. *Advanced Configuration and Power Interface Specification*, revision 2.0c, Compaq Computer Corp., Intel Corp., Microsoft Corp., Phoenix Technologies Ltd., and Toshiba Corp., 25 Aug. 2003.
37. J. Montanaro et al., "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, Nov. 1996, pp. 1703-1714.
38. D. Mizoguchi et al., "A 1.2Gb/s/pin Wireless Superconnect Based on Inductive Inter-chip Signaling (IIS)," *Digest of Tech. Papers 2004 IEEE Int'l Solid-State Circuits Conf.* (ISSCC 04), IEEE Press, 2004, pp. 142-143.
39. A. Liu et al., "A High-Speed Silicon Optical Modulator Based on a Metal-Oxide-Semiconductor Capture," *Nature*, vol. 427, no. 6975, Feb. 2004, pp. 615-618.
40. D.A.B. Miller, "Physical Reasons for Optical Interconnection," *Int'l J. Optoelectronics*, vol. 11, no. 3, May-June 1997, pp. 155-168.

Michael J. Flynn is professor emeritus of electrical engineering at Stanford University. He began his engineering career at IBM as a designer of mainframe computers; at Stanford he founded, and continues research at, the Architecture and Arithmetic group. Flynn has a BS from Manhattan College, an MS from Syracuse University, and a PhD from Purdue University. He is a Fellow of the IEEE and ACM.

Patrick Hung is consulting assistant professor of electrical engineering at Stanford University. His research interests include microprocessor design and system-level computer-aided design tools. Hung has a BS from the University of Hong Kong, and an MS and a PhD from Stanford University. He is a member of the IEEE and ACM.

Direct questions and comments about this article to Patrick Hung, Stanford University; hung@arith.stanford.edu.