

Energy efficiency is the new fundamental limiter of processor performance, way beyond numbers of processors.

BY SHEKHAR BORKAR AND ANDREW A. CHIEN

The Future of Microprocessors

MICROPROCESSORS—SINGLE-CHIP COMPUTERS—are the building blocks of the information world. Their performance has grown 1,000-fold over the past 20 years, driven by transistor speed and energy scaling, as well as by microarchitecture advances that exploited the transistor density gains from Moore’s Law. In the

next two decades, diminishing transistor-speed scaling and practical energy limits create new challenges for continued performance scaling. As a result, the frequency of operations will increase slowly, with energy the key limiter of performance, forcing designs to use large-scale parallelism, heterogeneous cores, and accelerators to achieve performance and energy efficiency. Software-hardware partnership to achieve efficient data orchestration is increasingly critical in the drive toward energy-proportional computing.

Our aim here is to reflect and project the macro trends shaping the future of microprocessors and sketch in broad strokes where processor design is going. We enumerate key research challenges and suggest promising research directions. Since dramatic changes are coming, we also seek to inspire the research community to in-

vent new ideas and solutions address how to sustain computing’s exponential improvement.

Microprocessors (see Figure 1) were invented in 1971,²⁸ but it’s difficult today to believe any of the early inventors could have conceived their extraordinary evolution in structure and use over the past 40 years. Microprocessors today not only involve complex micro-

» key insights

- **Moore’s Law continues but demands radical changes in architecture and software.**
- **Architectures will go beyond homogeneous parallelism, embrace heterogeneity, and exploit the bounty of transistors to incorporate application-customized hardware.**
- **Software must increase parallelism and exploit heterogeneous and application-customized hardware to deliver performance growth.**

architectures and multiple execution engines (cores) but have grown to include all sorts of additional functions, including floating-point units, caches, memory controllers, and media-processing engines. However, the defining characteristics of a microprocessor remain—a single semiconductor chip embodying the primary computation (data transformation) engine in a computing system.

Because our own greatest access and insight involves Intel designs and data, our graphs and estimates draw heavily on them. In some cases, they may not be representative of the entire industry but certainly represent a large fraction. Such a forthright view, solidly grounded, best supports our goals for this article.

20 Years of Exponential Performance Gains

For the past 20 years, rapid growth in microprocessor performance has been enabled by three key technology drivers—transistor-speed scaling, core microarchitecture techniques, and cache memories—discussed in turn in the following sections:

Transistor-speed scaling. The MOS transistor has been the workhorse for decades, scaling in performance by nearly five orders of magnitude and providing the foundation for today’s unprecedented compute performance. The basic recipe for technology scaling was laid down by Robert N. Dennard of IBM¹⁷ in the early 1970s and followed for the past three decades. The scaling recipe calls for reducing transistor

dimensions by 30% every generation (two years) and keeping electric fields constant everywhere in the transistor to maintain reliability. This might sound simple but is increasingly difficult to continue for reasons discussed later. Classical transistor scaling provided three major benefits that made possible rapid growth in compute performance.

First, the transistor dimensions are scaled by 30% (0.7x), their area shrinks 50%, doubling the transistor density every technology generation—the fundamental reason behind Moore’s Law. Second, as the transistor is scaled, its performance increases by about 40% (0.7x delay reduction, or 1.4x frequency increase), providing higher system performance. Third, to keep the electric field constant, supply voltage is reduced by 30%, reducing energy by 65%, or power (at 1.4x frequency) by 50% (active power = CV^2f). Putting it all together, in every technology generation transistor integration doubles, circuits are 40% faster, and system power consumption (with twice as many transistors) stays the same. This serendipitous scaling (almost too good to be true) enabled three-orders-of-magnitude increase in microprocessor performance over the past 20 years. Chip architects exploited transistor density to create complex architectures and transistor speed to increase frequency, achieving it all within a reasonable power and energy envelope.

Core microarchitecture techniques. Advanced microarchitectures have deployed the abundance of transistor-integration capacity, employing a dizzying array of techniques, including pipelining, branch prediction, out-of-order execution, and speculation, to deliver ever-increasing performance. Figure 2 outlines advances in microarchitecture, showing increases in die area and performance and energy efficiency (performance/watt), all normalized in the same process technology. It uses characteristics of Intel microprocessors (such as 386, 486, Pentium, Pentium Pro, and Pentium 4), with performance measured by benchmark SpecInt (92, 95, and 2000 representing the current benchmark for the era) at each data point. It compares each microarchitecture advance with a design without the ad-

Figure 1. Evolution of Intel microprocessors 1971–2009.

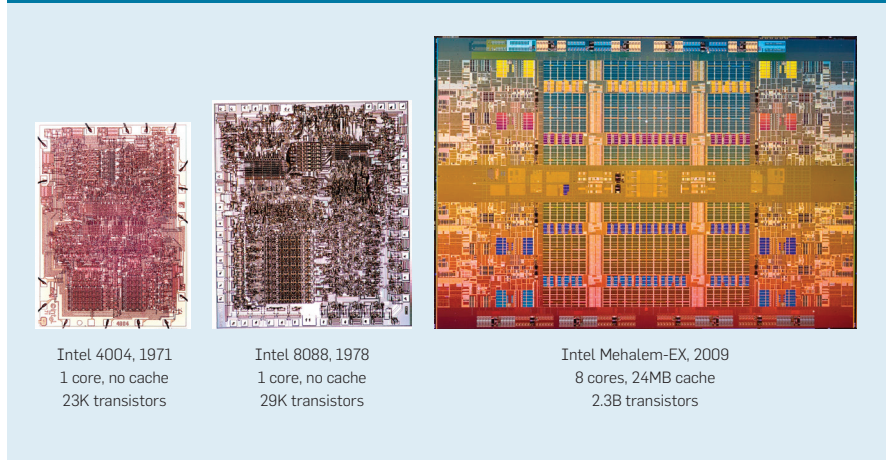
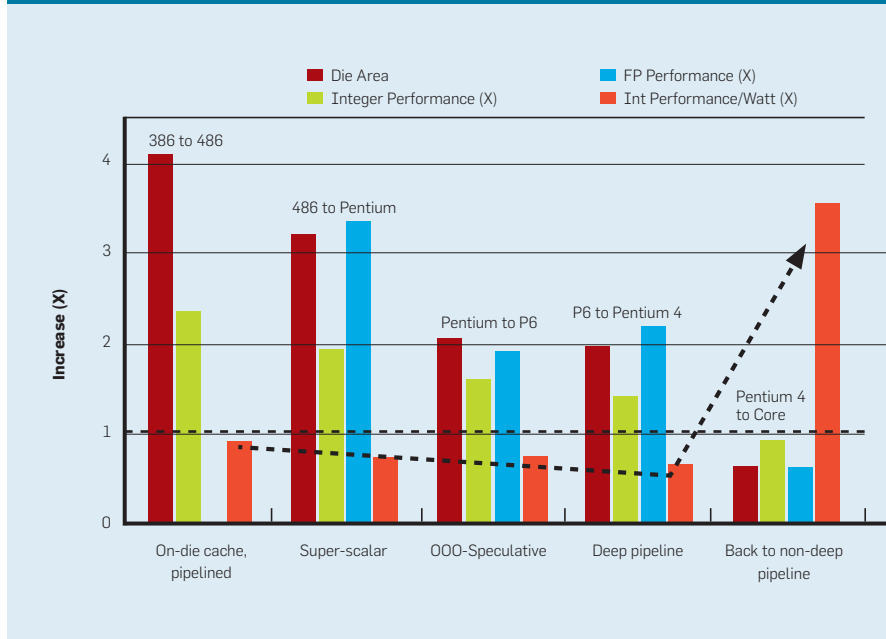


Figure 2. Architecture advances and energy efficiency.



vance (such as introducing an on-die cache by comparing 486 to 386 in 1 μ technology and superscalar microarchitecture of Pentium in 0.7 μ technology with 486).

This data shows that on-die caches and pipeline architectures used transistors well, providing a significant performance boost without compromising energy efficiency. In this era, superscalar, and out-of-order architectures provided sizable performance benefits at a cost in energy efficiency. Of these architectures, deep-pipelined design seems to have delivered the lowest performance increase for the same area and power increase as out-of-order and speculative design, incurring the greatest cost in energy efficiency. The term “deep pipelined architecture” describes deeper pipeline, as well as other circuit and microarchitectural techniques (such as trace cache and self-resetting domino logic) employed to achieve even higher frequency. Evident from the data is that reverting to a non-deep pipeline reclaimed energy efficiency by dropping these expensive and inefficient techniques.

When transistor performance increases frequency of operation, the performance of a well-tuned system generally increases, with frequency subject to the performance limits of other parts of the system. Historically, microarchitecture techniques exploiting the growth in available transistors have delivered performance increases empirically described by Pollack’s Rule,³² whereby performance increases (when not limited by other parts of the system) as the square root of the number of transistors or area of a processor (see Figure 3). According to Pollack’s Rule, each new technology generation doubles the number of transistors on a chip, enabling a new microarchitecture that delivers a 40% performance increase. The faster transistors provide an additional 40% performance (increased frequency), almost doubling overall performance within the same power envelope (per scaling theory). In practice, however, implementing a new microarchitecture every generation is difficult, so microarchitecture gains are typically less. In recent microprocessors, the increasing drive for energy efficiency has

caused designers to forego many of these microarchitecture techniques.

As Pollack’s Rule broadly captures area, power, and performance trade-offs from several generations of microarchitecture, we use it as a rule of thumb to estimate single-thread performance in various scenarios throughout this article.

Cache memory architecture. Dynamic memory technology (DRAM) has also advanced dramatically with Moore’s Law over the past 40 years but with different characteristics. For example, memory density has doubled nearly every two years, while performance has improved more slowly (see Figure 4a). This slower improvement in cycle time has produced a memory bottleneck that could reduce a system’s overall performance. Figure 4b outlines the increasing speed disparity, growing from 10s to 100s of processor clock cycles per memory access. It has lately flattened out due to the flattening of processor clock frequency.

Unaddressed, the memory-latency gap would have eliminated and could still eliminate most of the benefits of processor improvement.

The reason for slow improvement of DRAM speed is practical, not technological. It’s a misconception that DRAM technology based on capacitor storage is inherently slower; rather, the memory organization is optimized for density and lower cost, making it slower. The DRAM market has demanded large capacity at minimum cost over speed, depending on small and fast caches on the microprocessor die to emulate high-performance memory by providing the necessary bandwidth and low latency based on data locality. The emergence of sophisticated, yet effective, memory hierarchies allowed DRAM to emphasize density and cost over speed. At first, processors used a single level of cache, but, as processor speed increased, two to three levels of cache hierarchies were introduced to span the growing speed gap between

Figure 3. Increased performance vs. area in the same process technology follows Pollack’s Rule.

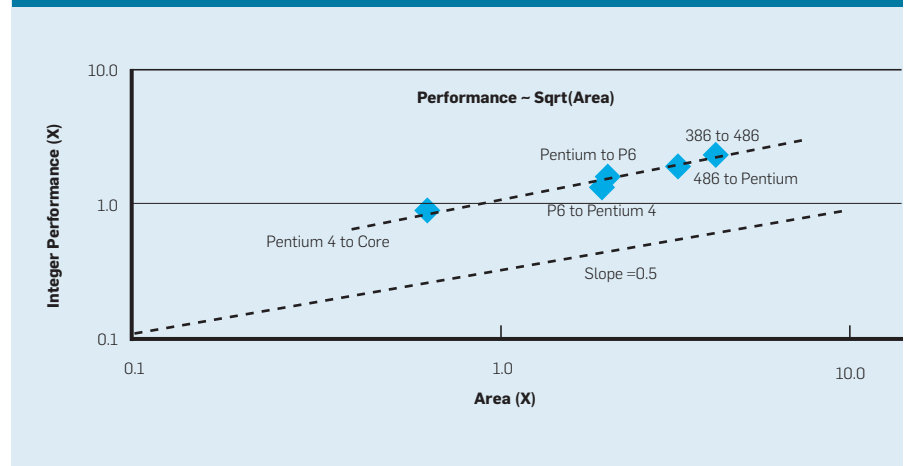
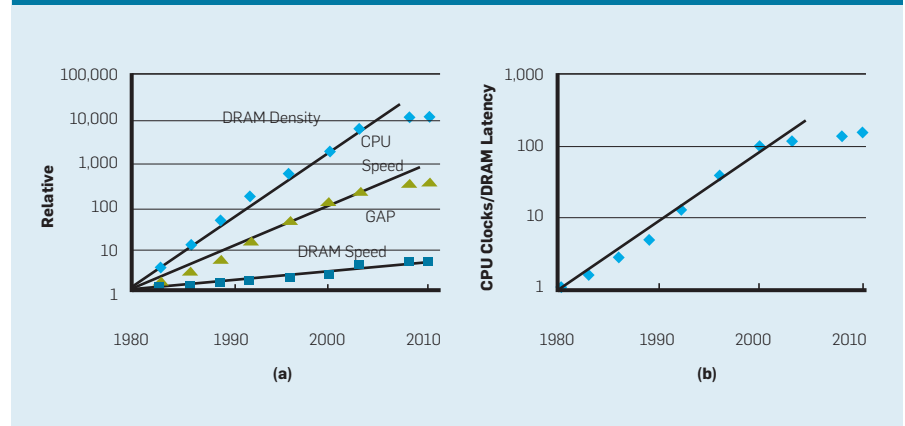


Figure 4. DRAM density and performance, 1980–2010.



processor and memory.^{33,37} In these hierarchies, the lowest-level caches were small but fast enough to match the processor's needs in terms of high bandwidth and low latency; higher levels of the cache hierarchy were then optimized for size and speed.

Figure 5 outlines the evolution of on-die caches over the past two decades, plotting cache capacity (a) and percentage of die area (b) for Intel microprocessors. At first, cache sizes increased slowly, with decreasing die

area devoted to cache, and most of the available transistor budget was devoted to core microarchitecture advances. During this period, processors were probably cache-starved. As energy became a concern, increasing cache size for performance has proven more energy efficient than additional core-microarchitecture techniques requiring energy-intensive logic. For this reason, more and more transistor budget and die area are allocated in caches.

The transistor-scaling-and-micro-

architecture-improvement cycle has been sustained for more than two decades, delivering 1,000-fold performance improvement. How long will it continue? To better understand and predict future performance, we decouple performance gain due to transistor speed and microarchitecture by comparing the same microarchitecture on different process technologies and new microarchitectures with the previous ones, then compound the performance gain.

Figure 6 divides the cumulative 1,000-fold Intel microprocessor performance increase over the past two decades into performance delivered by transistor speed (frequency) and due to microarchitecture. Almost two-orders-of-magnitude of this performance increase is due to transistor speed alone, now leveling off due to the numerous challenges described in the following sections.

The Next 20 Years

Microprocessor technology has delivered three-orders-of-magnitude performance improvement over the past two decades, so continuing this trajectory would require at least 30x performance increase by 2020. Micropro-

Figure 5. Evolution of on-die caches.

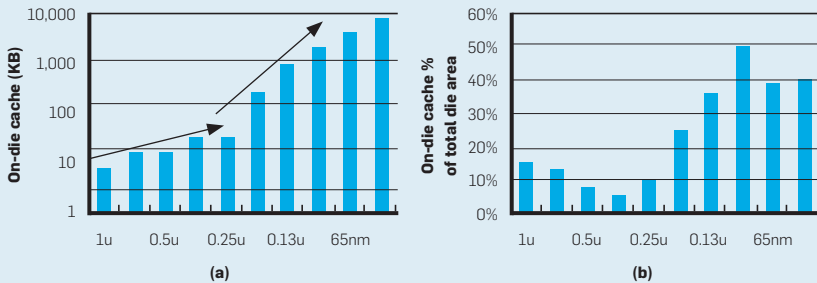


Figure 6. Performance increase separated into transistor speed and microarchitecture performance.

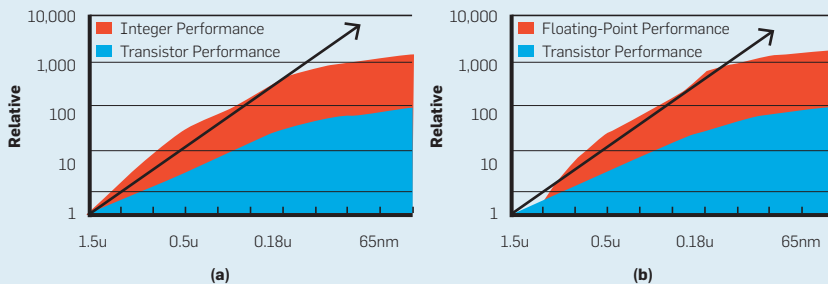


Figure 7. Unconstrained evolution of a microprocessor results in excessive power consumption.

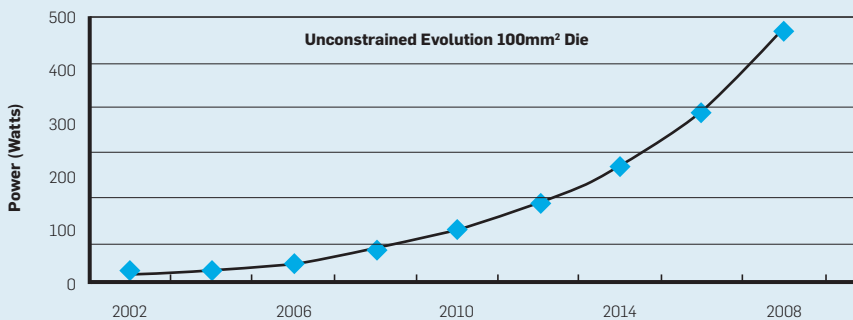


Table 1. New technology scaling challenges.

Decreased transistor scaling benefits: Despite continuing miniaturization, little performance improvement and little reduction in switching energy (decreasing performance benefits of scaling) [ITRS].

Flat total energy budget: package power and mobile/embedded computing drives energy-efficiency requirements.

Table 2. Ongoing technology scaling.

Increasing transistor density (in area and volume) and count: through continued feature scaling, process innovations, and packaging innovations.

Need for increasing locality and reduced bandwidth per operation: as performance of the microprocessor increases, and the data sets for applications continue to grow.

cessor-performance scaling faces new challenges (see Table 1) precluding use of energy-inefficient microarchitecture innovations developed over the past two decades. Further, chip architects must face these challenges with an ongoing industry expectation of a 30x performance increase in the next decade and 1,000x increase by 2030 (see Table 2).

As the transistor scales, supply voltage scales down, and the threshold voltage of the transistor (when the transistor starts conducting) also scales down. But the transistor is not a perfect switch, leaking some small amount of current when turned off, increasing exponentially with reduction in the threshold voltage. In addition, the exponentially increasing transistor-integration capacity exacerbates the effect; as a result, a substantial portion of power consumption is due to leakage. To keep leakage under control, the threshold voltage cannot be lowered further and, indeed, must increase, reducing transistor performance.¹⁰

As transistors have reached atomic dimensions, lithography and variability pose further scaling challenges, affecting supply-voltage scaling.¹¹ With limited supply-voltage scaling, energy and power reduction is limited, adversely affecting further integration of transistors. Therefore, transistor-integration capacity will continue with scaling, though with limited performance and power benefit. The challenge for chip architects is to use this integration capacity to continue to improve performance.

Package power/total energy consumption limits number of logic transistors. If chip architects simply add more cores as transistor-integration capacity becomes available and operate the chips at the highest frequency the transistors and designs can achieve, then the power consumption of the chips would be prohibitive (see Figure 7). Chip architects must limit frequency and number of cores to keep power within reasonable bounds, but doing so severely limits improvement in microprocessor performance.

Consider the transistor-integration capacity affordable in a given power envelope for reasonable die size. For regular desktop applications the pow-

Death of 90/10 Optimization, Rise of 10×10 Optimization

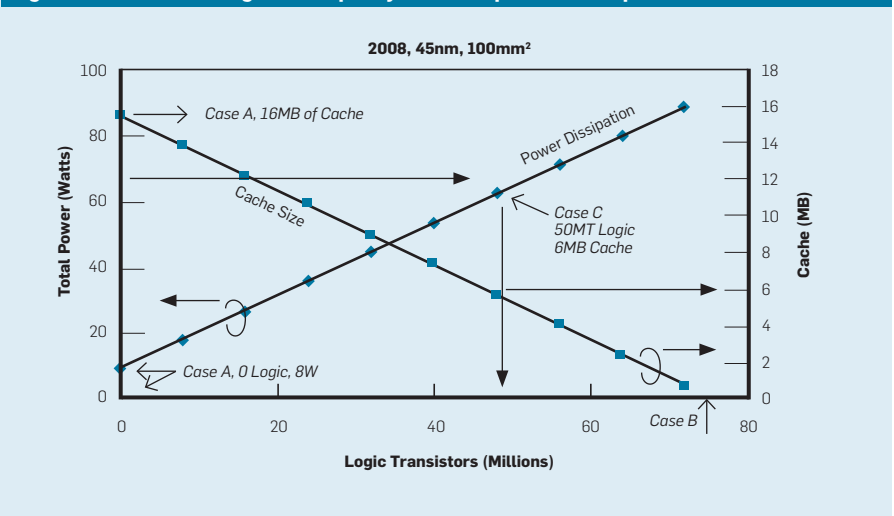
Traditional wisdom suggests investing maximum transistors in the 90% case, with the goal of using precious transistors to increase single-thread performance that can be applied broadly. In the new scaling regime typified by slow transistor performance and energy improvement, it often makes no sense to add transistors to a single core as energy efficiency suffers. Using additional transistors to build more cores produces a limited benefit—increased performance for applications with thread parallelism. In this world, 90/10 optimization no longer applies. Instead, optimizing with an accelerator for a 10% case, then another for a different 10% case, then another 10% case can often produce a system with better overall energy efficiency and performance. We call this “10×10 optimization,”¹⁴ as the goal is to attack performance as a set of 10% optimization opportunities—a different way of thinking about transistor cost, operating the chip with 10% of the transistors active—90% inactive, but a different 10% at each point in time.

Historically, transistors on a chip were expensive due to the associated design effort, validation and testing, and ultimately manufacturing cost. But 20 generations of Moore’s Law and advances in design and validation have shifted the balance. Building systems where the 10% of the transistors that can operate within the energy budget are configured optimally (an accelerator well-suited to the application) may well be the right solution. The choice of 10 cases is illustrative, and a 5×5, 7×7, 10×10, or 12×12 architecture might be appropriate for a particular design.

er envelope is around 65 watts, and the die size is around 100mm². Figure 8 outlines a simple analysis for 45nm process technology node; the x-axis is the number of logic transistors integrated on the die, and the two y-axes are the amount of cache that would fit and the power the die would consume. As the number of logic transistors on the die increases (x-axis), the size of the cache decreases, and power dissipation increases. This analysis assumes average activity factor for logic and

cache observed in today’s microprocessors. If the die integrates no logic at all, then the entire die could be populated with about 16MB of cache and consume less than 10 watts of power, since caches consume less power than logic (Case A). On the other hand, if it integrates no cache at all, then it could integrate 75 million transistors for logic, consuming almost 90 watts of power (Case B). For 65 watts, the die could integrate 50 million transistors for logic and about 6MB of cache (Case C).

Figure 8. Transistor integration capacity at a fixed power envelope.



This design point matches the dual-core microprocessor on 45nm technology (Core2 Duo), integrating two cores of 25 million transistors each and 6MB of cache in a die area of about 100mm².

If this analysis is performed for future technologies, assuming (our best estimates) modest frequency increase 15% per generation, 5% reduction in supply voltage, and 25% reduction of

capacitance, then the results will be as they appear in Table 1. Note that over the next 10 years we expect increased total transistor count, following Moore's Law, but logic transistors increase by only 3x and cache transistors increase more than 10x. Applying Pollack's Rule, a single processor core with 150 million transistors will provide only about 2.5x microarchitecture performance improvement over today's 25-million-transistor core, well shy of our 30x goal, while 80MB of cache is probably more than enough for the cores (see Table 3).

The reality of a finite (essentially fixed) energy budget for a microprocessor must produce a qualitative shift in how chip architects think about architecture and implementation. First, energy-efficiency is a key metric for these designs. Second, energy-proportional computing must be the ultimate goal for both hardware architecture and software-application design. While this ambition is noted in macro-scale computing in large-scale data centers,⁵ the idea of micro-scale energy-proportional computing in microprocessors is even more challenging. For microprocessors operating within a finite energy budget, energy efficiency corresponds directly to higher performance, so the quest for extreme energy efficiency is the ultimate driver for performance.

In the following sections, we outline key challenges and sketch potential approaches. In many cases, the challenges are well known and the subject of significant research over many years. In all cases, they remain critical but daunting for the future of microprocessor performance:

Organizing the logic: Multiple cores and customization. The historic measure of microprocessor capability is the single-thread performance of a traditional core. Many researchers have observed that single-thread performance has already leveled off, with only modest increases expected in the coming decades. Multiple cores and customization will be the major drivers for future microprocessor performance (total chip performance). Multiple cores can increase computational throughput (such as a 1x–4x increase could result from four cores), and customization can reduce execution la-

Figure 9. Three scenarios for integrating 150-million logic transistors into cores.

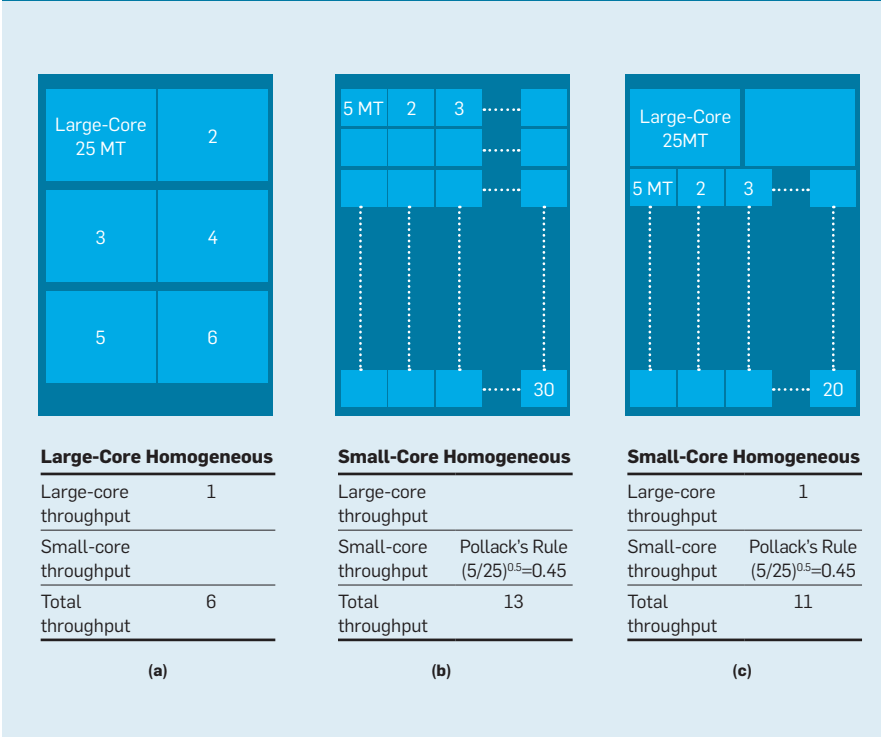


Figure 10. A system-on-a-chip from Texas Instruments.

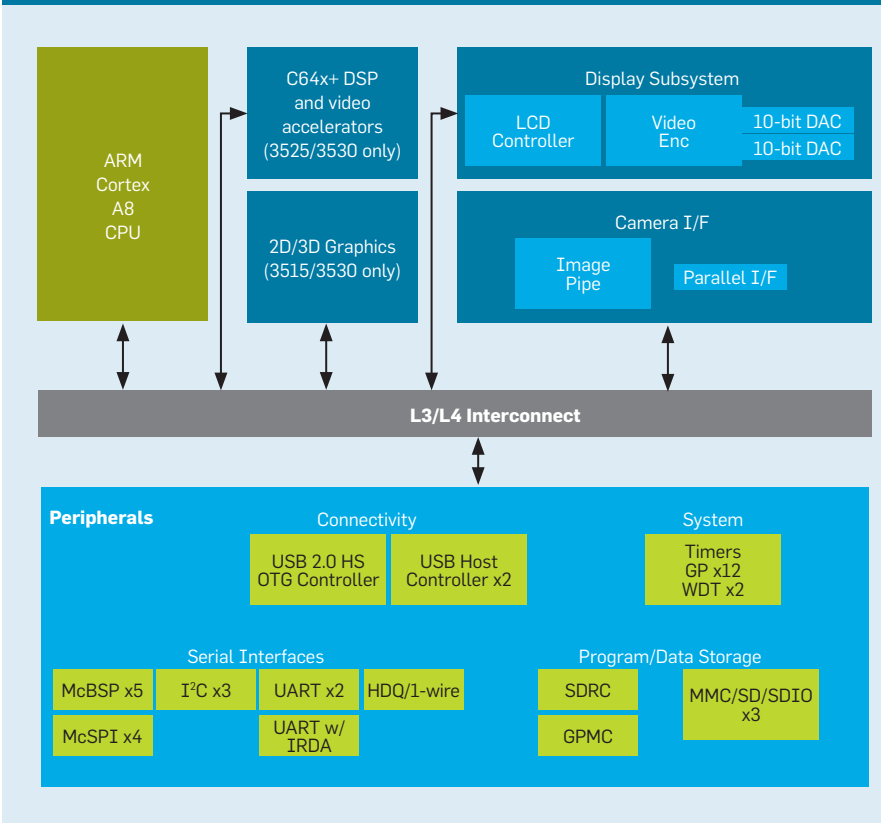


Table 3. Extrapolated transistor integration capacity in a fixed power envelope.

Year	Logic Transistors (Millions)	Cache MB
2008	50	6
2014	100	25
2018	150	80

tency. Clearly, both techniques—multiple cores and customization—can improve energy efficiency, the new fundamental limiter to capability.

Choices in multiple cores. Multiple cores increase computational throughput by exploiting Moore’s Law to replicate cores. If the software has no parallelism, there is no performance benefit. However, if there is parallelism, the computation can be spread across multiple cores, increasing overall computational performance (and reducing latency). Extensive research on how to organize such systems dates to the 1970s.^{29,39}

Industry has widely adopted a multicore approach, sparking many questions about number of cores and size/power of each core and how they coordinate.^{6,36} But if we employ 25-million-transistor cores (circa 2008), the 150-million-logic-transistor budget expected in 2018 gives 6x potential throughput improvement (2x from frequency and 3x from increased logic transistors), well short of our 30x goal. To go further, chip architects must consider more radical options of smaller cores in greater numbers, along with innovative ways to coordinate them.

Looking to achieve this vision, consider three potential approaches to deploying the feasible 150 million logic transistors, as in Table 1. In Figure 9, option (a) is six large cores (good single-thread performance, total potential throughput of six); option (b) is 30 smaller cores (lower single-thread performance, total potential throughput of 13); and option (c) is a hybrid approach (good single-thread performance for low parallelism, total potential throughput of 11).

Many more variations are possible on this spectrum of core size and num-

ber of cores, and the related choices in a multicore processor with uniform instruction set but heterogeneous implementation are an important part of increasing performance within the transistor budget and energy envelope.

Choices in hardware customization. Customization includes fixed-function accelerators (such as media codecs, cryptography engines, and compositing engines), programmable accelerators, and even dynamically customizable logic (such as FPGAs and other dynamic structures). In general, customization increases computational performance by exploiting hardwired or customized computation units, customized wiring/interconnect for data movement, and reduced instruction-sequence overheads at some cost in generality. In addition, the level of parallelism in hardware can be customized to match the precise needs of the computation; computation benefits from hardware customization only when it matches the specialized hardware structures. In some cases, units hardwired to a particular data representation or computational algorithm can achieve 50x–500x greater energy efficiency than a general-purpose register organization. Two studies^{21,22} of a media encoder and TCP offload engine illustrate the large energy-efficiency improvement that is possible.

Due to battery capacity and heat-dissipation limits, for many years energy has been the fundamental limiter for computational capabil-

ity in smartphone system-on-a-chip (SoC). As outlined in Figure 10, such an SoC might include as many as 10 to 20 accelerators to achieve a superior balance of energy efficiency and performance. This example could also include graphics, media, image, and cryptography accelerators, as well as support for radio and digital signal processing. As one might imagine, one of these blocks could be a dynamically programmable element (such as an FPGA or a software-programmable processor).

Another customization approach constrains the types of parallelism that can be executed efficiently, enabling a simpler core, coordination, and memory structures; for example, many CPUs increase energy efficiency by restricting memory access structure and control flexibility in single-instruction, multiple-data or vector (SIMD) structures,^{1,2} while GPUs encourage programs to express structured sets of threads that can be aligned and executed efficiently.^{26,30} This alignment reduces parallel coordination and memory-access costs, enabling use of large numbers of cores and high peak performance when applications can be formulated with a compatible parallel structure. Several microprocessor manufacturers have announced future mainstream products that integrate CPUs and GPUs.

Customization for greater energy or computational efficiency is a long-standing technique, but broad adop-

Table 4. Logic organization challenges, trends, directions.

Challenge	Near-Term	Long-Term
Integration and memory model	I/O-based interaction, shared memory spaces, explicit coherence management	Intelligent, automatic data movement among heterogeneous cores, managed by software-hardware partnership
Software transparency	Explicit partition and mapping, virtualization, application management	Hardware-based state adaptation and software-hardware partnership for management
Lower-power cores	Heterogeneous cores, vector extensions, and GPU-like techniques to reduce instruction- and data-movement cost	Deeper, explicit storage hierarchy within the core; integrated computation in registers
Energy management	Hardware dynamic voltage scaling and intelligent adaptive management, software core selection and scheduling	Predictive core scheduling and selection to optimize energy efficiency and minimize data movement
Accelerator variety	Increasing variety, library-based encapsulation (such as DX and OpenGL) for specific domains	Converged accelerators in a few application categories and increasing open programmability for the accelerators

tion has been slowed by continued improvement in microprocessor single-thread performance. Developers of software applications had little incentive to customize for accelerators that might be available on only a fraction of the machines in the field and for which

the performance advantage might soon be overtaken by advances in the traditional microprocessor. With slowing improvement in single-thread performance, this landscape has changed significantly, and for many applications, accelerators may be the only

path toward increased performance or energy efficiency (see Table 4). But such software customization is difficult, especially for large programs (see the sidebar “Decline of 90/10 Optimization, Rise of 10x10 Optimization”).

Orchestrating data movement: Memory hierarchies and interconnects. In future microprocessors, the energy expended for data movement will have a critical effect on achievable performance. Every nano-joule of energy used to move data up and down the memory hierarchy, as well as to synchronize across and data between processors, takes away from the limited budget, reducing the energy available for the actual computation. In this context, efficient memory hierarchies are critical, as the energy to retrieve data from a local register or cache is far less than the energy to go to DRAM or to secondary storage. In addition, data must be moved between processing units efficiently, and task placement and scheduling must be optimized against an interconnection network with high locality. Here, we examine energy and power associated with data movement on the processor die.

Today’s processor performance is on the order of 100Giga-op/sec, and a 30x increase over the next 10 years would increase this performance to 3Tera-op/sec. At minimum, this boost requires 9Tera-operands or 64b x 9Tera-operands (or 576Tera-bits) to be moved each second from registers or memory to arithmetic logic, consuming energy.

Figure 11(a) outlines typical wire delay and energy consumed in moving one bit of data on the die. If the operands move on average 1mm (10% of die size), then at the rate of 0.1pJ/bit, the 576Tera-bits/sec of movement consumes almost 58 watts with hardly any energy budget left for computation. If most operands are kept local to the execution units (such as in register files) and the data movement is far less than 1mm, on, say, the order of only 0.1mm, then the power consumption is only around 6 watts, allowing ample energy budget for the computation.

Cores in a many-core system are typically connected through a network-on-a-chip to move data around the cores.⁴⁰ Here, we examine the ef-

Figure 11. On-die interconnect delay and energy (45nm).

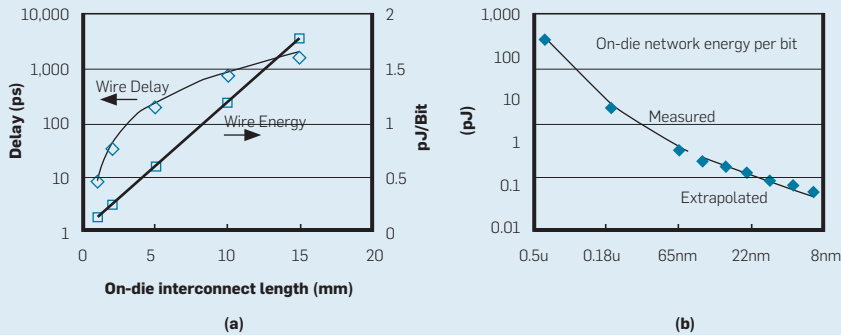


Figure 12. Hybrid switching for network-on-a-chip.

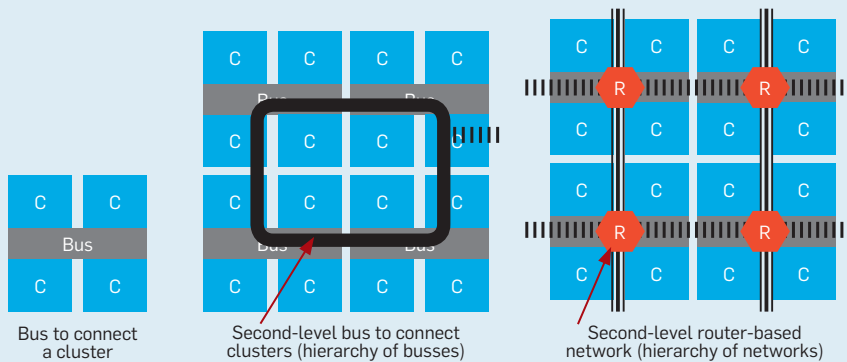


Table 5. Data movement challenges, trends, directions.

Challenge	Near-Term	Long-Term
Parallelism	Increased parallelism	Heterogeneous parallelism and customization, hardware/runtime placement, migration, adaptation for locality and load balance
Data Movement/ Locality	More complex, more exposed hierarchies; new abstractions for control over movement and “snooping”	New memory abstractions and mechanisms for efficient vertical data locality management with low programming effort and energy
Resilience	More aggressive energy reduction; compensated by recovery for resilience	Radical new memory technologies (new physics) and resilience techniques
Energy Proportional Communication	Fine-grain power management in packet fabrics	Exploitation of wide data, slow clock, and circuit-based techniques
Reduced Energy	Low-energy address translation	Efficient multi-level naming and memory-hierarchy management

fect of such a network on power consumption. Figure 11(b) shows the energy consumed in moving a bit across a hop in such a network, measured in historic networks, and extrapolated into the future from previous assumptions. If only 10% of the operands move over the network, traversing 10 hops on average, then at the rate of 0.06pJ/bit the network power would be 35 watts, more than half the power budget of the processor.

As the energy cost of computation is reduced by voltage scaling (described later), emphasizing compute throughput, the cost of data movement starts to dominate. Therefore, data movement must be restricted by keeping data locally as much as possible. This restriction also means the size of local storage (such as a register file) must increase substantially. This increase is contrary to conventional thinking of register files being small and thus fast. With voltage scaling the frequency of operation is lower anyway, so it makes sense to increase the size of the local storage at the expense of speed.

Another radical departure from conventional thinking is the role of the interconnect network on the chip. Recent parallel machine designs have been dominated by packet-switching,^{6,8,24,40} so multicore networks adopted this energy-intensive approach. In the future, data movement over these networks must be limited to conserve energy, and, more important, due to the large size of local storage data bandwidth, demand on the network will be reduced. In light of these findings on-die-network architectures need revolutionary approaches (such as hybrid packet/circuit switching⁴).

Many older parallel machines used irregular and circuit-switched networks^{31,41}; Figure 12 describes a return to hybrid switched networks for on-chip interconnects. Small cores in close proximity could be interconnected into clusters with traditional buses that are energy efficient for data movement over short distances. The clusters could be connected through wide (high-bandwidth) low-swing (low-energy) busses or through packet- or circuit-switched networks, depending on distance. Hence the network-on-a-chip could be hierarchical and heterogeneous, a radical departure from the

Figure 13. Improving energy efficiency through voltage scaling.

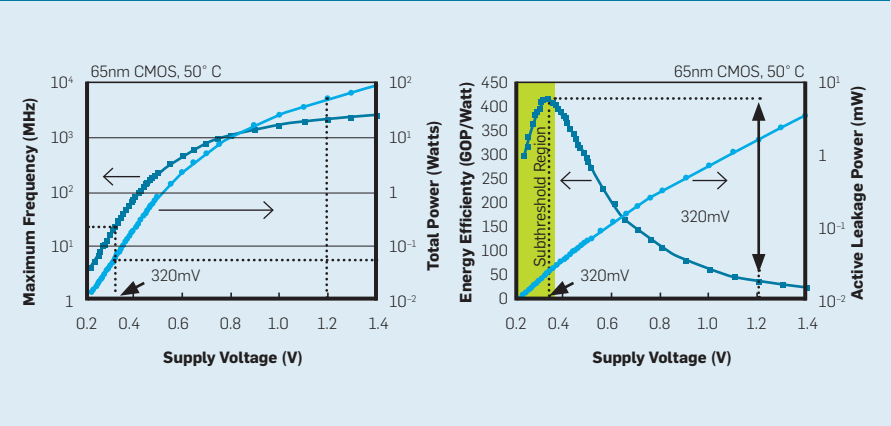


Table 6. Circuits challenges, trends, directions.

Challenge	Near-Term	Long-Term
Power, energy efficiency	Continuous dynamic voltage and frequency scaling, power gating, reactive power management	Discrete dynamic voltage and frequency scaling, near threshold operation, proactive fine-grain power and energy management
Variation	Speed binning of parts, corrections with body bias or supply voltage changes, tighter process control	Dynamic reconfiguration of many cores by speed
Gradual, temporal, intermittent, and permanent faults	Guard-bands, yield loss, core sparing, design for manufacturability	Resilience with hardware/software co-design, dynamic in-field detection, diagnosis, reconfiguration and repair, adaptability, and self-awareness

traditional parallel-machine approach (see Table 5).

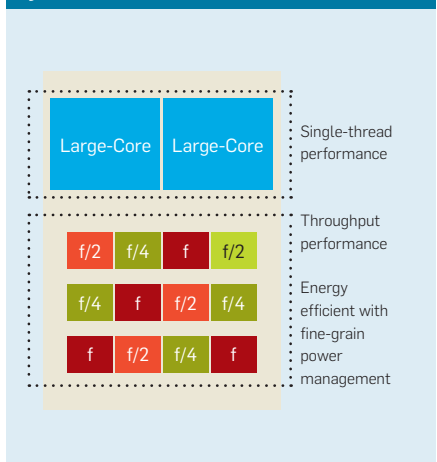
The role of microprocessor architect must expand beyond the processor core, into the whole platform on a chip, optimizing the cores as well as the network and other subsystems.

Pushing the envelope: Extreme circuits, variability, resilience. Our analysis showed that in the power-constrained scenario, only 150 million logic transistors for processor cores and 80MB of cache will be affordable due to energy by 2018. Note that 80MB of cache is not necessary for this system, and a large portion of the cache-transistor budget can be utilized to integrate even more cores if it can be done with the power-consumption density of a cache, which is 10x less than logic. This approach can be achieved through aggressive scaling of supply voltage.²⁵

Figure 13 outlines the effectiveness of supply-voltage scaling when the chip is designed for it. As the supply voltage is reduced, frequency

also reduces, but energy efficiency increases. When the supply voltage is reduced all the way to the transistor's threshold, energy efficiency increases by an order of magnitude. Employing this technique on large cores would dramatically reduce single-thread performance and is hence not recommended. However, smaller cores used

Figure 14. A heterogeneous many-core system with variation.



for throughput would certainly benefit from it. Moreover, the transistor budget from the unused cache could be used to integrate even more cores with the power density of the cache. Aggressive voltage scaling provides an avenue for utilizing the unused transistor-integration capacity for logic to deliver higher performance.

Aggressive supply-voltage scaling comes with its own challenges (such as variations). As supply voltage is reduced toward a transistor’s threshold voltage, the effect of variability is even worse, because the speed of a circuit is proportional to the voltage overdrive (supply voltage minus threshold voltage). Moreover, as supply voltage approaches the threshold, any small change in threshold voltage affects the speed of the circuit. Therefore, variation in the threshold voltage manifests itself as variation in the speed of the core, the slowest circuit in the core determines the frequency of operation of the core, and a large core is more susceptible to lower frequency of operation due to variations. On the other hand, a large number of small cores has a better distribution of fast and slow small cores and can better even out the effect of variations. We next discuss an example system that is variation-tolerant, energy-efficient, energy-proportional, and fine-grain power managed.

A hypothetical heterogeneous processor (see Figure 14) consists of a small number of large cores for single-thread performance and many small cores for throughput performance. Supply voltage and the frequency of any

given core are individually controlled such that the total power consumption is within the power envelope. Many small cores operate at lower voltages and frequency for improved energy efficiency, while some small cores operate near threshold voltage at the lowest frequency but at higher energy efficiency, and some cores may be turned off completely. Clock frequencies need not be continuous; steps (in powers of two) keep the system synchronous and simple without compromising performance while also addressing variation tolerance. The scheduler dynamically monitors workload and configures the system with the proper mix of cores and schedules the workload on the right cores for energy-proportional computing. Combined heterogeneity, aggressive supply-voltage scaling, and fine-grain power (energy) management enables utilization of a larger fraction of transistor-integration capacity, moving closer to the goal of 30x increase in compute performance (see Table 6).

Software challenges renewed: Programmability versus efficiency. The end of scaling of single-thread performance already means major software challenges; for example, the shift to symmetric parallelism has created perhaps the greatest software challenge in the history of computing,^{12,15} and we expect future pressure on energy-efficiency will lead to extensive use of heterogeneous cores and accelerators, further exacerbating the software challenge. Fortunately, the past decade has seen increasing adoption of high-level “productivity” languages^{20,34,35} built on

advanced interpretive and compiler technologies, as well as increasing use of dynamic translation techniques. We expect these trends to continue, with higher-level programming, extensive customization through libraries, and sophisticated automated performance search techniques (such as autotuning) will be even more important.

Extreme studies^{27,38} suggest that aggressive high-performance and extreme-energy-efficient systems may go further, eschewing the overhead of programmability features that software engineers have come to take for granted; for example, these future systems may drop hardware support for a single flat address space (which normally wastes energy on address manipulation/computing), single-memory hierarchy (coherence and monitoring energy overhead), and steady rate of execution (adapting to the available energy budget). These systems will place more of these components under software control, depending on increasingly sophisticated software tools to manage the hardware boundaries and irregularities with greater energy efficiency. In extreme cases, high-performance computing and embedded applications may even manage these complexities explicitly. Most architectural features and techniques we’ve discussed here shift more responsibility for distribution of the computation and data across the compute and storage elements of microprocessors to software.^{13,18} Shifting responsibility increases potential achievable energy efficiency, but realizing it depends on significant advances in applications, compilers and runtimes, and operating systems to understand and even predict the application and workload behavior.^{7,16,19} However, these advances require radical research breakthroughs and major changes in software practice (see Table 7).

Conclusion

The past 20 years were truly the great old days for Moore’s Law scaling and microprocessor performance; dramatic improvements in transistor density, speed, and energy, combined with microarchitecture and memory-hierarchy techniques delivered 1,000-fold microprocessor performance improvement. The next 20 years—the

Table 7. Software challenges, trends, directions.

Challenge	Near-Term	Long-Term
1,000-fold software parallelism	Data parallel languages and “mapping” of operators, library and tool-based approaches	New high-level languages, compositional and deterministic frameworks
Energy-efficient data movement and locality	Manual control, profiling, maturing to automated techniques (auto-tuning, optimization)	New algorithms, languages, program analysis, runtime, and hardware techniques
Energy management	Automatic fine-grain hardware management	Self-aware runtime and application-level techniques that exploit architecture features for visibility and control
Resilience	Algorithmic, application-software approaches, adaptive checking and recovery	New hardware-software partnerships that minimize checking and recomputation energy


pretty good new days, as progress continues—will be more difficult, with Moore’s Law scaling producing continuing improvement in transistor density but comparatively little improvement in transistor speed and energy. As a result, the frequency of operation will increase slowly. Energy will be the key limiter of performance, forcing processor designs to use large-scale parallelism with heterogeneous cores, or a few large cores and a large number of small cores operating at low frequency and low voltage, near threshold. Aggressive use of customized accelerators will yield the highest performance and greatest energy efficiency on many applications. Efficient data orchestration will increasingly be critical, evolving to more efficient memory hierarchies and new types of interconnect tailored for locality and that depend on sophisticated software to place computation and data so as to minimize data movement. The objective is ultimately the purest form of energy-proportional computing at the lowest-possible levels of energy. Heterogeneity in compute and communication hardware will be essential to optimize for performance for energy-proportional computing and coping with variability. Finally, programming systems will have to comprehend these restrictions and provide tools and environments to harvest the performance.

While no one can reliably predict the end of Si CMOS scaling, for this future scaling regime, many electrical engineers have begun exploring new types of switches and materials (such as compound semiconductors, carbon nanotubes, and graphene) with different performance and scaling characteristics from Si CMOS, posing new types of design and manufacturing challenges. However, all such technologies are in their infancy, probably not ready in the next decade to replace silicon but will pose the same challenges with continued scaling. Quantum electronics (such as quantum dots) are even farther out and when realized will reflect major challenges of its own, with yet newer models of computation, architecture, manufacturing, variability, and resilience.

Because the future winners are far from clear today, it is way too early to

predict whether some form of scaling (perhaps energy) will continue or there will be no scaling at all. The pretty good old days of scaling that processor design faces today are helping prepare us for these new challenges. Moreover, the challenges processor design will face in the next decade will be dwarfed by the challenges posed by these alternative technologies, rendering today’s challenges a warm-up exercise for what lies ahead.

Acknowledgments

This work was inspired by the Exascale study working groups chartered in 2007 and 2008 by Bill Harrod of DARPA. We thank him and the members and presenters to the working groups for valuable insightful discussions over the past few years. We also thank our colleagues at Intel who have improved our understanding of these issues through many thoughtful discussions. Thanks, too, to the anonymous reviewers whose extensive feedback greatly improved the article. 

References

1. Advanced Vector Extensions. Intel; http://en.wikipedia.org/wiki/Advanced_Vector_Extensions
2. Altivec, Apple, IBM, Freescale; <http://en.wikipedia.org/wiki/Altivec>
3. Amdahl, G. Validity of the single-processor approach to achieving large-scale computing capability. AFIPS Joint Computer Conference (Apr. 1967), 483–485.
4. Anders, M. et al. A 4.1Tb/s bisection-bandwidth 560Gb/s/W streaming circuit-switched 8x8 mesh network-on-chip in 45nm CMOS. International Solid State Circuits Conference (Feb. 2010).
5. Barroso, L.A. and Hölzle, U. The case for energy-proportional computing. *IEEE Computer* 40, 12 (Dec. 2007).
6. Bell, S. et al. TILE64 processor: A 64-core SoC with mesh interconnect. IEEE International Solid-State Circuits Conference (2008).
7. Bienia, C. et al. The PARSEC benchmark suite: Characterization and architectural implications. The 17th International Symposium on Parallel Architectures and Compilation Techniques (2008).
8. Blumrich, M. et al. *Design and Analysis of the Blue Gene/L Torus Interconnection Network*. IBM Research Report, 2003.
9. Borkar, S. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro* 25, 6 (Nov.–Dec. 2005).
10. Borkar, S. Design challenges of technology scaling. *IEEE Micro* 19, 4 (July–Aug. 1999).
11. Borkar, S. et al. Parameter variations and impact on circuits and microarchitecture. The 40th Annual Design Automation Conference (2003).
12. Catanzaro, B. et al. Ubiquitous parallel computing from Berkeley, Illinois, and Stanford. *IEEE Micro* 30, 2 (2010).
13. Cray, Inc. *Chapel Language Specification*. Seattle, WA, 2010; <http://chapel.cray.com/spec/spec-0.795.pdf>
14. Chien, A. 10x10: A general-purpose architectural approach to heterogeneity and energy efficiency. *The Third Workshop on Emerging Parallel Architectures at the International Conference on Computational Science* (June 2011).
15. Chien, A. Pervasive parallel computing: An historic opportunity for innovation in programming and architecture. ACM Principles and Practice of Parallel Programming (2007).
16. Cooper, B. et al. Benchmarking cloud serving systems

- with YCSB. ACM Symposium on Cloud Computing (June 2010).
17. Dennard, R. et al. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE Journal of Solid State Circuits* SC-9, 5 (Oct. 1974), 256–268.
18. Fatahalian, K. et al. Sequoia: Programming the memory hierarchy. ACM/IEEE Conference on Supercomputing (Nov. 2006).
19. Flinn, J. et al. Managing battery lifetime with energy-aware adaptation. *ACM Transactions on Computer Systems* 22, 2 (May 2004).
20. Gosling, J. et al. *The Java Language Specification, Third Edition*. Addison-Wesley, 2005.
21. Hameed, R. et al. Understanding sources of inefficiency in general-purpose chips. International Symposium on Computer Architecture (2010).
22. Hoskote, Y. et al. A TCP offload accelerator for 10Gb/s Ethernet in 90-nm CMOS. *IEEE Journal of Solid-State Circuits* 38, 11 (Nov. 2003).
23. International Technology Roadmap for Semiconductors, 2009; <http://www.itrs.net/Links/2009ITRS/Home2009.htm>
24. Karamcheti, V. et al. Comparison of architectural support for messaging in the TMC CM-5 and Cray T3D. International Symposium on Computer Architecture (1995).
25. Kaul, H. et al. A 320mV 56W 411GOPS/Watt ultra-low-voltage motion-estimation accelerator in 65nm CMOS. *IEEE Journal of Solid-State Circuits* 44, 1 (Jan. 2009).
26. The Khronos Group. *OpenCL, the Open Standard for Heterogeneous Parallel Programming*, Feb. 2009; <http://www.khronos.org/opencl/>
27. Kogge, P. et al. *Exascale Computing Study: Technology Challenges in Achieving an Exascale System*; http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf
28. Mazor, S. The history of microcomputer-invention and evolution. *Proceedings of the IEEE* 83, 12 (Dec. 1995).
29. Noguchi, K., Ohnishi, I., and Morita, H. Design considerations for a heterogeneous tightly coupled multiprocessor system. AFIPS National Computer Conference (1975).
30. Nvidia Corp. *CUDA Programming Guide Version 2.0*, June 2008; http://www.nvidia.com/object/cuda_home_new.html
31. Pfister, G. et al. The research parallel processor prototype (RP3): Introduction and architecture. *International Conference on Parallel Processing* (Aug. 1985).
32. Pollack, F. *Pollack’s Rule of Thumb for Microprocessor Performance and Area*; http://en.wikipedia.org/wiki/Pollack’s_Rule
33. Przybylski, S.A. et al. Characteristics of performance-optimal multi-level cache hierarchies. International Symposium on Computer Architecture (June 1989).
34. Richter, J. *The CLR Via C#, Second Edition*, 1997.
35. Ruby Documentation Project. *Programming Ruby: The Pragmatic Programmer’s Guide*; <http://www.ruby-doc.org/docs/ProgrammingRuby/>
36. Seiler, L. et al. Larrabee: Many-core x86 architecture for visual computing. *ACM Transactions on Graphics* 27, 3 (Aug. 2008).
37. Strecker, W. Transient behavior of cache memories. *ACM Transactions on Computer Systems* 1, 4 (Nov. 1983).
38. Sarkar, V. et al. *Exascale Software Study: Software Challenges in Extreme-Scale Systems*; <http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/ECSS%20report%20101909.pdf>
39. Tartar, J. Multiprocessor hardware: An architectural overview. ACM Annual Conference (1980).
40. Weingold, E. et al. Baring it all to software: Raw machines. *IEEE Computer* 30, 9 (Sept. 1997).
41. Wulf, W. and Bell, C.G. C.mmp: A multi-mini-processor. *AFIPS Joint Computer Conferences* (Dec. 1972).

Shekhar Borkar (Shekhar.Y.Borkar@intel.com) is an Intel Fellow and director of exascale technology at Intel Corporation, Hillsboro, OR.

Andrew A. Chien (Andrew.Chien@alum.mit.edu) is former vice president of research at Intel Corporation and currently adjunct professor in the Computer Science and Engineering Department at the University of California, San Diego.