Computing Performance: Game Over or Next Level?

Samuel H. Fuller, *Analog Devices Inc.*
Lynette I. Millett, *National Research Council*

The end of dramatic exponential growth in single-processor performance marks the end of the dominance of the single microprocessor in computing. The era of sequential computing must give way to an era in which parallelism holds the forefront. Although important scientific and engineering challenges lie ahead, this is an opportune time for innovation in programming systems and computing architectures.

Last year, the Computer Science and Telecommunications Board (CSTB) of the US National Academy of Sciences released *The Future of Computing Performance: Game Over or Next Level?*[1] With sponsorship from the US National Science Foundation, the CSTB convened a committee of experts to identify key challenges to continued growth in computing performance and to outline a research agenda for meeting the emerging computing needs of the 21st century. These experts brought diverse perspectives in the fields of semiconductor technology, computer architecture, programming languages and methods, and applications to explore challenges to sustaining performance growth and meeting broad societal expectations for computing now and in the future.

The committee's observations, findings, and recommendations can be broadly summarized in two categories: energy and power constraints on growth in computing performance, and a proposed research agenda that emphasizes new approaches to software and parallelism to meet future expectations for performance growth.

COMPUTING GROWTH DEPENDENCE

Information technology has transformed how we work and live—and has the potential to continue doing so. IT helps bring distant people together, coordinate disaster response, enhance economic productivity, enable new medical diagnoses and treatments, add new efficiencies to our economy, improve weather prediction and climate modeling, broaden educational access, strengthen national defense, advance science, and produce and deliver content for education and entertainment.

These transformations have been made possible by sustained improvements in computer performance. We have been living in a world where information processing costs have been decreasing exponentially year after year. Moore's law—which originally referred to an empirical observation about the most economically favorable rate for industry to increase the number of transistors on a chip—has come to be associated with the expectation that microprocessors will become faster, communication bandwidth will increase, storage will become less expensive, and, more broadly, computers will become faster. Most notably, the performance of individual computer processors increased some 10,000 times over the past two decades, without substantial power consumption increases.

Although some might say they do not want or need a faster computer, users and the computer industry now depend on continuing this performance growth. Much IT innovation depends on taking advantage of computing performance's leading edge. The IT industry annually generates a trillion dollars and has even larger indirect effects throughout society.

This huge economic engine depends on a sustained demand for IT products and services, which in turn fuels demand for constantly improving performance. More broadly, virtually every sector of society—manufacturing, financial services, education, science, government, the military, and entertainment—now depends on this continued growth in computing performance to drive industrial productivity, increase efficiency, and enable innovation. The performance achievements have driven an implicit, pervasive expectation that future IT advances will occur as an inevitable continuation of the stunning advances IT has experienced in the past half-century.

Growth in single-processor performance has stalled—or at best is being increased only marginally over time.

Software developers themselves have come to depend on performance growth across several dimensions:

- adding visible features and ever more sophisticated interfaces to existing applications;
- increasing "hidden" (nonfunctional) value—such as improved security, reliability, and other trustworthiness features—without degrading the performance of existing functions;
- using higher-level abstractions, programming languages, and systems that require more computing power but reduce development time and improve software quality by making the development of correct programs and component integration easier; and
- anticipating performance improvements and creating innovative, computationally intensive applications even before the required performance is available at low cost.

Five decades of exponential performance growth have also made dominant the general-purpose microprocessor at the heart of every personal computer. This stems first from a cycle of economies of scale, wherein each computer generation has been both faster and less expensive than the previous one. Second, increased software portability lets current and forthcoming software applications run correctly and faster on new computers.

These economies have resulted from the application of Moore's law to transistor density, along with innovative approaches to effectively harness the new transistors that have become available. Software portability has been preserved by keeping instruction sets compatible over many generations of microprocessors, even as the underlying microprocessor technology underwent substantial enhancements, allowing investments in software to be amortized over long periods.

The success of this virtuous cycle dampened interest in the development of alternative computer and programming models. Alternative architectures might have been technically superior (for example, faster or more power-efficient) in specific domains, but, generally speaking, if they did not offer software compatibility, they could not easily compete in the marketplace and were overtaken by the ever-improving general-purpose processors available at relatively low cost.

SINGLE-PROCESSOR PERFORMANCE-GROWTH CONSTRAINTS

By the 2000s, however, it had become apparent that processor performance growth faced two major constraints.

First, the ability to increase clock speeds locked horns with power limits. The densest, highest-performance, and most power-efficient integrated circuits (ICs) are constructed from complementary metal-oxide semiconductor (CMOS) technology.

By 2004, the long-fruitful strategy of scaling down the size of CMOS circuits, reducing the supply voltage, and increasing the clock rate had become infeasible. Since a chip's power consumption is in proportion to the clock speed times the supply voltage squared, the inability to continue to lower the supply voltage halted developers' ability to increase the clock speed without increasing power dissipation.[2] The resulting power consumption exceeded the few hundred watts per chip level that can practically be dissipated in a mass-market computing system, as well as the practical limit of a few watts for mobile, battery-powered devices. The ultimate consequence has been that growth in single-processor performance has stalled—or at best is being increased only marginally over time.

Second, efforts to improve individual processors' internal architecture have netted diminishing returns. Many advances in the architecture of general-purpose sequential processors, such as deeper pipelines and speculative execution, have contributed to successful exploitation of increasing transistor densities. Today, however, there appears to be little opportunity to significantly increase performance by improving the internal structure of existing sequential processors.

The slowdown in processor performance growth, clock speed, and power since 2004 is evident in Figure 1, which also shows the continued, exponential growth in the number of transistors per chip. The original Moore's law projection of increasing transistors per chip remains unabated even as performance has stalled. The 2009 edition of the *International Technology Roadmap for Semiconductors* (www.itrs.net/Links/2009ITRS/Home2009.htm) predicts this growth continuing through the next decade, but we will probably be unable to continue increasing transistor density for CMOS circuits at the current pace for more than the next 10 years.

Figure 2 shows this expectation gap using a logarithmic vertical scale. In 2010, this gap for single-processor performance is approximately a factor of 10; by 2020, the gap will have grown to about a factor of 1,000. Most economic or societal sectors implicitly or explicitly expect computing to deliver steady, exponentially increasing performance, but these graphs show traditional single-processor computing systems will not match expectations.

By 2020, we will see a large "expectation gap" for single processors. After many decades of dramatic exponential growth, single-processor performance is slowing and not expected to improve in the foreseeable future. Energy and power constraints play an important and growing role in computing performance. Computer systems require energy to operate, and, as with any device, the more energy needed, the more expensive the system is to operate and maintain. Moreover, the energy consumed by the system ends up as heat that must be removed. Even with new parallel models and solutions, the performance of most future computing systems will be limited by power or energy in ways the computer industry and researchers have yet to confront.

For example, the benefits of replacing a single, highly complex processor with increasing numbers of simpler processors will eventually reach a limit when further simplification costs more in performance than it saves in power. Power constraints are thus inevitable for systems ranging from handheld devices to the largest computing datacenters, even as the transition is made to parallel systems.

Total energy consumed by computing systems is already substantial and continues to grow rapidly in the US and elsewhere around the world. As is the case in other economic sectors, the total energy consumed by computing will come under increasing pressure.

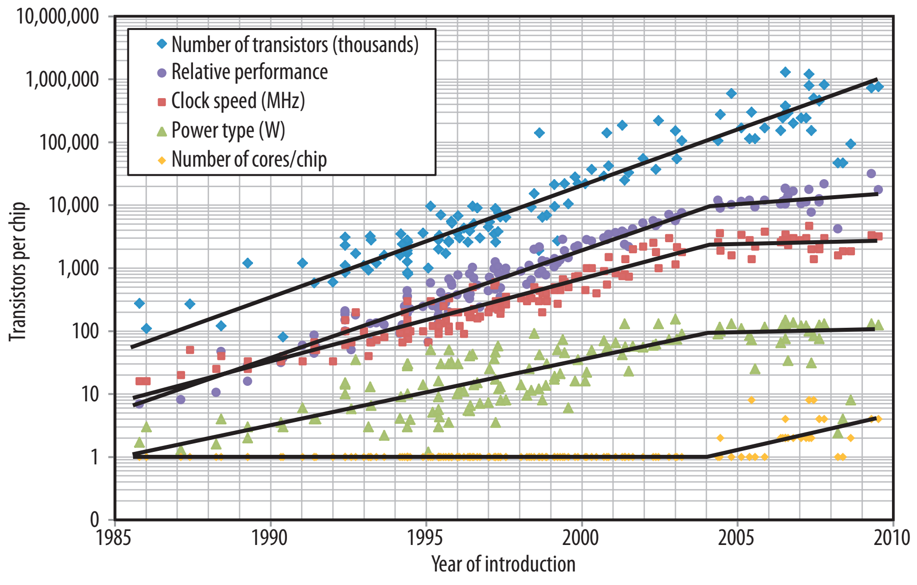

Figure 1. Transistors, frequency, power, performance, and processor cores over time. The original Moore's law projection of increasing transistors per chip remains unabated even as performance has stalled.

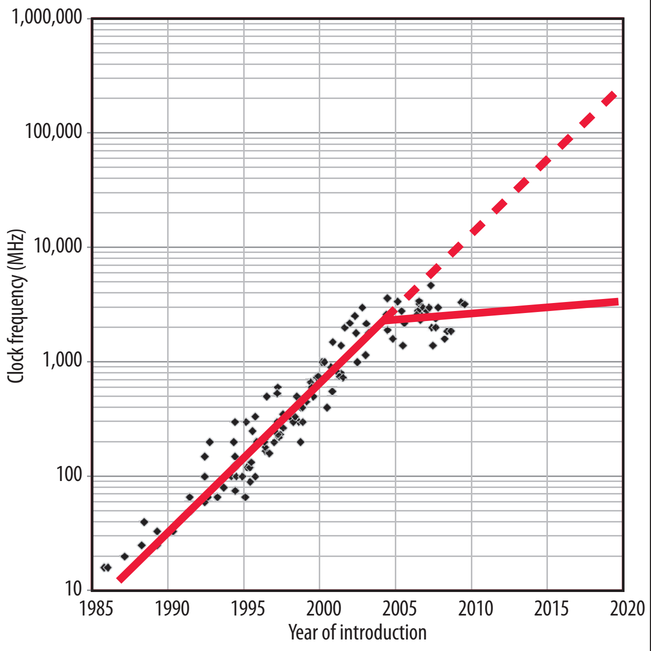

Figure 2. Historical growth in single-processor performance and a forecast of processor performance to 2020, based on the ITRS roadmap. A dashed line represents expectations if single-processor performance had continued its historical trend.

Even if we succeed in sidestepping the limits on single-processor performance, total energy consumption will remain an important concern, and growth in performance will become limited by power consumption within a decade.

In short, the single processor and the sequential programming model that dominated computing since its birth in the 1940s will no longer be sufficient to deliver the continued growth in performance needed to facilitate future IT advances. Moreover, whether power and energy will be showstoppers or just significant constraints remains an open question. Although these issues pose major technical challenges, they will also drive considerable innovation in computing by forcing us to rethink the von Neumann model that has prevailed since the 1940s.

SOLVING WITH PARALLELISM

Future growth in computing performance must come from parallelism. Today, most software developers think and program using a sequential programming model to create software for single general-purpose microprocessors. The microprocessor industry has already begun to deliver parallel hardware in mainstream products with chip multiprocessors (CMPs), an approach that places new burdens on software developers to build applications that take advantage of multiple, distinct cores.

Whether power and energy will be showstoppers or just significant constraints remains an open question.

Although developers have found reasonable ways to use two or even four cores effectively by running independent tasks on each one, they have not, for the most part, parallelized individual tasks to make full use of the available computational capacity. Moreover, if industry continues to follow the same trends, they will soon be delivering chips with hundreds of cores. Harnessing these will require new techniques for parallel computing, including breakthroughs in software models, languages, and tools. Developers of both hardware and software will need to focus more attention on overall system performance, likely at the expense of time to market and the efficiency of the virtuous cycle previously described.

The computer science and engineering communities have worked for decades on the hard problems associated with parallelism. For example, high-performance computing for science and engineering applications has depended on particular parallel-programming techniques such as implementing the message passing interface (MPI). In other cases, domain-specific languages and abstractions such as MapReduce[3] have provided interfaces with behind-the-scenes parallelism and well-chosen abstractions developed by experts—technologies that hide the complexity of parallel programming from application developers.

These efforts have typically involved a small cadre of programmers with highly specialized training in parallel programming who work on relatively narrow computing problems. None of this work has, however, come close to enabling widespread use of parallel programming for a wide array of computing problems.

A few research universities, including MIT, the University of Washington, and the University of California, Berkeley, have launched or revived research programs in parallelism. The topic has found a renewed focus in industry at companies such as Nvidia. However, these initial investments are not commensurate with the magnitude of the technical challenges or the stakes. Moreover, history shows that technology advances of this sort often take a decade or more.[4] The results of such research are needed today to sustain historical trends in computing performance, which already puts us a decade behind. Even with concerted investment, there is no guarantee that widely applicable solutions will be found. If they cannot be, we need to know this quickly so that we can explore other avenues.

MEETING THE CHALLENGES

Current technological challenges affect not only computing but also the many sectors of society that now depend on advances in IT and computation. These suggest national and global economic repercussions. At the same time, the crisis in computing performance has pointed to new opportunities for innovation in diverse hardware and software infrastructures that excel in metrics other than single-processor performance, such as low-power consumption and aggregate delivery of throughput cycles. There are opportunities for major changes in system architectures. Further, we need extensive investment in whole-system research to lay the foundation for next-generation computing environments.

The CSTB committee's recommendations are broadly aimed at federal research agencies, the computing and information technology industry, and educators, and they fall into two categories. The first is research. The best science and engineering minds must be brought to bear on our most daunting challenges. The second category is practice and education. Better practice in developing computer hardware and software today will provide a foundation for future performance gains. Education will empower the emerging generation of technical experts to understand different and in some cases not-yet-developed parallel models for thinking about IT, computation, and software.

RESEARCH RECOMMENDATIONS

The committee urges investment in several crosscutting areas of research, including algorithms, broadly usable parallel programming methods, rethinking the canonical computing stack, parallel architectures, and power efficiency.

Researchers must invest in and develop algorithms that can exploit parallel processing. Today, relatively little software is explicitly parallel. To obtain the desired performance, many—if not most—software designers must grapple with parallelism. For some applications, they might still be able to write sequential programs, leaving it to compilers and other software tools to extract the parallelism in the underlying algorithms. For more complex applications, it might be necessary for programmers to write explicitly parallel programs. Parallel approaches are already used in some applications when no viable alternative is available. The committee believes that careful attention to parallelism will become the rule rather than the exception.

Further, it will be important to invest in research on and development of programming methods that will enable efficient use of parallel systems by typical programmers as well as by experts in parallel systems. Many of today's programming models, languages, compilers, hypervisors, and operating systems are targeted primarily at single-processor hardware. In the future, these layers will need to target, optimize programs for, and be optimized themselves for explicitly parallel hardware.

Rethinking programming models

The intellectual keystone of this endeavor is rethinking programming models so that programmers can express application parallelism naturally. This will let parallel software be developed for diverse systems rather than specific configurations, and let system software deal with balancing computation and minimizing communication among multiple computational units.

This situation is reminiscent of the late 1970s, when programming models and tools were inadequate for building substantially more complex software. Better programming models—such as structured programming in the 1970s, object orientation in the 1980s, and managed programming languages in the 1990s—have made it possible to produce much more sophisticated software. Analogous advances in the form of better tools and additional training will be needed to increase programmer productivity for parallel systems.

The ability to express application parallelism so that an application runs faster as more cores are added would provide a key breakthrough. The most prevalent parallel-programming languages do not provide this performance portability. A related question is what to do with the enormous body of legacy sequential code, which will only contribute to substantial performance improvements if it can be parallelized.

Experience has shown that parallelizing sequential code or highly sequential algorithms effectively is exceedingly difficult in general. Writing software that expresses the type of parallelism required to exploit chip multiprocessor hardware requires new software engineering processes and tools, including new programming languages that ease the expression of parallelism, and a new software stack that can exploit and map the parallelism to diverse and evolving hardware. It will also require training programmers to solve their problems with parallel computational thinking.

The models themselves might or might not be explicitly parallel; whether or when most programmers should be exposed to explicit parallelism remains an open question. A single, universal programming model might or might not exist, so multiple models—including some domain-specific ones—should be explored.

We need additional research in the development of new libraries and programming languages, with appropriate compilation and runtime support that embodies the new programming models. It seems reasonable to expect that

Advances in the form of better tools and additional training will be needed to increase programmer productivity for parallel systems.

some programming models, libraries, and languages will be suited for a broad base of skilled but not superstar programmers. They could even appear on the surface to be sequential or declarative. Others, however, will target efficiency, contributing to the highest performance for critical subsystems that are to be extensively reused, and thus be intended for a smaller set of expert programmers.

System software

Research should also focus on system software for highly parallel systems. Although today's operating systems can handle some modest parallelism, future systems will include many more processors whose allocation, load balancing, and data communication and synchronization interactions will be difficult to handle well. Solving those problems will require a rethinking of how computation and communication resources are viewed, much as demands for increased memory led to the introduction of virtual memory a half-century ago.

Long-term efforts should focus on rethinking the canonical computing "stack"—applications, programming language, compiler, runtime, virtual machine, operating system, hypervisor, and architecture—in light of parallelism and resource-management challenges. Computer scientists and engineers typically manage complexity by separating the interface from its implementation. In conventional computer systems, developers do this recursively to form a computing stack of applications, programming language, compiler, runtime, virtual machine, operating system, hypervisor, and architecture components.

Whether today's conventional stack provides the right framework to support parallelism and manage resources remains unclear. The structure and elements of the stack itself should be a focus of long-term research exploration.

Rethinking architecture

We must invest in research on and development of parallel architectures driven by applications, including enhancements to chip multiprocessor systems and conventional data-parallel architectures, cost-effective designs for application-specific architectures, and support for radically different approaches. In addition, advances in architecture and hardware will play an important role. One path forward continues to refine CMPs and associated architectural approaches. We must determine if today's CMP approaches are suitable for designing most computers.

We must invest in research on and development of parallel architectures driven by applications.

The current CMP architecture, the heart of this architectural franchise, keeps companies investing heavily. But CMP architectures bring their own challenges. We must determine if large numbers of cores work in most computer deployments, such as desktops and even mobile phones. We must then see how the cores can be harnessed temporarily, in an automated or semiautomated fashion, to overcome sequential bottlenecks. Leveraging the mechanisms and policies to best exploit locality, we should keep data stored close to other data that might be needed at the same time or for particular computations, while avoiding communications bottlenecks. Finally, we must address how to handle synchronization and scheduling, how to address the challenges associated with power and energy, and what the new architectures mean for such system-level features as reliability and security.

Using homogeneous processors in CMP architectures provides one approach, but computer architectures that include multiple or heterogeneous cores, some of which might be more capable than others, or even use different instruction-set architectures, could prove most effective. Special-purpose processors have long exploited parallelism, notably graphics processing units (GPUs) and digital signal processor (DSP) hardware. These have been successfully deployed in important segments of the market. Other niches like these could be filled by GPUs and DSPs, or computing cores that use more graphics for GPU support of general-purpose programs might cause differences between the two approaches to blur.

Perhaps some entirely new architectural approach will prove more successful. If systems with CMP architectures cannot be effectively programmed, an alternative will be needed. Work in this general area could sidestep conventional cores and view the chip as a tabula rasa with billions of transistors, translating into hundreds of functional units. The effective organization of these units into a programmable architecture is an open question. Exploratory computing systems, based on programmable gate arrays, offer a step in this direction, but we need continued innovation to develop programming systems that can harness the field-programmable gate array's potential parallelism.

Another place where fundamentally different approaches might be needed could involve alternatives to CMOS. There are many advantages to sticking with today's silicon-based CMOS technology, which has proven remarkably scalable over many generations of microprocessors, and around which an enormous industrial and experience base has been established. However, it will also be essential to invest in new computation substrates whose underlying power efficiency promises to be fundamentally better than that of silicon-based CMOSs. Computing has benefited in the past from order-of-magnitude performance improvements in power consumption in the progression from vacuum tubes, to discrete bipolar transistors, to ICs first based on bipolar transistors, to N-type metal-oxide-semiconductor (NMOS) logic, to CMOS. No alternative has approached commercial availability yet, although some show potential.

In the best case, investment will yield devices and manufacturing methods as yet unforeseen that will dramatically surpass the CMOS IC. Worst case, no new technology will emerge to help solve current problems. This uncertainty argues for investment in multiple approaches as soon as possible, and computer system designers would be well advised not to expect one of the new devices to appear in time to obviate the development of new, parallel architectures built on proven CMOS technology.

We need better performance immediately. Society cannot wait the decade or two it would take to identify, refine, and apply a new technology that might not materialize. Moreover, even if researchers do discover a groundbreaking technology, the investment in parallelism would not be wasted because its advances would probably exploit the new technology as well.

Power efficiency

Because energy consumption and power dissipation increasingly limit computing systems, developing efficient power sources is critical. We must invest in research to make computer systems more power-efficient at all system levels, including software, application-specific approaches, and alternative devices. R&D efforts should address ways in which software and system architectures can improve power efficiency, such as exploiting locality and the use of domain-specific execution units. R&D should also be aimed at making logic gates more power-efficient. Such efforts should address alternative physical devices beyond incremental improvements in today's CMOS circuits.

Exploiting parallelism alone cannot ensure continued growth in computer performance. Developers have many potential avenues for investigating better power efficiency, some of which require sustained attention to known engineering issues and others that require further research. These include the following approaches:

- Redesign the delivery of power to and removal of heat from computing systems for increased efficiency.
- Design and deploy systems in which we use the absolute maximum fraction of power to do the computing, and less for routing power to the system and removing heat from it. New standards—including ones that set ever more aggressive targets—might provide useful incentives for the development of better techniques.
- Develop alternatives to the general-purpose processor that exploit locality.
- Develop domain-specific or application-specific processors analogous to GPUs and DSPs that provide better performance and power consumption characteristics than general-purpose processors for other specific application domains.
- Investigate possible new, lower-power device technology beyond CMOS.

Additional research should focus on system designs and software configurations that reduce power consumption, such as when resources are idle, or reducing power-consumption mapping applications to domain-specific and heterogeneous hardware units, limiting the amount of communication among disparate hardware units.

Although the shift toward CMPs will let industry continue to scale the performance of CMPs based on general-purpose processors for some time, general-purpose CMPs will eventually reach their own limits. CMP designers can trade off single-thread performance of individual processors against lower energy dissipation per instruction, thus allowing more instructions by multiple processors while holding the chip's power dissipation constant. However, that is possible only within a limited range of energy performance.

Beyond some limit, however, lowering energy per instruction by processor simplification can lead to degradation in overall CMP performance. This happens when processor performance starts to decrease faster than energy per instruction, which then requires new approaches to create more energy-efficient computers.

It might be that general-purpose CMPs will prove to be an inefficient solution in the long run, and we will need to create more application-optimized processing units. Tuning hardware and software toward a specific type of application provides a much more energy-efficient solution.

However, the current design trend is away from building customized solutions, because increasing design complexity has caused nonrecurring engineering costs for designing chips to grow rapidly. High costs limit the range of potential market segments to the few that have volume high enough to justify the initial engineering investment. A shift to more application-optimized computing systems, if necessary, demands a new design approach that would let application-specific chips be created at reasonable cost.

Additional research should focus on system designs and software configurations that reduce power consumption.

PRACTICE AND EDUCATION RECOMMENDATIONS

Implementing the proposed research agenda, although crucial for progress, will take time. Meanwhile, society has an immediate and pressing need to use current and emerging CMP systems effectively. Efforts in current development and engineering practices and education are also important. The CSTB committee encourages development of open interface standards for parallel programming to promote cooperation and innovation by sharing rather than proliferating proprietary programming environments.

Private-sector firms are often incentivized to create proprietary interfaces and implementations to establish a competitive advantage. However, a lack of standardization can impede progress because the presence of so many incompatible approaches deprives most from achieving the benefits of wide adoption and reuse—a major reason industry participates in standards efforts. The committee encourages the development of programming-interface standards that can facilitate wide adoption of parallel programming even as they foster competition in other areas.

We must develop tools and methods for transforming legacy applications to parallel systems. Whatever long-term success we achieve in the effective use of parallel systems from rethinking algorithms and developing new programming methods will probably come at the expense of the backward- and cross-platform compatibility that has been an IT economic cornerstone for decades. To salvage value from the nation's current, substantial IT investment, we must seek ways to bring sequential programs into the parallel world.

The committee urges industry and academia to develop “power tools” that will help experts migrate legacy code to tomorrow’s parallel computers. In addition, emphasis should be placed on tools and strategies to enhance code creation, maintenance, verification, and the adaptation of parallel programs.

Computer science education must increase the emphasis on parallelism by using a variety of methods and approaches to prepare students for the shape of the computing resources they will work with through their careers. We must encourage those who will develop the future’s parallel software. To sustain IT innovation, we will need a workforce that is adept in writing parallel applications that run well on parallel hardware, in creating parallel software systems, and in designing parallel hardware.

Both undergraduate and graduate students in computer science, as well as those in other fields that intensively use computing, will need to be educated in parallel programming. The engineering, science, and computer science curricula at both the undergraduate and graduate levels should begin to incorporate an emphasis on parallel computational thinking, parallel algorithms, and parallel programming.

With respect to the computer-science curriculum, given that no general-purpose paradigm has emerged, universities should teach diverse parallel-programming languages, abstractions, and approaches until effective ways of teaching and programming emerge. The necessary shape of the needed changes will not be clear until reasonably general parallel-programming methods have been devised and shown to be promising.

In relation to this goal, we must improve the programming workforce’s ability to cope with parallelism’s new challenges. On the one hand, this will involve retraining today’s programmers. On the other, it will demand developing new models and abstractions to make parallel programming more accessible to typically skilled programmers.

The end of dramatic exponential growth in single-processor performance also ends the single microprocessor’s dominance. The era of sequential computing must give way to a new era in which parallelism holds the forefront. There is no guarantee we can make parallel computing as common and easy to use as yesterday’s sequential single-processor computer systems, but unless we aggressively pursue the efforts suggested by the CSTB committee’s recommendations, it will be *game over* for growth in computing performance. This has larger implications. If parallel programming and software efforts fail to become widespread, the development of exciting new applications that drive the computer industry will slow and affect many other parts of the economy.

Although important scientific and engineering challenges lie ahead, this is an opportune time for innovation in programming systems and computing architectures. We have already begun to see diversity in computer designs to optimize for such considerations as power and throughput. The next generation of discoveries will likely require advances at all levels of the computing systems’ hardware and software to achieve the *next level* of benefits to society.

Acknowledgments

We thank the entire membership and staff of the Committee on Sustaining Growth in Computing Performance for their contributions to this article: Samuel H. Fuller (chair), Luiz Andre Barroso, Robert P. Colwell, William J. Dally, Dan Dobberpuhl, Pradeep Dubey, Mark D. Hill, Mark Horowitz, David Kirk, Monica Lam, Kathryn S. McKinley, Charles Moore, Katherine Yelick, Lynette I. Millett, and Jon Eisenberg. Committee member Mark Horowitz curated the data for the graphs. Support for this project was provided by the National Science Foundation under award CNS-0630358.

References

1. National Research Council, *The Future of Computing Performance: Game Over or Next Level?* Nat’l Academies Press, 2010.
2. R.H. Dennard et al., “Design of Ion-Implanted MOSFETS with Very Small Physical Dimensions,” *IEEE J. Solid State Circuits*, vol. 9, no. 5, 1974, pp. 256-268.
3. J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” *Comm. ACM*, vol. 51, no. 1, 2008, pp. 107-113.
4. National Research Council, *Evolving the High-Performance Computing and Communications Initiative to Support the Nation’s Information Infrastructure*, Nat’l Academies Press, 1995.

***Samuel H. Fuller** is the CTO and vice president of research and development at Analog Devices Inc. He received a PhD in electrical engineering from Stanford University. He is an IEEE Fellow. Contact him at sam.fuller@analog.com.*

***Lynette I. Millett** is senior staff officer at the Computer Science and Telecommunications Board, National Research Council of the National Academy of Sciences. She received an MSc in computer science from Cornell University. Contact her at lmillett@nas.edu.*

The Computer Science and Telecommunications Board of the National Academies will host a symposium planned for 22 February 2011 in Washington, D.C., to explore future directions in sustaining computing performance improvements. See http://cstb.org for more information.

Selected CS articles and columns are available for free at http://ComputingNow.computer.org.