Computer Organization and Assembly Language
CS / EE 320
Spring 2025

Lecture 1

Shahid Masud

Difference between Computer **Organization** and Computer **Architecture**

Architecture: View from outside, high level specs (Programs looking towards CPU Instructions)

Organization: View from Inside (CPU looking outwards towards buses, memory and peripherals)

Performance Graphs from the report of National Academy, USA

‘The Future of Computer Performance

Game Over or Next Lavel

Ed. Samuel H Fuller

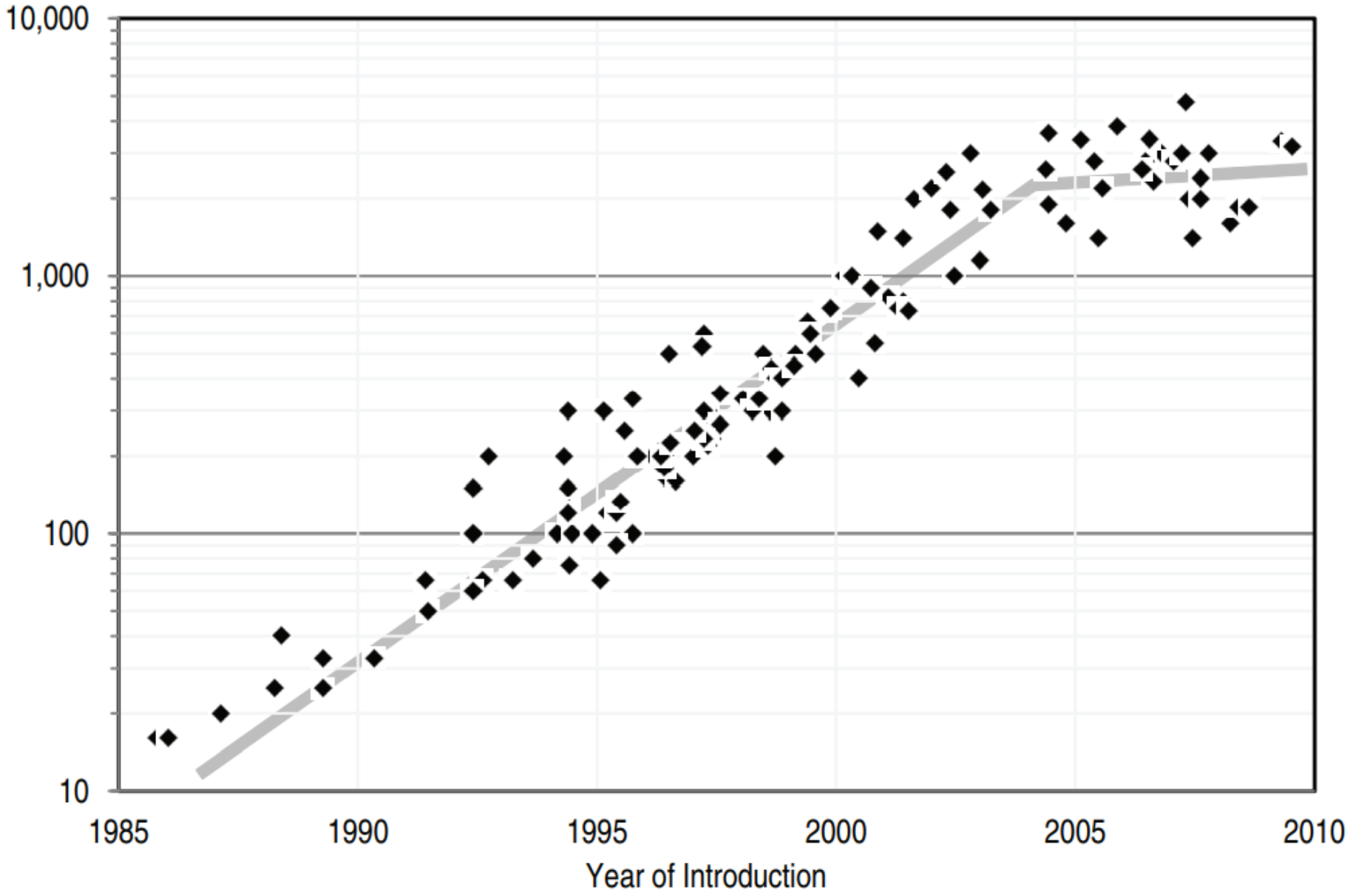


FIGURE 3.3 Microprocessor-clock frequency (MHz) over time (1985-2010).

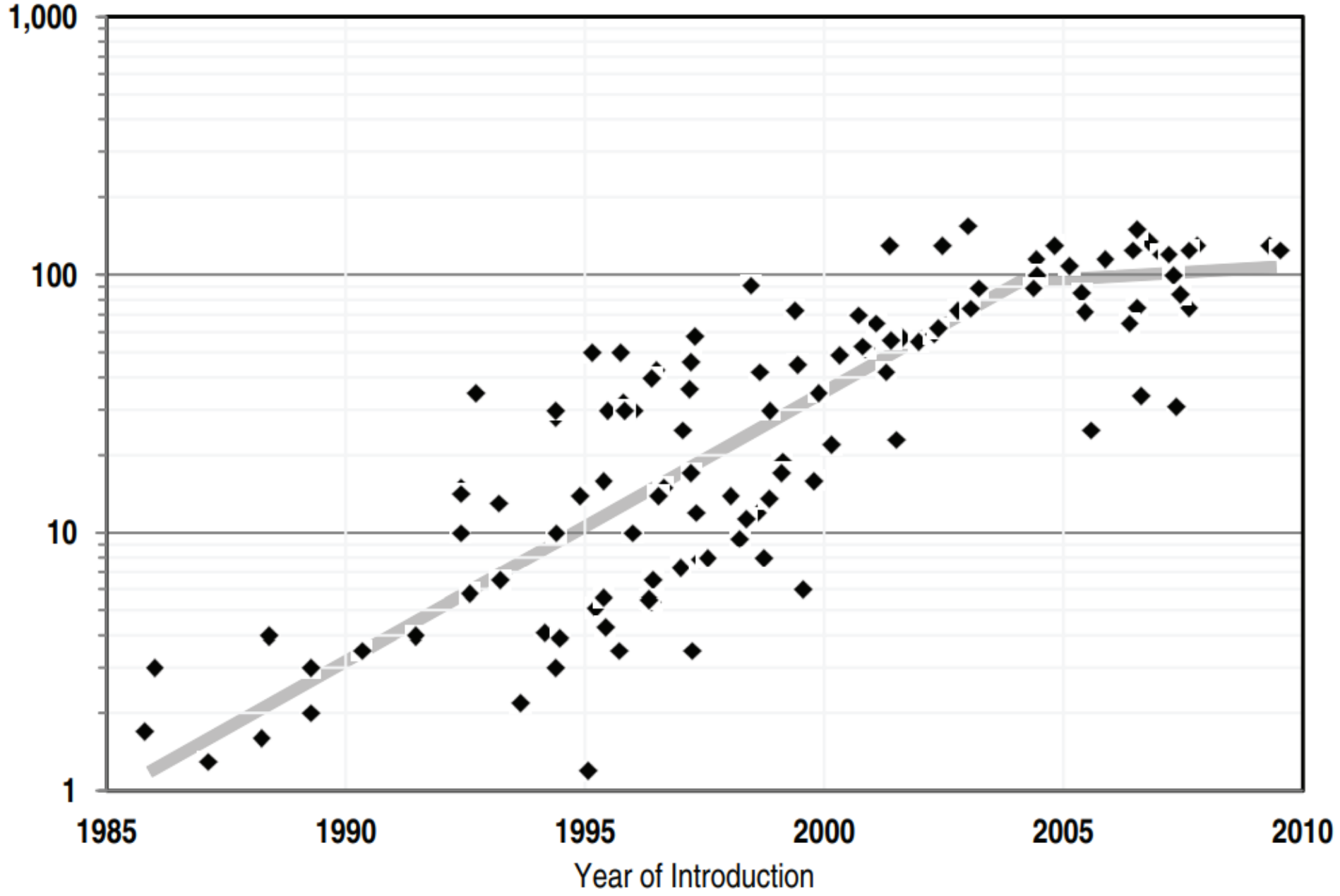


FIGURE 3.1 Microprocessor power dissipation (watts) over time (1985-2010).

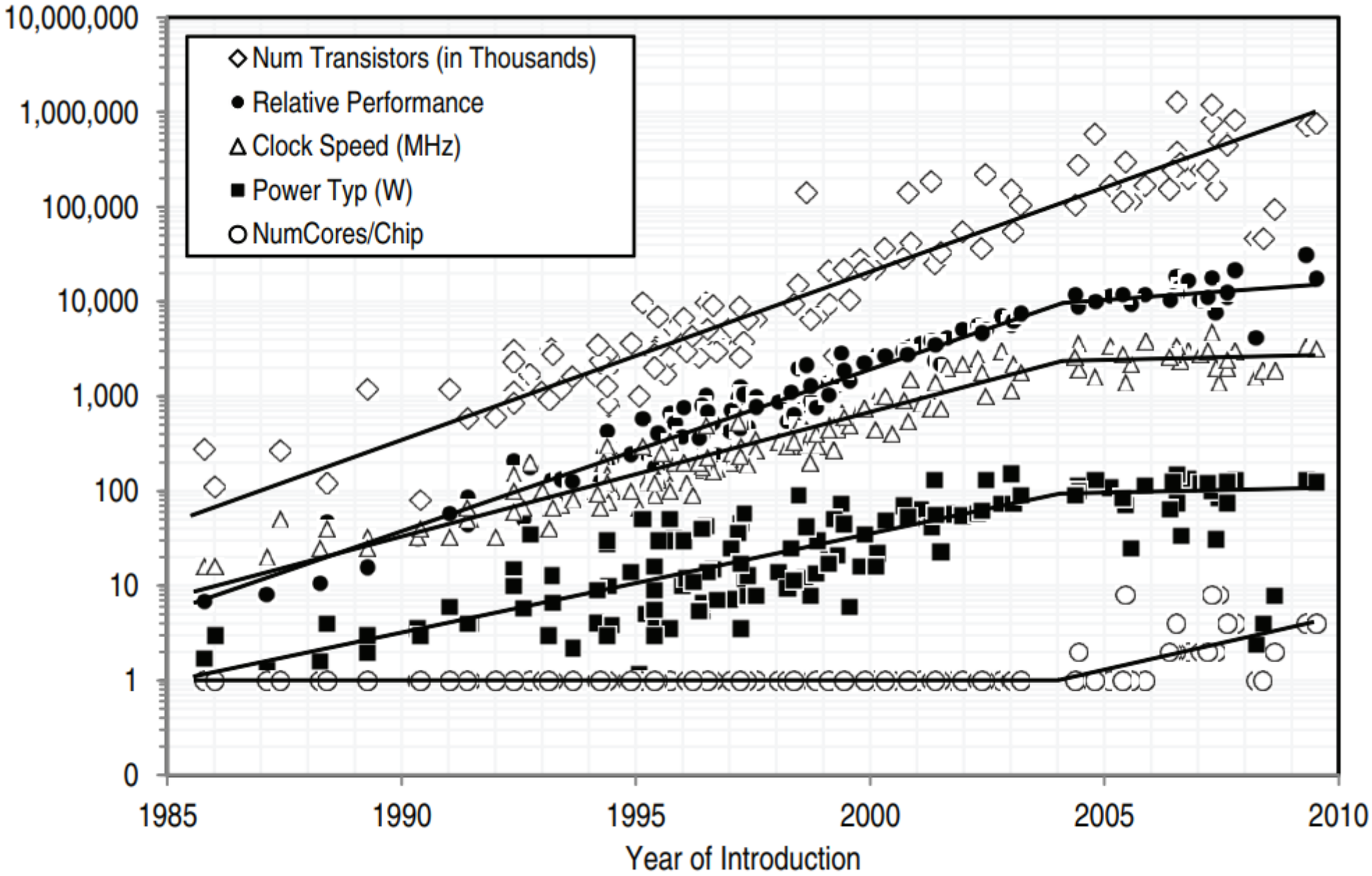


FIGURE 2.1 Transistors, frequency, power, performance, and cores over time

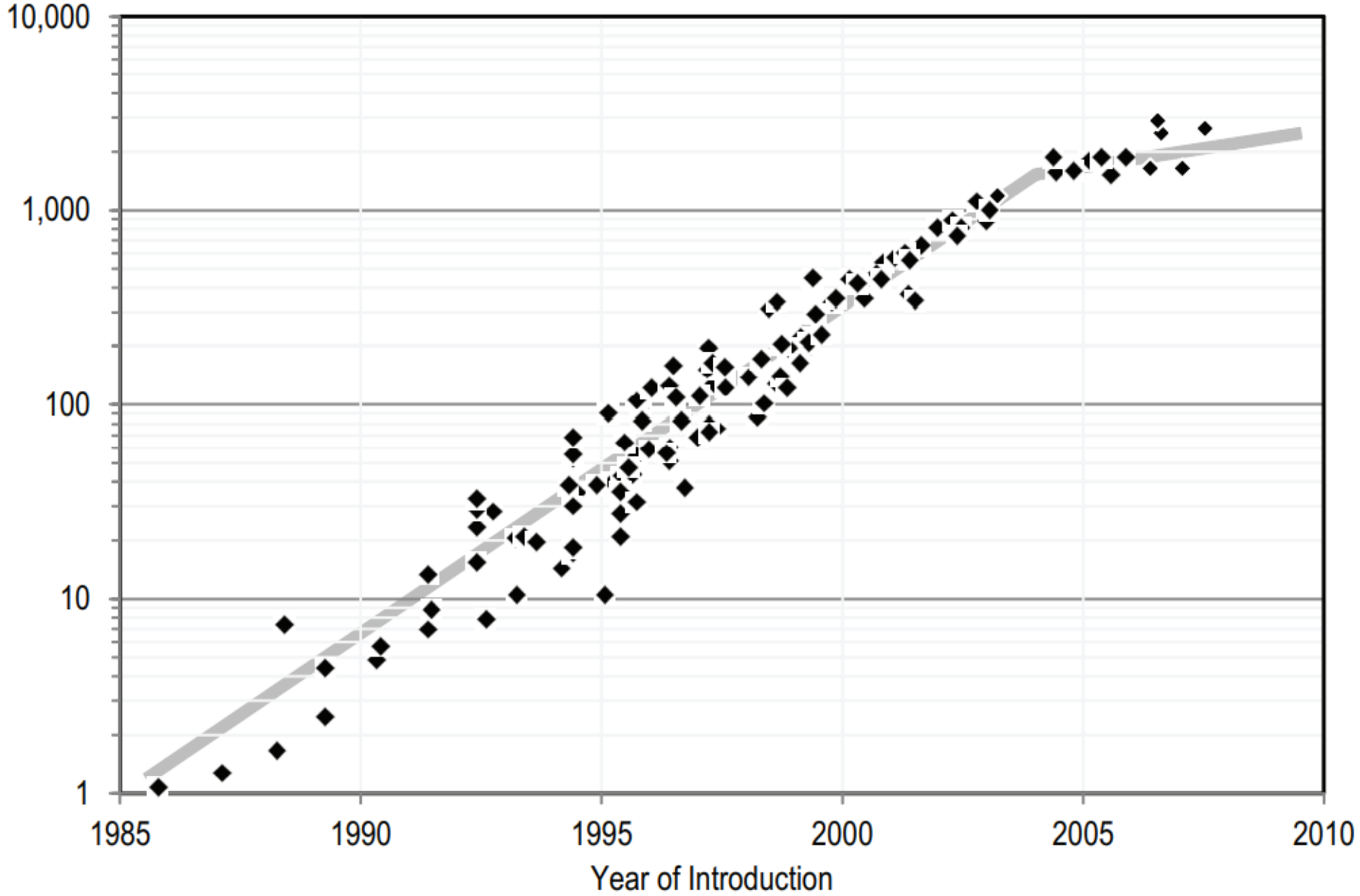


FIGURE A.1 Integer application performance (SPECint2000) over time (1985-2010).

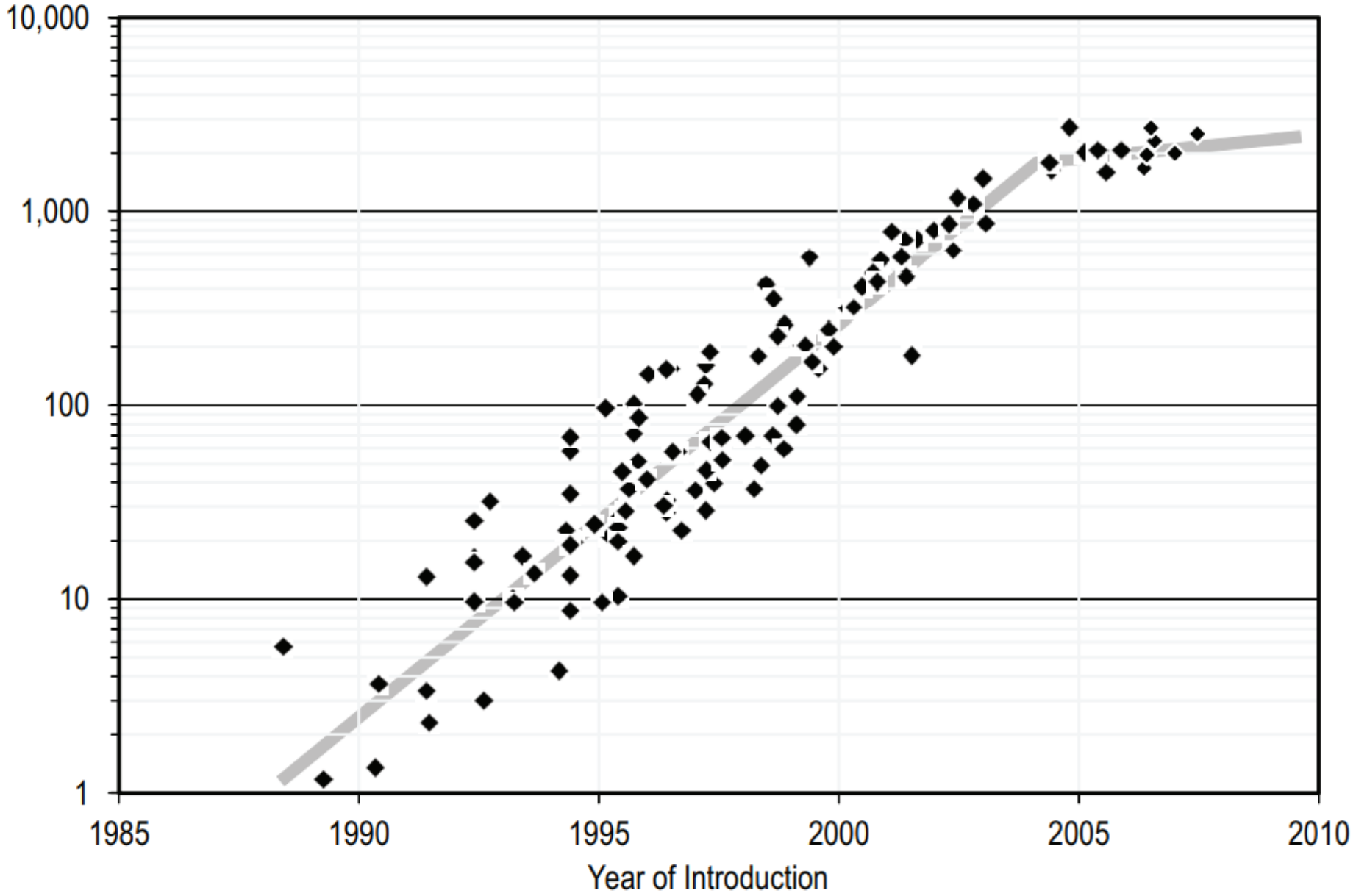


FIGURE A.2 Floating-point application performance (SPECfp2000) over time (1985-2010).

Dennard Scaling

Robert Dennard and colleagues described in 1974 a scaling methodology for metal-oxide-semiconductor field-effect transistors (MOSFETs) that would deliver consistent improvements in transistor area, performance, and power reduction.[3] The methodology called for the scaling of transistor gate length, gate width, gate oxide thickness, and supply voltage all by the same scale factor, and increasing channel doping by the inverse of the same scale factor (see Figure 1). The result would be transistors with smaller area, higher drive current (higher performance), and lower parasitic capacitance (lower active power). This method for scaling MOSFET transistors is generally referred to as "classic" or "traditional" scaling and was very successfully used by the industry up until the 130-nm generation in the early 2000s.

Device or circuit parameter	Scaling factor
Device dimension *tox*, *L*, *W*	$1/\kappa$
Doping concentration *Na*	κ
Voltage *V*	$1/\kappa$
Current *I*	$1/\kappa$
Capacitance $\varepsilon A/t$	$1/\kappa$
Delay time/circuit *VC/I*	$1/\kappa$
Power dissipation/circuit *VI*	$1/\kappa^2$
Power density *VI/A*	1

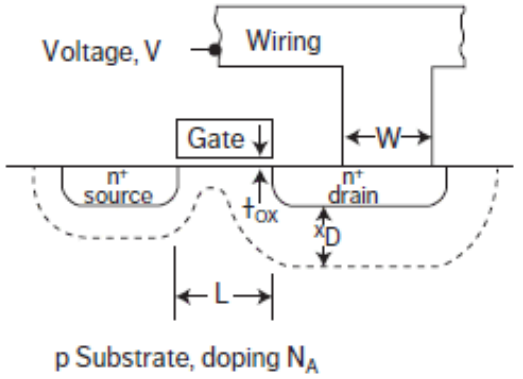


Figure 1.
Traditional MOSFET scaling as described by robert dennard.

Depiction of Dennard Scaling Effects

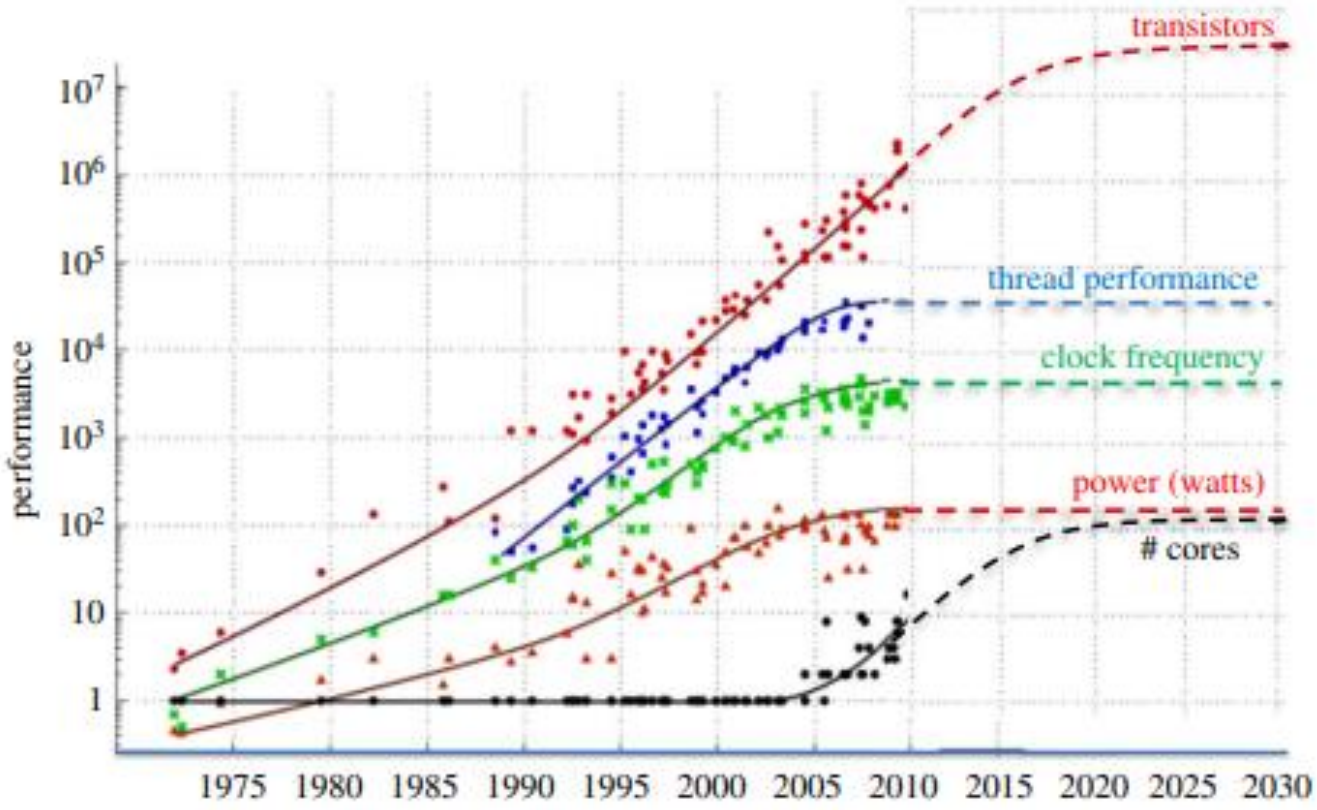
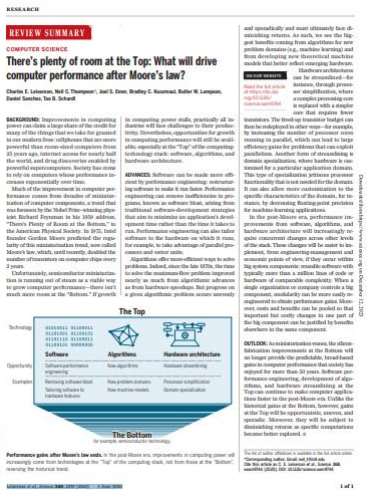


Figure 2. Sources of computing performance have been challenged by the end of Dennard scaling in 2004. All additional approaches to further performance improvements end in approximately 2025 due to the end of the roadmap for improvements to semiconductor lithography. Figure from Kunle Olukotun, Lance Hammond, Herb Sutter, Mark Horowitz and extended by John Shalf. (Online version in colour.)

Future Improvements in Computer Performance


RESEARCH

REVIEW SUMMARY

COMPUTER SCIENCE

There's plenty of room at the Top: What will drive computer performance after Moore's law?

Charles E. Leiserson, Neil C. Thompson, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, Tao B. Schardl

BACKGROUND: Improvements in computing power can claim a large share of the credit for many of the things that we take for granted in our modern lives: cellphones that are more powerful than room-sized computers from 25 years ago, internet access for nearly half the world, and drug discoveries enabled by powerful supercomputers. Society has come to rely on computers whose performance increases exponentially over time.

Much of the improvement in computer performance comes from decades of miniaturization of computer components, a trend that was foreseen by the Nobel Prize–winning physicist Richard Feynman in his 1959 address, "There's Plenty of Room at the Bottom," to the American Physical Society. In 1975, Intel founder Gordon Moore predicted the regularity of this miniaturization trend, now called Moore's law, which, until recently, doubled the number of transistors on computer chips every 2 years.

Unfortunately, semiconductor miniaturization is running out of steam as a viable way to grow computer performance—there isn't much more room at the "Bottom." If growth in computing power stalls, practically all industries will face challenges to their productivity. Nevertheless, opportunities for growth in computing performance will still be available, especially at the "Top" of the computing technology stack: software, algorithms, and hardware architecture.

ADVANCES: Software can be made more efficient by performance engineering: restructuring software to make it run faster. Performance engineering can remove inefficiencies in programs, known as software bloat, arising from traditional software-development strategies that aim to minimize an application's development time rather than the time it takes to run. Performance engineering can also tailor software to the hardware on which it runs, for example, to take advantage of parallel processors and vector units.

Algorithms offer more efficient ways to solve problems. Indeed, since the late 1970s, the time to solve the maximum-flow problem improved nearly as much from algorithmic advances as from hardware speedups. But progress on a given algorithmic problem occurs unevenly and sporadically and must ultimately face diminishing returns. As such, we see the biggest benefits coming from algorithms for new problem domains (e.g., machine learning) and from developing new theoretical machine models that better reflect emerging hardware.

Hardware architecture can be streamlined—for instance, through processor simplification, where a complex processing core is replaced with a simpler core that requires fewer transistors. The freed-up transistor budget can then be redeployed in other ways—for example, by increasing the number of processor cores running in parallel, which can lead to large efficiency gains for problems that can exploit parallelism. Another form of streamlining is domain specialization, where hardware is customized for a particular application domain. This type of specialization jettisons processor functionality that is not needed for the domain. It can also allow more customization to the specific characteristics of the domain, for instance, by decreasing floating-point precision for machine-learning applications.

In the post-Moore era, performance improvements from software, algorithms, and hardware architecture will increasingly require concurrent changes across other levels of the stack. These changes will be easier to implement, from engineering-management and economic points of view, if they occur within big system components: reusable software with typically more than a million lines of code or hardware of comparable complexity. When a single organization or company controls a big component, modularity can be more easily reengineered to obtain performance gains. Moreover, costs and benefits can be pooled so that important but costly changes in one part of the big component can be justified by benefits elsewhere in the same component.

OUTLOOK: As miniaturization wanes, the silicon-fabrication improvements at the Bottom will no longer provide the predictable, broad-based gains in computer performance that society has enjoyed for more than 50 years. Software performance engineering, development of algorithms, and hardware streamlining at the Top can continue to make computer applications faster in the post-Moore era. Unlike the historical gains at the Bottom, however, gains at the Top will be opportunistic, uneven, and episodic. Moreover, they will be subject to diminishing returns as specific computations become better explored. ■

Performance gains after Moore's law ends. In the post-Moore era, improvements in computing power will increasingly come from technologies at the "Top" of the computing stack, not from those at the "Bottom", reversing the historical trend.

Ref: Leiserson et al., Science 368, 1079 (2020) 5 June 2020

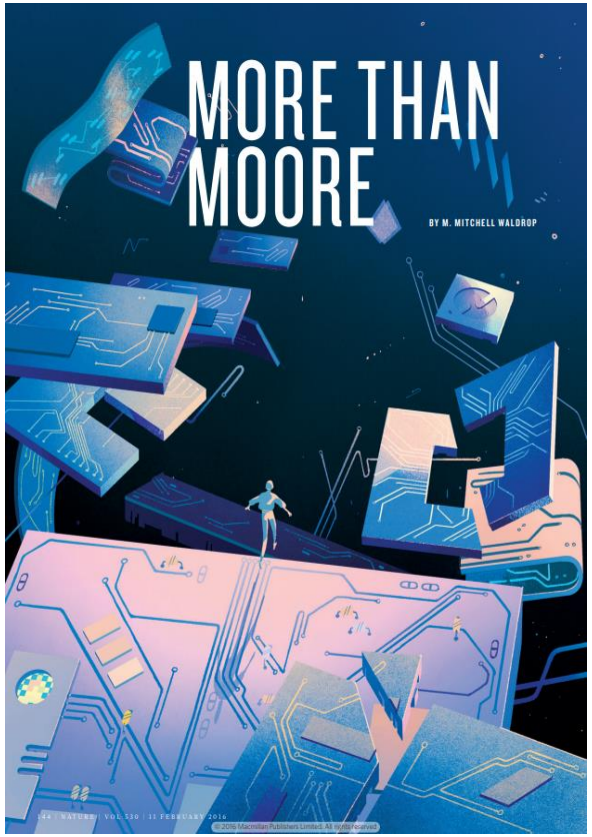
Example of Performance Gains through Architecture

Table 1. Speedups from performance engineering a program that multiplies two 4096-by-4096 matrices. Each version represents a successive refinement of the original Python code. "Running time" is the running time of the version. "GFLOPS" is the billions of 64-bit floating-point operations per second that the version executes. "Absolute speedup" is time relative to Python, and "relative speedup," which we show with an additional digit of precision, is time relative to the preceding line. "Fraction of peak" is GFLOPS relative to the computer's peak 835 GFLOPS. See Methods for more details.

Version	Implementation	Running time (s)	GFLOPS	Absolute speedup	Relative speedup	Fraction of peak (%)
1	Python	25,552.48	0.005	1	–	0.00
2	Java	2,372.68	0.058	11	10.8	0.01
3	C	542.67	0.253	47	4.4	0.03
4	Parallel loops	69.80	1.969	366	7.8	0.24
5	Parallel divide and conquer	3.80	36.180	6,727	18.4	4.33
6	plus vectorization	1.10	124.914	23,224	3.5	14.96
7	plus AVX intrinsics	0.41	337.812	62,806	2.7	40.45

AVX = Intel Advanced Vector Extension

Look at some recent research papers for inspiration



THEME ARTICLE: MICROPROCESSOR AT 50

The 50 Year History of the Microprocessor as Five Technology Eras

John L. Hennessy, *Stanford University, Stanford, CA, 94305, USA*

The evolution of the microprocessor can be organized into five eras, each distinguished by common trends in the evolution of microprocessors. Most of these eras are around ten years and represent a shift from the previous era. I have had the privilege of being involved in some way for roughly 48 of the 50 years, so this is also a somewhat personal view.

FIRST DECADE 1971–1981

This decade, which began with the birth of the Intel 4004 in 1971, was dominated by three trends:

1. an increase in width as Moore's law-enabled processors to go from 4 to 8 to 16 and eventually 32 bits;
2. a rapid increase in instruction sets, often motivated by assembly language examples and enabled by microcode implementations (see the early Intel and Zilog microprocessors);
3. a rapid increase in clock speeds enabled by faster transistors.

The emergence of the personal computer (Apple II in 1977 and IBM PC in 1979) enabled the shrinkwrap software industry and reinforced the importance of object code compatibility, which the first microprocessors did not exhibit. The Motorola 68000, which appeared in production in the late 1980s, was the first 32-bit microprocessor and offered many of the features associated with minicomputers.

RISC AND PIPELINING YEARS 1981–1995

As Moore's law progressed, it seemed likely that microprocessors would evolve into full blown computers. The accompanying growth in DRAM capacity (from 1 Kib in 1971 to 16 Kib in 1981) reduced the need to hand-optimize code and program in assembly language. The subsequent movement to higher level languages inspired the groups at IBM, Berkeley, and Stanford to explore what became the RISC ideas. In addition to targeting compiler output, rather than handcrafted assembly language, they also emphasized elimination of microcode and compilation to an efficient hardware implementation.

The RISC ideas led to an explosion in the use of pipelining in microprocessors, which generated a rapid increase in clock rate performance. This era was characterized by incredible annual performance growth of approximately 1.5 times, enabled by the inclusion of caches and much faster clocks.

The capstone events in this era were the introduction of 64-bit processors (the R4000, followed by the DEC Alpha, and others) and a growth in pipeline depth from 5 to 7–10 or more stages, which led to clock rates rivaling ECL mainframes.

INTENSIVE ILP YEARS 1995–2005

The third era is characterized by an intensive focus on exploiting instruction-level parallelism and trying to reduce the clock cycles per instruction to less than 1. The ILP-intensive processors fall into two broad categories: superscalar and very long instruction word (VLIW). The superscalar processors used a combination of hardware techniques and software scheduling to issue more than one instruction per clock, while the VLIW approach relied on little hardware support and intensive compiler scheduling to organize independent operations into issue packets. The VLIW approach did not succeed in the end, due to a variety of factors, most importantly the inability to achieve high performance on less structured integer programs.

The initial superscalar processors (such as the MIPS R8000, PowerPC 604, and later Intel Pentium) used static scheduling, meaning that instruction issue was blocked if the next instruction's operands were not yet available. This approach was rapidly followed by a shift to dynamic scheduling (allowing instructions to be executed out of order) and speculation (allowing instructions to be executed before a preceding branch

0272-1732 © 2021 IEEE
Digital Object Identifier 10.1109/MM.2021.3112301
Date of current version 19 November 2021.


REVIEW SUMMARY

COMPUTER SCIENCE

There's plenty of room at the Top: What will drive computer performance after Moore's law?

Charles E. Leiserson, Neil C. Thompson*, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, Tao B. Schardl

turing lecture

DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

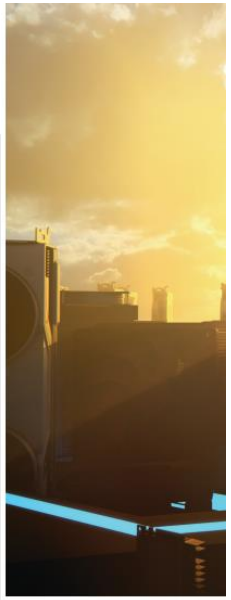
BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

WE BEGAN OUR Turing Lecture June 4, 2018[11] with a review of computer architecture since the 1960s. In addition to that review, here, we highlight current challenges and identify future opportunities, projecting another golden age for the field of computer architecture in the next decade, much like the 1980s when we did the research that led to our award, delivering gains in cost, energy, and security, as well as performance.

"Those who cannot remember the past are condemned to repeat it." —George Santayana, 1905

Software talks to hardware through a vocabulary called an instruction set architecture (ISA). By the early 1960s, IBM had four incompatible lines of computers, each with its own ISA, software stack, I/O system, and market niche—targeting small business, large business, scientific, and real time, respectively. IBM



engineers, including ACM A.M. Turing Award laureate Fred Brooks, Jr., thought they could create a single ISA that would efficiently unify all four of these ISA bases.

They needed a technical solution for how computers as inexpensive as

» key insights

- Software advances can inspire architecture innovation.
- Elevating the hardware/software interface creates opportunities for architecture innovation.
- The marketplace ultimately settles architecture debates.

Processor Driver vs Memory Driven Computing

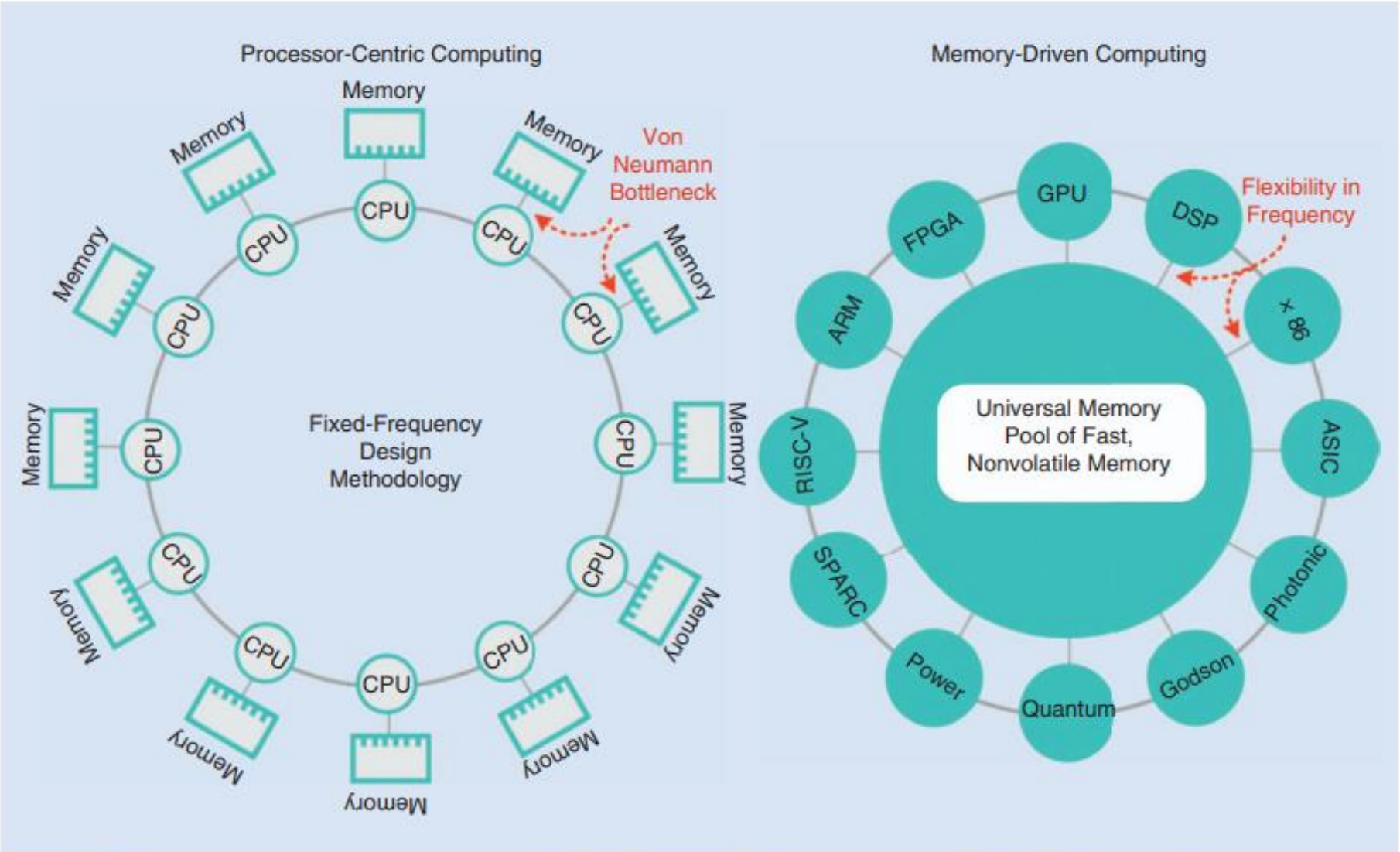


FIGURE 7: Memory-driven computing requires flexible clocking [59]. DSP: digital signal processor. ASIC: application-specific integrated circuit; RISC-V: reduced instruction set computing.

Topics

1. Discussion on Moore's Law, through paper 'MORE THAN MOORE' by M. Mitchell Waldrop, published In Nature, February 2016
2. Computer Performance Graphs from National Academy, USA, report
3. Introduction to the specialization area of 'Computer Architecture' through paper by Hennessy and Patterson, 'A New Golden Age for Computer Architecture', published in Communications of the ACM, February 2019, pages 48 to 60.

   Important Directions for Computers of the future:

   a. Moore's Law failing to keep up
   b. Domain Specific Architectures
   c. Enhanced security features inside microprocessors
   d. Open Instruction Sets and Extension of Instruction Sets through Customized Accelerators
   e. Agile Hardware Design for Microprocessors
   f. Combination of Domain Specific Languages and Domain Specific Architectures

4. Intel Processor Timeline
5. Reviewed course outline including detailed topics and grading breakup of lectures and labs.

Video of Turing Lecture by Patterson, 2019

David Patterson - A New Golden Age for Computer Architecture: History, Challenges and Opportunities – YouTube