Computer Architecture
CS 621 / EE 520
Spring 2025

Lecture 1

Shahid Masud

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

Old Approach

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

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

New View

The task the computer designer faces is a complex one: determine what attributes are important for a new computer, then design a computer to maximize performance and energy efficiency while staying within cost, power, and availability constraints. This task has many aspects, including instruction set design, functional organization, logic design, and implementation. The implementation may encompass integrated circuit design, packaging, power, and cooling. Optimizing the design requires familiarity with a very wide range of technologies, from compilers and operating systems to logic design and packaging.

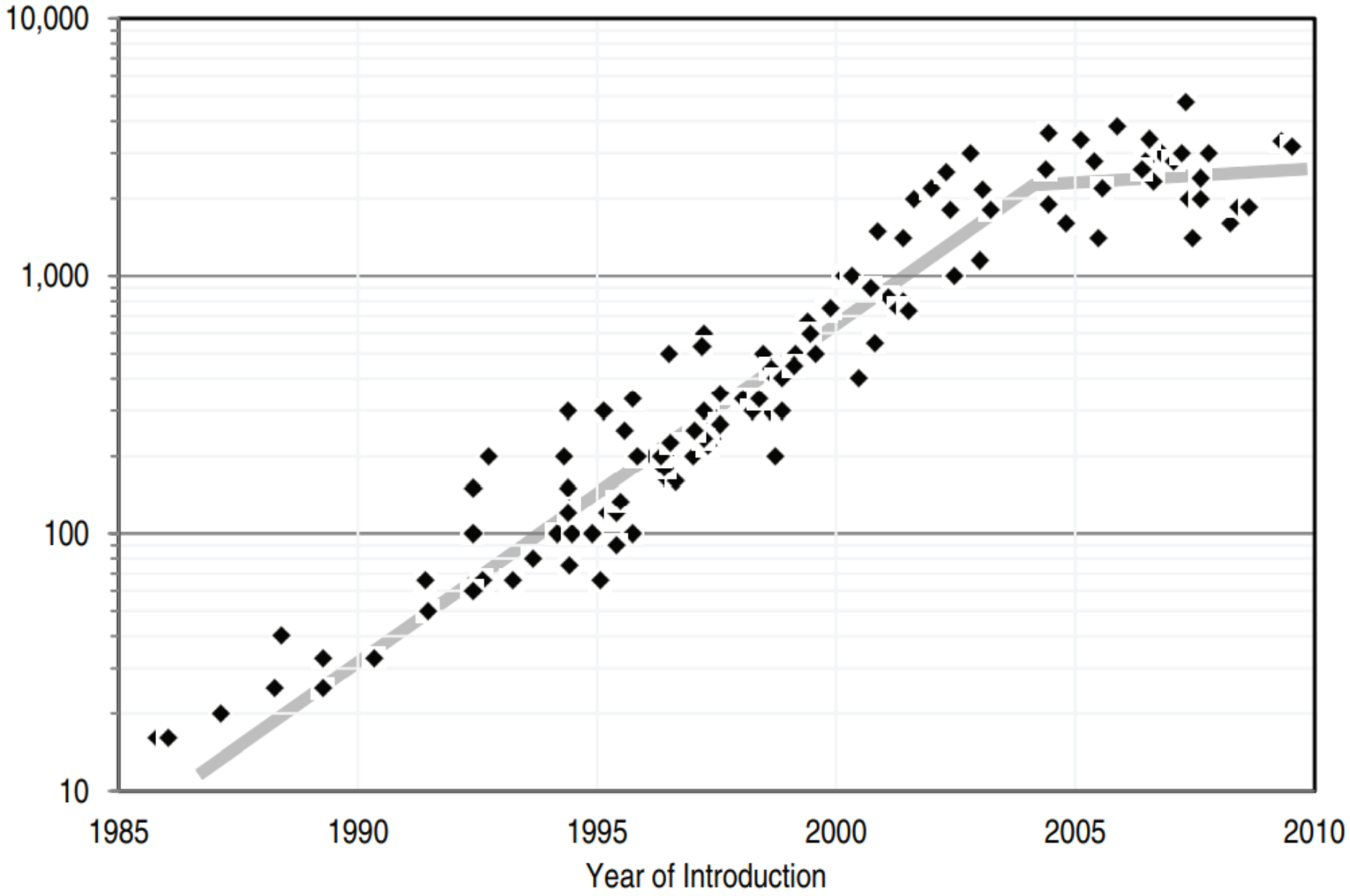


FIGURE 3.3 Microprocessor-clock frequency (MHz) over time (1985-2010).

Clock Rate Improvement

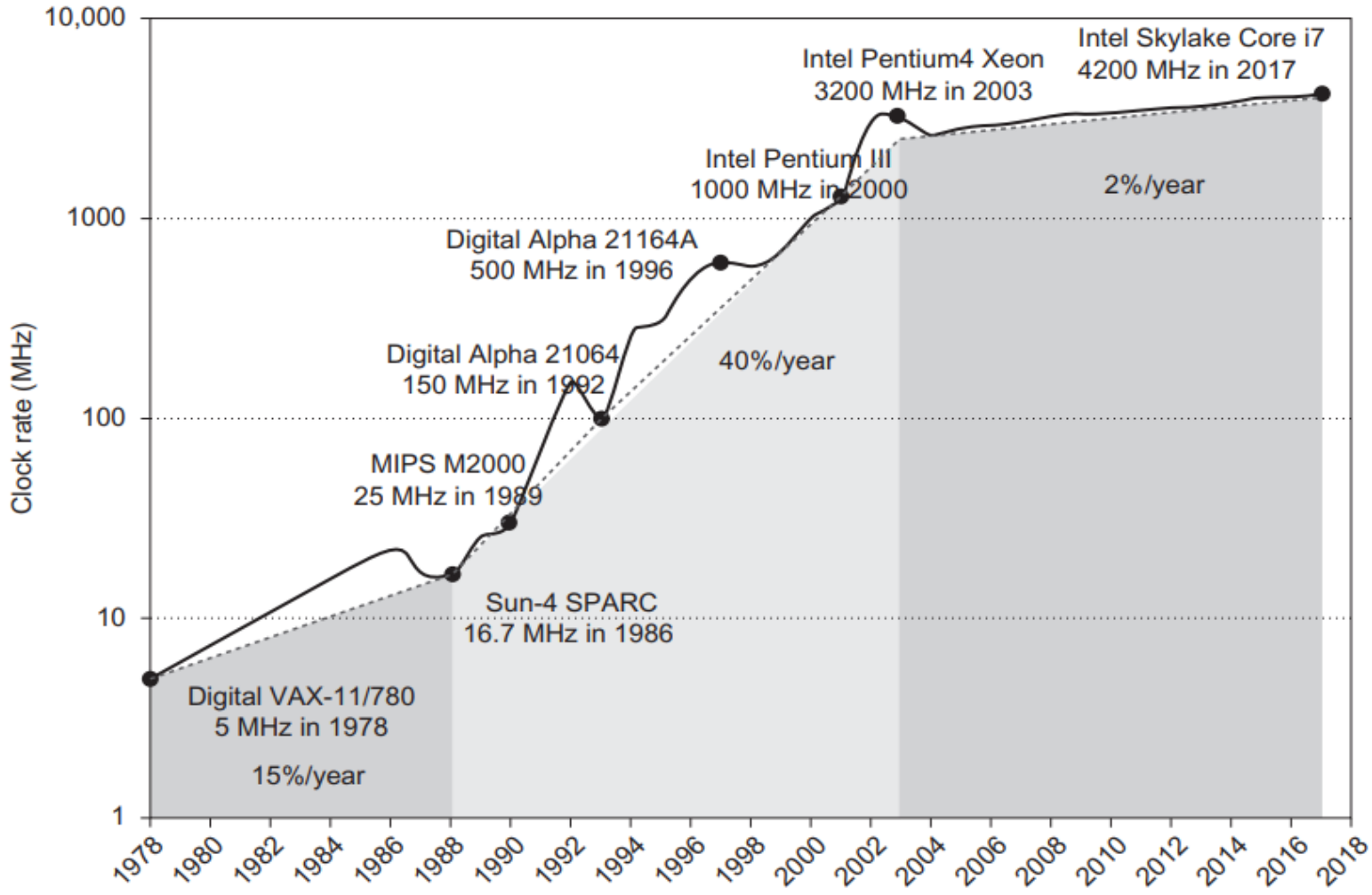


Figure 1.11 Growth in clock rate of microprocessors in Figure 1.1. Between 1978 and 1986, the clock rate improved less than 15% per year while performance improved by 22% per year. During the "renaissance period" of 52% performance improvement per year between 1986 and 2003, clock rates shot up almost 40% per year. Since then, the clock rate has been nearly flat, growing at less than 2% per year, while single processor performance improved recently at just 3.5% per year.

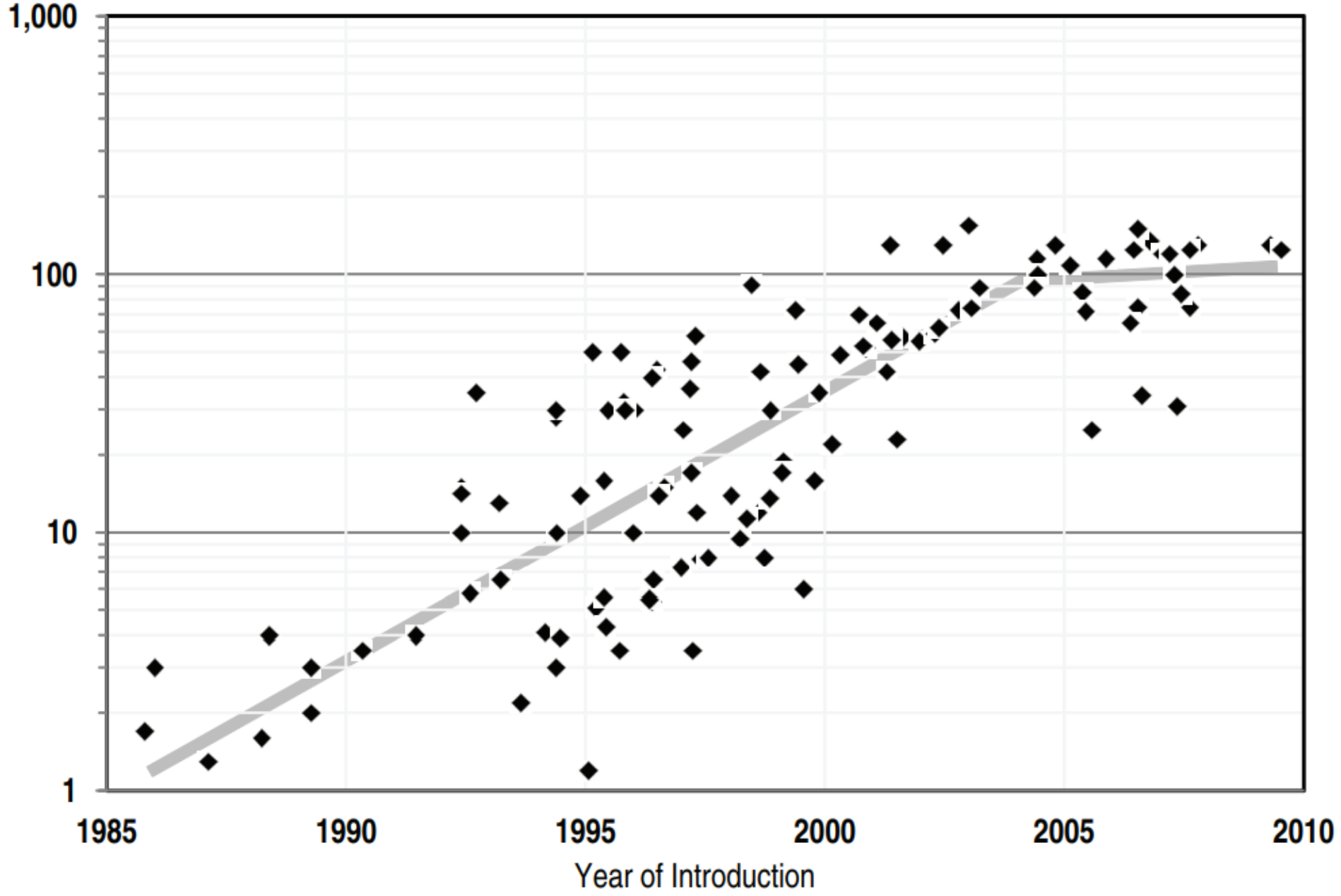


FIGURE 3.1 Microprocessor power dissipation (watts) over time (1985-2010).

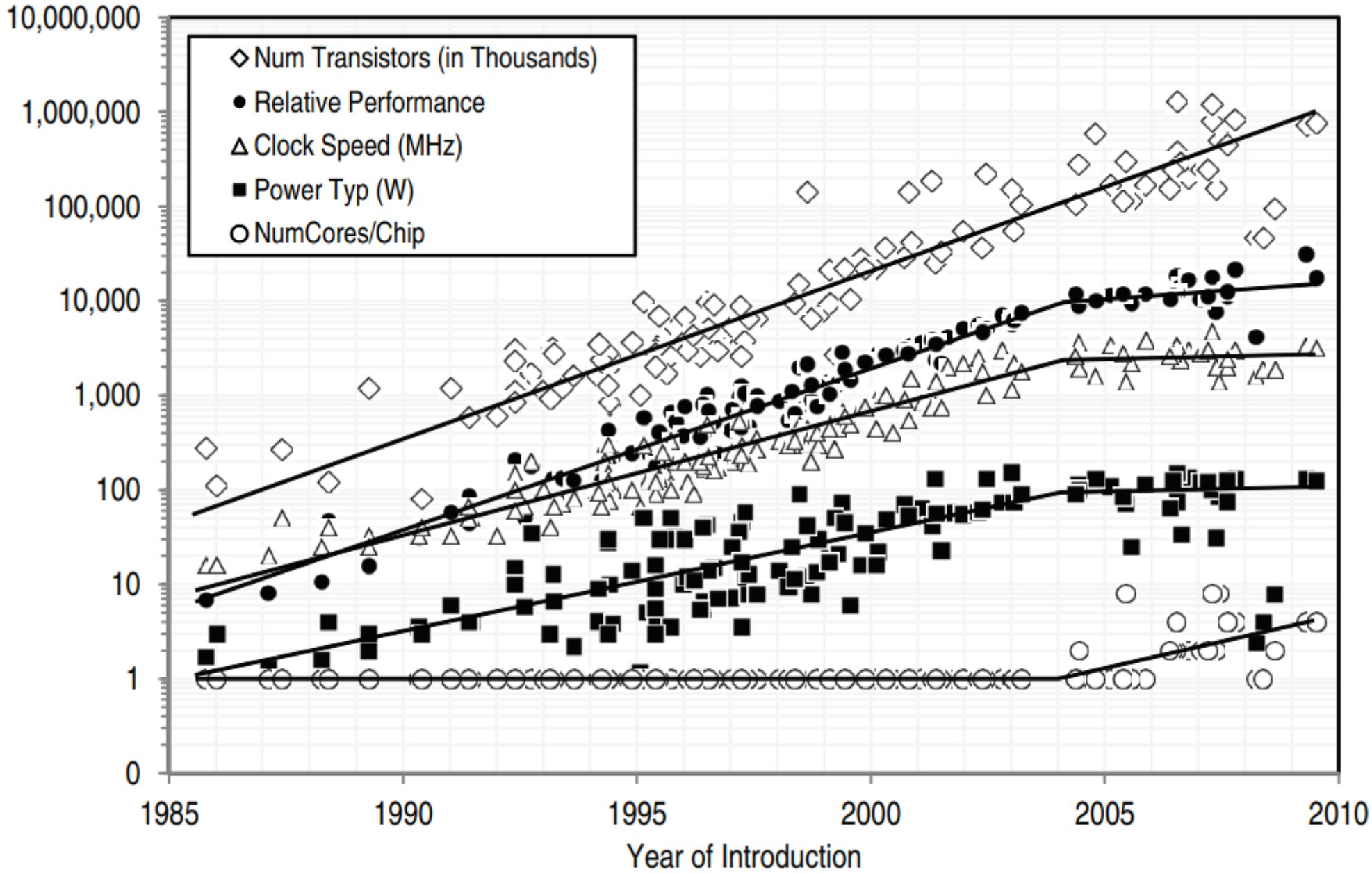


FIGURE 2.1 Transistors, frequency, power, performance, and cores over time

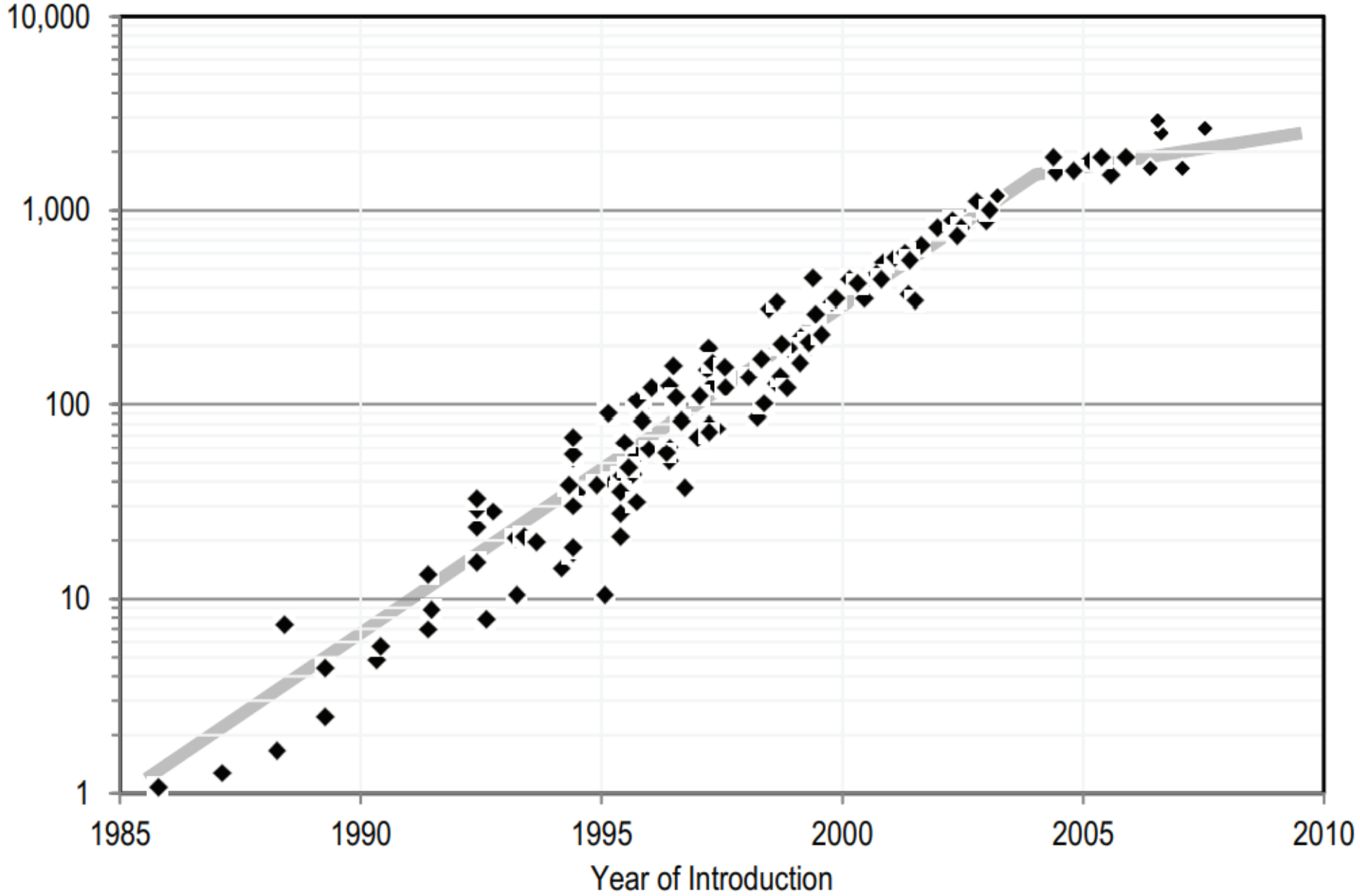


FIGURE A.1 Integer application performance (SPECint2000) over time (1985-2010).

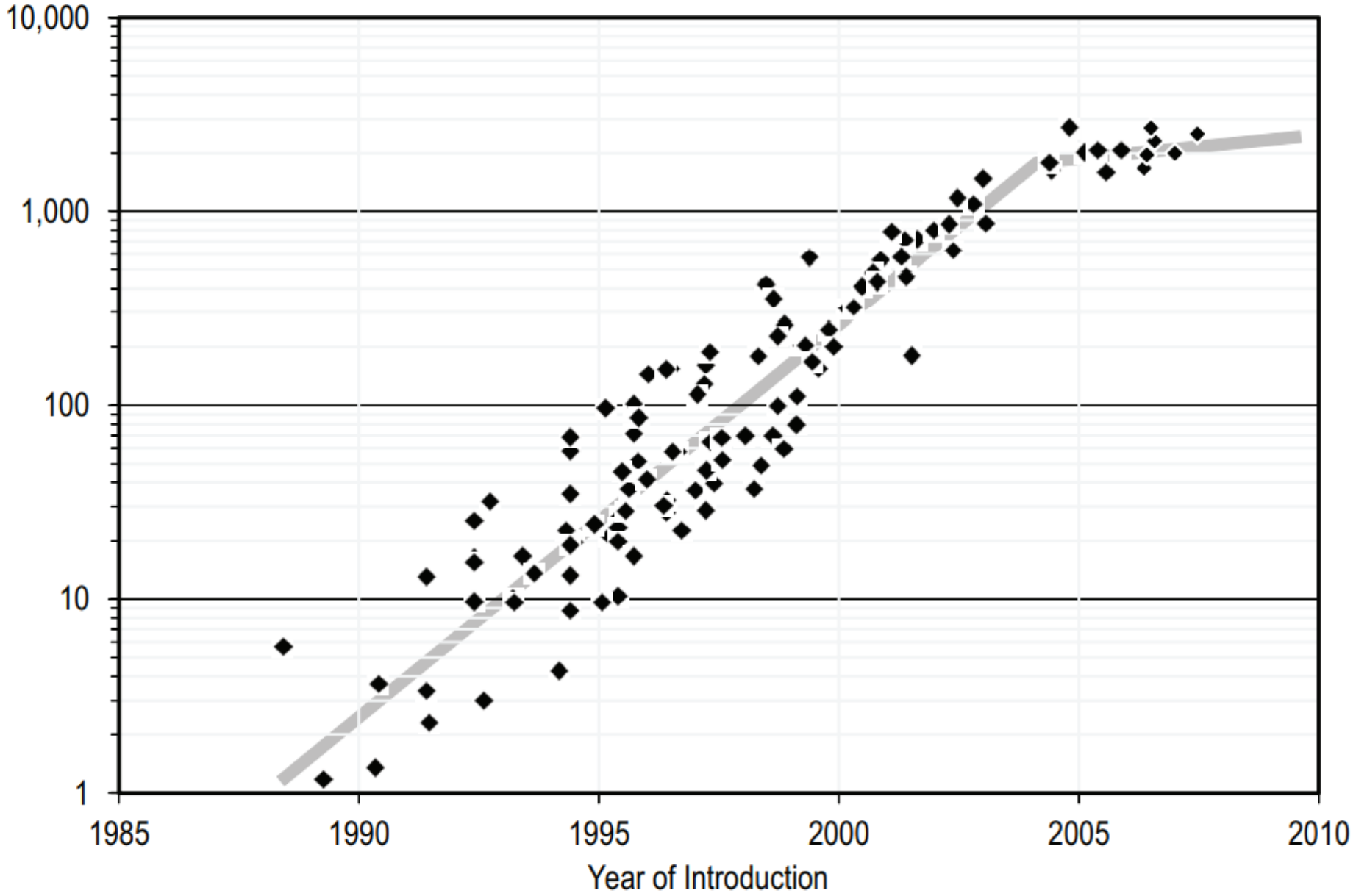


FIGURE A.2 Floating-point application performance (SPECfp2000) over time (1985-2010).

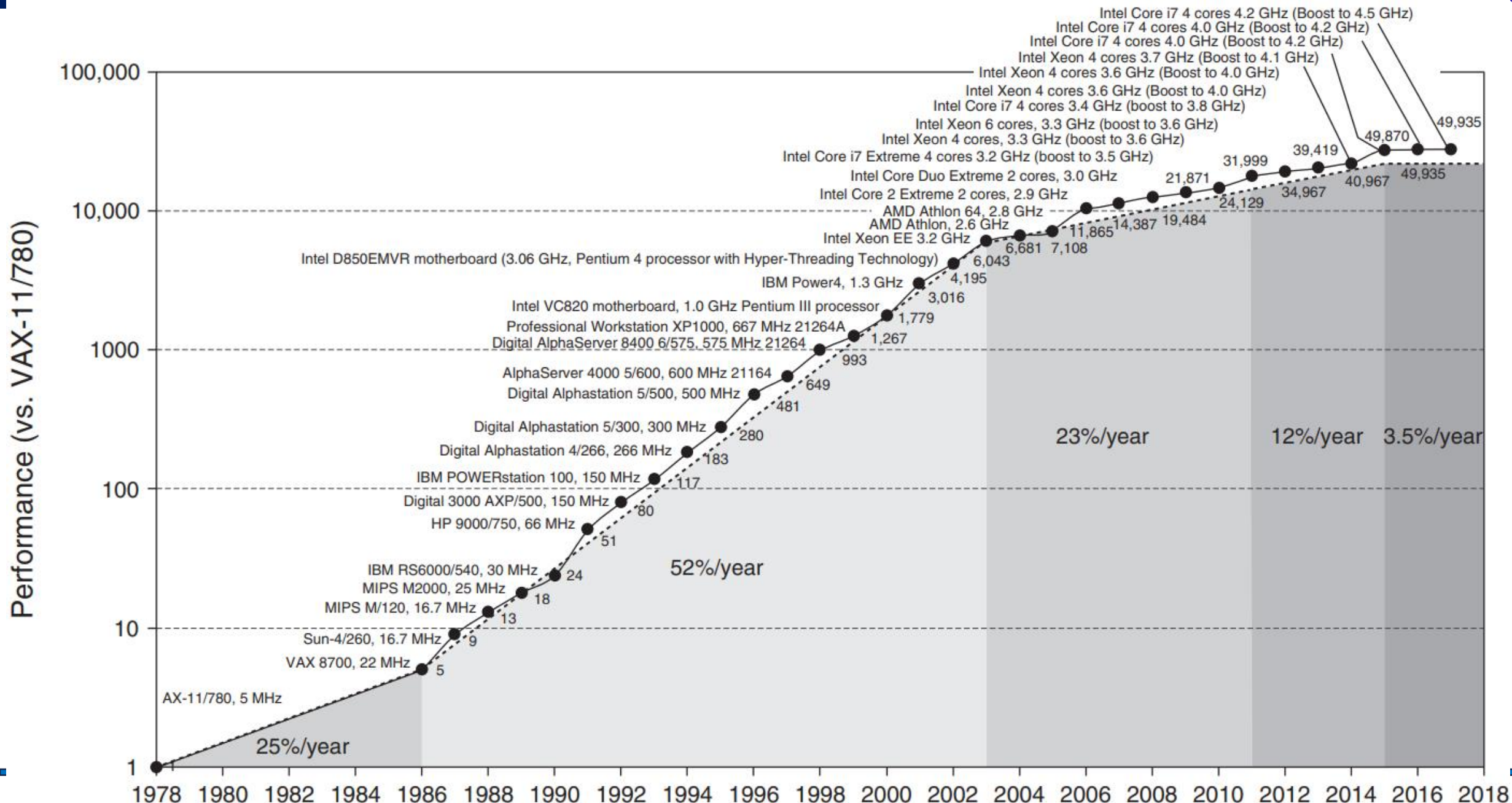
Technology Road-Blocks in last few years

Much of the improvement in computer performance over the last 40 years has been provided by computer architecture advancements that have leveraged Moore's Law and Dennard scaling to build larger and more parallel systems. Moore's Law is the observation that the maximum number of transistors in an integrated circuit doubles approximately every two years. Dennard scaling refers to the reduction of MOS supply voltage in concert with the scaling of feature sizes, so that as transistors get smaller, their power density stays roughly constant. With the end of Dennard scaling a decade ago, and the recent slowdown of Moore's Law due to a combination of physical limitations and economic factors, the sixth edition of the preeminent textbook for our field couldn't be more timely. Here are some reasons.

What is Post Moore's Law Computing

First, because domain-specific architectures can provide equivalent performance and power benefits of three or more historical generations of Moore's Law and Dennard scaling, they now can provide better implementations than may ever be possible with future scaling of general-purpose architectures. And with the diverse application space of computers today, there are many potential areas for architectural innovation with domain-specific architectures. Second, high-quality implementations of open-source architectures now have a much longer lifetime due to the slowdown in Moore's Law. This gives them more opportunities for continued optimization and refinement, and hence makes them more attractive. Third, with the slowing of Moore's Law, different technology components have been scaling heterogeneously. Furthermore, new technologies such as 2.5D stacking, new nonvolatile memories, and optical interconnects have been developed to provide more than Moore's Law can supply alone. To use these new technologies and nonhomogeneous scaling effectively, fundamental design decisions need to be reexamined from first principles. Hence it is important for

Computer Performance Improvement



Explanation of Figure

Figure 1.1 Growth in processor performance over 40 years. This chart plots program performance relative to the VAX 11/780 as measured by the SPEC integer benchmarks (see Section 1.8). Prior to the mid-1980s, growth in processor performance was largely technology-driven and averaged about 22% per year, or doubling performance every 3.5 years. The increase in growth to about 52% starting in 1986, or doubling every 2 years, is attributable to more advanced architectural and organizational ideas typified in RISC architectures. By 2003 this growth led to a difference in performance of an approximate factor of 25 versus the performance that would have occurred if it had continued at the 22% rate. In 2003 the limits of power due to the end of Dennard scaling and the available instruction-level parallelism slowed uniprocessor performance to 23% per year until 2011, or doubling every 3.5 years. (The fastest SPECintbase performance since 2007 has had automatic parallelization turned on, so uniprocessor speed is harder to gauge. These results are limited to single-chip systems with usually four cores per chip.) From 2011 to 2015, the annual improvement was less than 12%, or doubling every 8 years in part due to the limits of parallelism of Amdahl's Law. Since 2015, with the end of Moore's Law, improvement has been just 3.5% per year, or doubling every 20 years! Performance for floating-point-oriented calculations follows the same trends, but typically has 1% to 2% higher annual growth in each shaded region. Figure 1.11 on page 27 shows the improvement in clock rates for these same eras. Because SPEC has changed over the years, performance of newer machines is estimated by a scaling factor that relates the performance for different versions of SPEC: SPEC89, SPEC92, SPEC95, SPEC2000, and SPEC2006. There are too few results for SPEC2017 to plot yet.

A Modern Computer System

The nature of applications is also changing. Speech, sound, images, and video are becoming increasingly important, along with predictable response time that is so critical to the user experience. An inspiring example is Google Translate. This application lets you hold up your cell phone to point its camera at an object, and the image is sent wirelessly over the Internet to a warehouse-scale computer (WSC) that recognizes the text in the photo and translates it into your native language. You can also speak into it, and it will translate what you said into audio output in another language. It translates text in 90 languages and voice in 15 languages.

Dannard Scaling

In 1974 Robert Dennard observed that power density was constant for a given area of silicon even as you increased the number of transistors because of smaller dimensions of each transistor. Remarkably, transistors could go faster but use less power. *Dennard scaling* ended around 2004 because current and voltage couldn't keep dropping and still maintain the dependability of integrated circuits.

Depiction of Dennard Scaling Effects

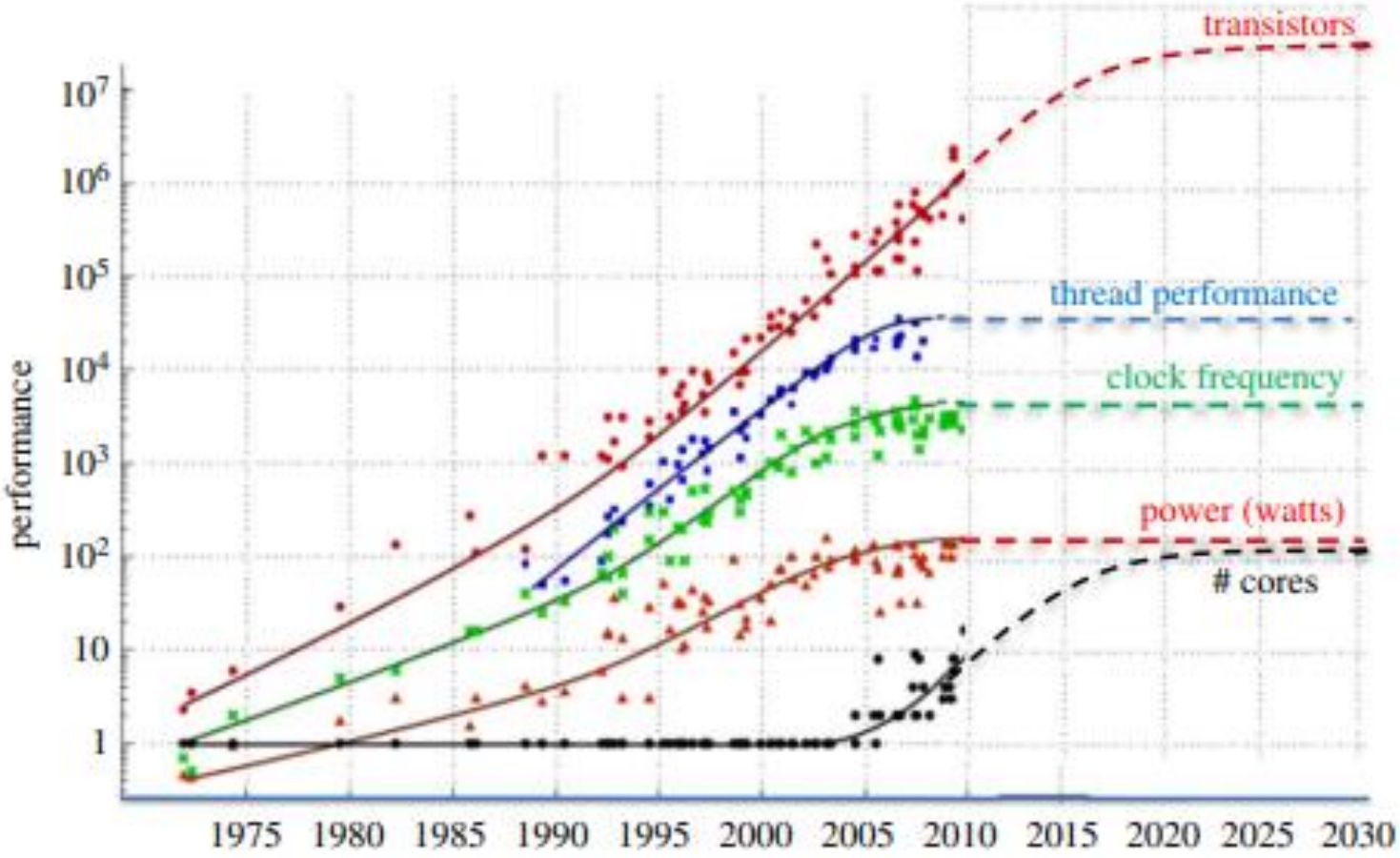


Figure 2. Sources of computing performance have been challenged by the end of Dennard scaling in 2004. All additional approaches to further performance improvements end in approximately 2025 due to the end of the roadmap for improvements to semiconductor lithography. Figure from Kunle Olukotun, Lance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

Amdahl's Law – Parallel or Accelerator

Amdahl's Law (Section 1.9) prescribes practical limits to the number of useful cores per chip. If 10% of the task is serial, then the maximum performance benefit from parallelism is 10 no matter how many cores you put on the chip.

Amdahl's Law defines the *speedup* that can be gained by using a particular feature. What is speedup? Suppose that we can make an enhancement to a computer that will improve performance when it is used. Speedup is the ratio

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

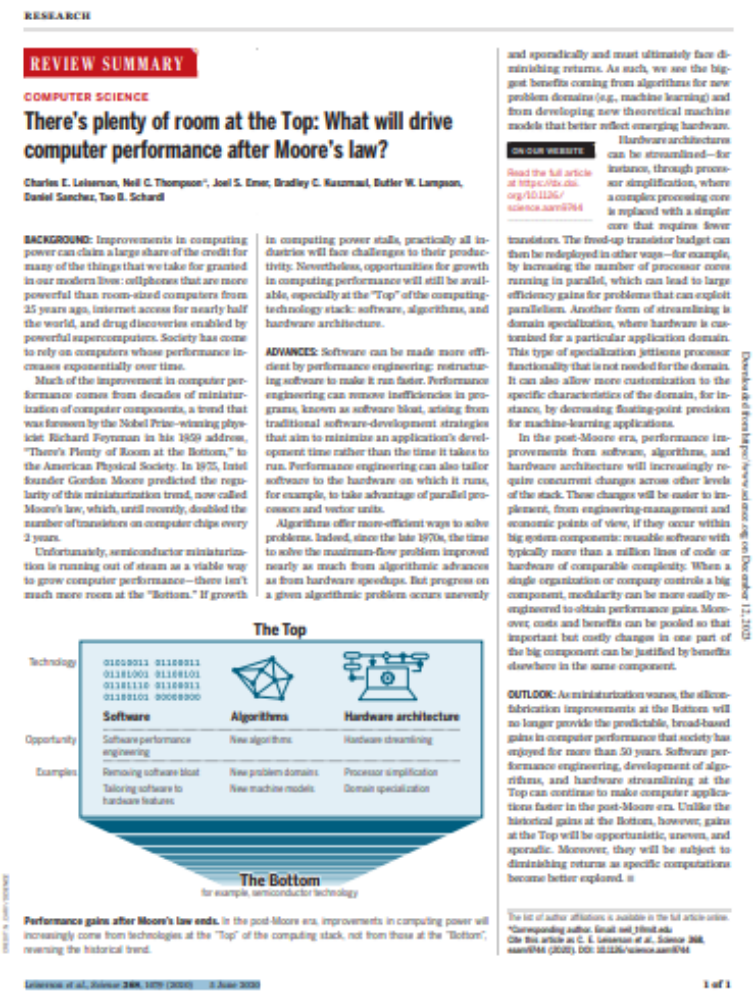
Moore's Law – Constraints on Computers

The second observation that ended recently is *Moore's Law*. In 1965 Gordon Moore famously predicted that the number of transistors per chip would double every year, which was amended in 1975 to every two years. That prediction lasted for about 50 years, but no longer holds. For example, in the 2010 edition of this book, the most recent Intel microprocessor had 1,170,000,000 transistors. If Moore's Law had continued, we could have expected microprocessors in 2016 to have 18,720,000,000 transistors. Instead, the equivalent Intel microprocessor has just 1,750,000,000 transistors, or off by a factor of 10 from what Moore's Law would have predicted.

The combination of

- transistors no longer getting much better because of the slowing of Moore's Law and the end of Dinnard scaling,
- the unchanging power budgets for microprocessors,
- the replacement of the single power-hungry processor with several energy-efficient processors, and
- the limits to multiprocessing to achieve Amdahl's Law

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 architectures 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 re-engineered 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.

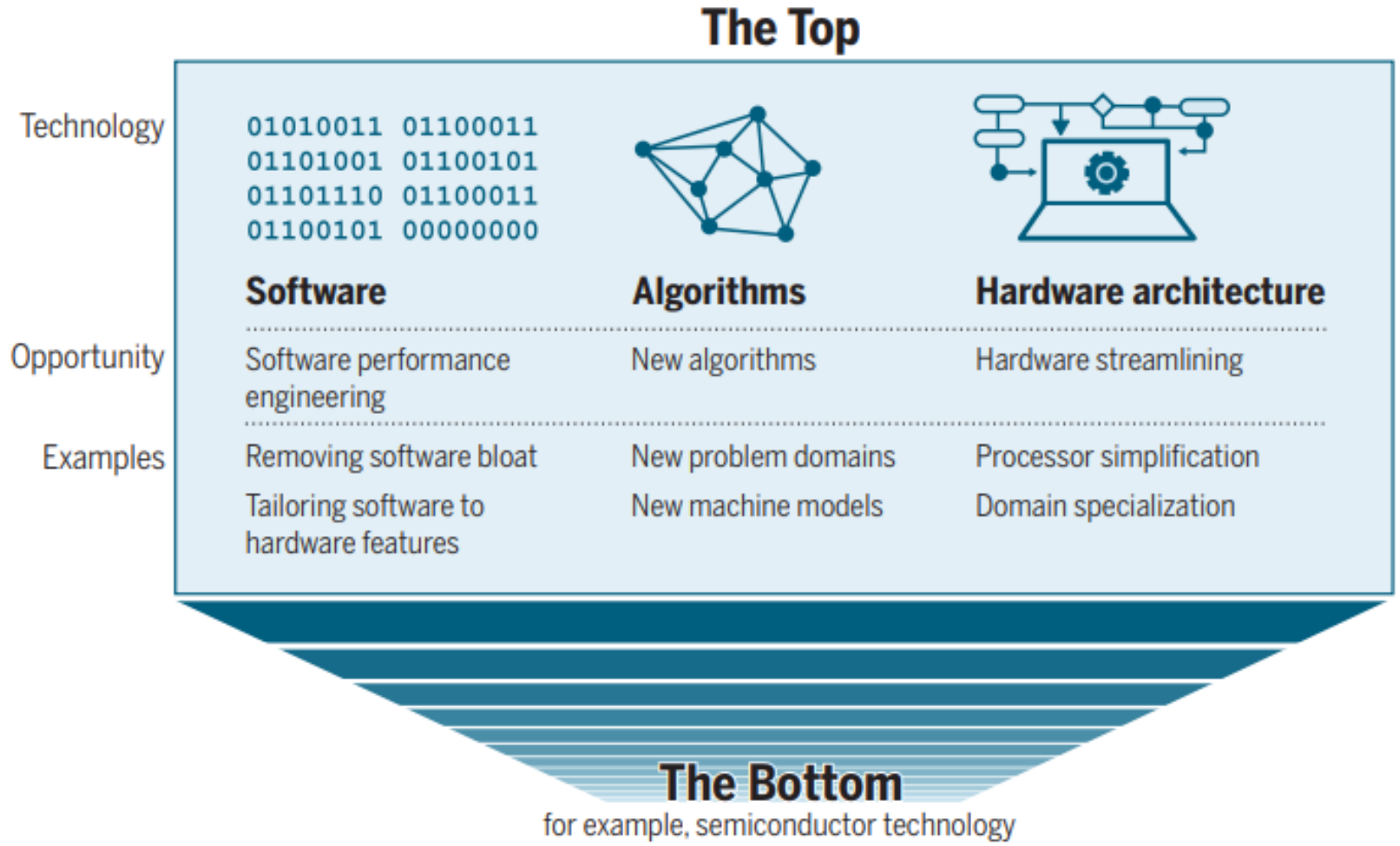
OUTLOOK: As miniaturization wanes, the silicon-fabrication improvements at the Bottom will no longer provide the predictable, broad-based gains in computer performance that society has enjoyed for more than 50 years. Software performance engineering, development of algorithms, and hardware streamlining 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 sporadic. Moreover, they will be subject to diminishing returns as specific computations become better explored. ■

ON OUR WEBSITE

Read the full article at https://dx.doi.org/10.1126/science.aam9744

The list of author affiliations is available in the full article online.
*Corresponding author. Email: neil_t@mit.edu
Cite this article as C. E. Leiserson et al., Science 368, eaam9744 (2020). DOI: 10.1126/science.aam9744

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

Main Classes of Computers

- Personal Mobile Devices
- Desktop Computing
- Servers
- Clusters / Warehouse Scale Computers

Parallelism and Parallel Computers

Parallelism at multiple levels is now the driving force of computer design across all four classes of computers, with energy and cost being the primary constraints. There are basically two kinds of parallelism in applications:

1. *Data-level parallelism (DLP)* arises because there are many data items that can be operated on at the same time.
2. *Task-level parallelism (TLP)* arises because tasks of work are created that can operate independently and largely in parallel.

Computer hardware in turn can exploit these two kinds of application parallelism in four major ways:

1. *Instruction-level parallelism* exploits data-level parallelism at modest levels with compiler help using ideas like pipelining and at medium levels using ideas like speculative execution.
2. *Vector architectures, graphic processor units (GPUs), and multimedia instruction sets* exploit data-level parallelism by applying a single instruction to a collection of data in parallel.
3. *Thread-level parallelism* exploits either data-level parallelism or task-level parallelism in a tightly coupled hardware model that allows for interaction between parallel threads.
4. *Request-level parallelism* exploits parallelism among largely decoupled tasks specified by the programmer or the operating system.

Flynn's Taxonomy

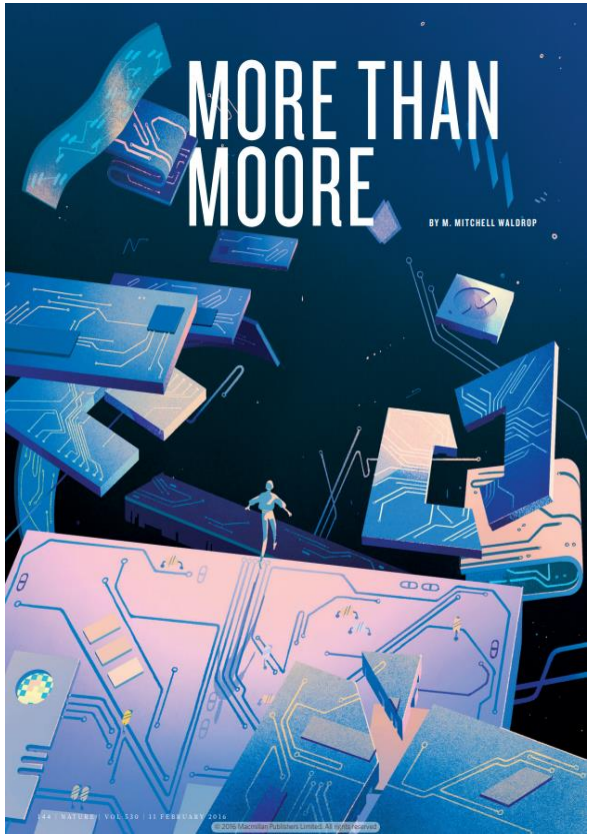
1. *Single instruction stream, single data stream* (SISD)—This category is the uniprocessor. The programmer thinks of it as the standard sequential computer, but it can exploit ILP. Chapter 3 covers SISD architectures that use ILP techniques such as superscalar and speculative execution.

2. *Single instruction stream, multiple data streams* (SIMD)—The same instruction is executed by multiple processors using different data streams. SIMD computers exploit *data-level parallelism* by applying the same operations to multiple items of data in parallel. Each processor has its own data memory (hence, the MD of SIMD), but there is a single instruction memory and control processor, which fetches and dispatches instructions. Chapter 4 covers DLP and three different architectures that exploit it: vector architectures, multimedia extensions to standard instruction sets, and GPUs.

3. *Multiple instruction streams, single data stream* (MISD)—No commercial multiprocessor of this type has been built to date, but it rounds out this simple classification.

4. *Multiple instruction streams, multiple data streams* (MIMD)—Each processor fetches its own instructions and operates on its own data, and it targets task-level parallelism. In general, MIMD is more flexible than SIMD and thus more generally applicable, but it is inherently more expensive than SIMD. For example, MIMD computers can also exploit data-level parallelism, although the overhead is likely to be higher than would be seen in an SIMD computer. This overhead means that grain size must be sufficiently large to exploit the parallelism efficiently. Chapter 5 covers tightly coupled MIMD architectures, which exploit *thread-level parallelism* because multiple cooperating threads operate in paral-

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.

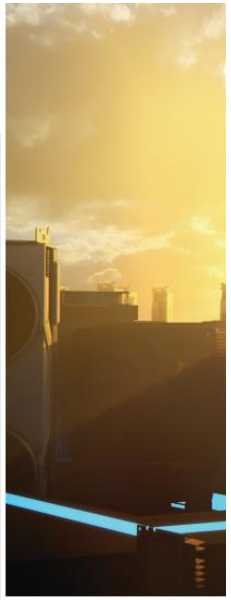
BY JOHN L. HENNESSY AND DAVID A. PATTERSON

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

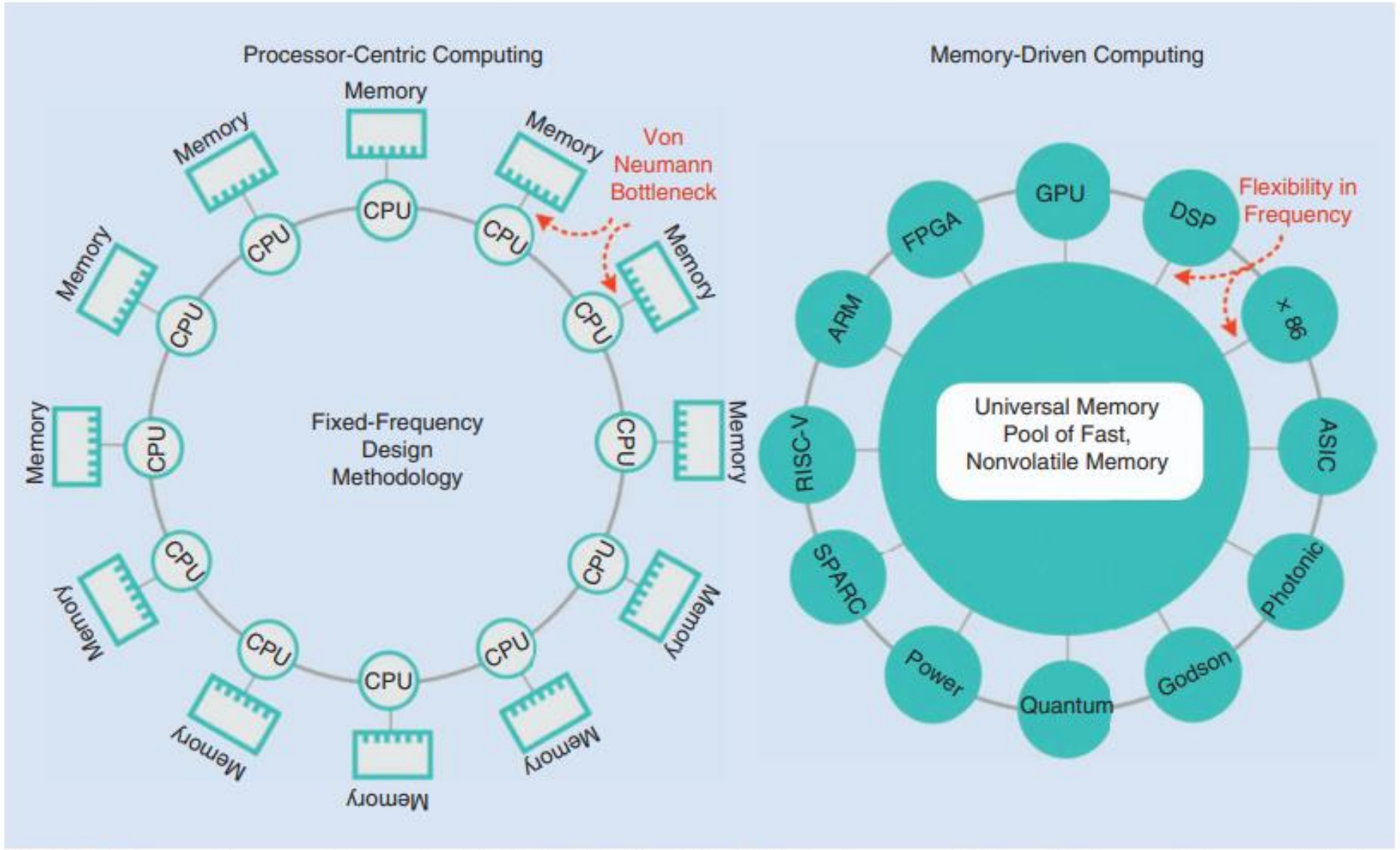


FIGURE 7: Memory-driven computing requires flexible clocking [59]. DSP: digital signal processor. ASIC: application-specific integrated circuit; RISC-V: reduced instruction set computing.

How does computer architecture impact performance?

1. **Processor Clock Speed**: The faster the processor, the quicker the system can execute instructions
2. **Number of Cores**: Modern processors often have multiple cores, which allow them to perform multiple tasks simultaneously. This can improve performance for multitasking and multi-threaded applications.
3. **Memory**: The amount and type of memory (RAM, DRAM) in a system can also affect performance by reducing the need to retrieve data from slower storage devices.
4. **Storage Devices**: Some type of storage devices like Solid-state drives (SSDs) are much faster than traditional hard disk drives (HDDs), leading to quicker boot times, faster file transfers, and quicker loading times for applications.
5. **Bus Design**: Bus connects the different parts of the computer. A faster bus allows data to be transferred more quickly between the processor, memory, and other components.
6. **Cache Size and Levels**: Processors often have a small amount of 'cache' memory, which is used to store frequently accessed data. A larger cache can improve performance by reducing the need to fetch data from main memory.
7. **Instruction Set Architecture (ISA)**: Some ISAs are more efficient than others, allowing the processor to perform more work in the same amount of time.
8. **Graphics Processing Unit (GPU) / Accelerators**: The performance of the GPU can significantly affect overall system performance. Other accelerators can have significant effect on performance.
9. **Cooling and Energy Requirements:** Overheating can cause a computer to slow down or even shut down to prevent damage. An inadequate power supply can lead to system instability and crashes, which can affect performance. A good power supply ensures all components receive the power they need to operate correctly.

Interesting 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. More Energy dissipation – too hot chips
   c. Domain Specific Architectures
   d. Enhanced security features inside microprocessors
   e. Open Instruction Sets and Extension of Instruction Sets through Customized Accelerators
   f. Agile Hardware Design for Microprocessors
   g. 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