Contemporary Engineering Sciences, Vol. 19, 2026, no. 1, 1 - 6
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ces.2026.93167

Post Moore's Law Computing

- A paradigm shift in the design and development of computing systems

Ping-Tsai Chung

School of Engineering, Computer Science & AI
Long Island University, New York, USA

Abstract

For several decades, Moore's Law served as a guiding principle for the advancement of computing technology. Moore's Law observed by Intel co-founder Gordon Moore in 1965, that the number of transistors on a microchip doubles approximately every two years, leading to a significant increase in computational power. However, as we approach the physical limitations of traditional semiconductor technology, experts anticipate that Moore's Law will no longer hold true. This has given rise to **the concept of post-Moore's Law computing**, which focuses on alternative approaches to sustain and enhance computing. In this paper, we explore some key aspects of post-Moore's Law computing to sustain and advance computing capabilities beyond the limits of traditional transistor scaling. By leveraging these innovations, researchers and engineers aim to unlock new possibilities and drive future advancements in computing power, efficiency, and functionality. Moore's Law only stops when innovation stops.

Keywords: Post-Moore's Law computing, innovation, central processing unit (CPU), graphics processing unit (GPUs), parallelism, neural processing unit (NPU), and Artificial Intelligence

I. INTRODUCTION

Moore's Law observed by Intel co-founder Gordon Moore in 1965 that the number of transistors in an integrated circuit (IC) doubles about every two years [1]. Moore's prediction has been used in the semiconductor industry to guide long-term planning and to set targets for research and development (R&D). Advancements in digital electronics, such as the reduction in quality-adjusted prices of microprocessors, the increase in memory capacity (RAM and flash), the improvement of sensors, and even the number and size of pixels in digital cameras, are strongly linked to Moore's law. These ongoing changes in digital electronics have been a driving force of technological and social change, productivity, and economic growth. However, as we approach the physical limitations of traditional semiconductor technology, experts anticipate that Moore's Law will no longer hold true. This has given rise to **the concept of post-Moore's Law computing**, which focuses on alternative approaches to sustain and enhance computing. In this paper, we explore some key aspects of post-Moore's Law computing to sustain and advance computing capabilities beyond the limits of traditional transistor scaling. By leveraging these innovations, researchers and engineers aim to unlock new possibilities and drive future advancements in computing power, efficiency, and functionality. Moore’s Law only stops when innovation stops.

II. PERFORMANCE EVALUATION

CPU (Central Processing Unit) is the "brain" of a computer, responsible for executing instructions and performing general-purpose calculations. A general-purpose Computer Architecture with CPU is designed for sequential processing; it handles computer instructions one after another. It is versatile and capable of handling a wide range of tasks.

The performance of a program depends on the algorithm, the programming language, the compiler, the architecture, and the actual hardware [2][3]. The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the Clock per instruction (CPI) by favoring slower or faster instructions. For example, if the algorithm uses more floating-point operations, it will tend to have a higher CPI. The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions.

The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler’s role can be very complex and affect the CPI in complex ways. The following performance equation, in terms of execution time (i.e., Seconds/Program) for running a program in (1), summarizes how these components affect the factors.

$$\text{CPU Execution Time} = \text{Instructions/Program} \times \text{Clock cycles/Instruction (CPI)} \times \text{Seconds/Clock cycle} \qquad (1)$$

The testing program depends on the programmer, algorithm involved in the testing program, the instruction set of architecture (ISA) of the computer, and the compiler that runs the testing program. The CPI depends on the system architecture and the microarchitecture, internal design and organization of a CPU processor that implements a given ISA such as circuit design, pipeline design, and technology. A strong microarchitecture is vital for processor performance and efficiency. The Instruction Set Architecture (ISA) affects all three aspects of CPU performance in (1), since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor. The critical system design issues for computer classes are listed below in Table 1.

Features	**Personal Mobile Device**	**Desktop**	**Server**	**Warehouse Scale Computers**
Critical System Design Issues	Cost, Energy, Media, Performance, Responsiveness	Price-Performance, Energy, Graphics Performance	Throughput, Availability Scalability, Energy	Price-Performance, Throughput, Energy, Proportionality

Table 1. Critical System Design Issues for computer classes

For developing a personal mobile device application, we are concerned about its Cost, Energy, Media, Performance, and Responsiveness design issues. For designing a Desktop Computer, we are concerned about Price-Performance, Energy, and Graphics Performance issues; however, for designing a server computer, we focus on throughput, availability, scalability, and energy. Furthermore, for designing a Warehouse Scale computer, we address Price-Performance, Throughput, Energy, and Proportionality that involves energy-proportional computing where a computer's power consumption scales linearly with its workload.

The weakness of CPU is not ideal for computationally intensive tasks that benefit from parallel processing. For four years, NVIDIA has waged a campaign to redefine the role of GPUs. Since the early 1990s, NVIDIA's GPUs have offloaded most of the graphics processing from CPUs in hundreds of millions of personal computers and videogame machines. NVIDIA continues to design its GPUs for superb graphics performance. Furthermore, since 2005, NVIDIA has developed GPUs for computing applications beyond graphics. **GPU computing** or it's often called **high-performance computing.** By harnessing the massively parallel-processing resources originally designed for 3D graphics, clever programmers can apply GPUs to a much broader range of computing applications. Some of those applications, such as video transcoding, still involve graphics to some degree. Whereas the CPU might spend an hour or more converting a recorded video uploading to YouTube or burning a DVD, a GPU can tear through the job in minutes. Originally, NVIDIA used the term **Compute Unified Device Architecture (CUDA)** to describe its GPU computing platform including architecture, run-time platform, and software-development tools. Now NVIDIA's next-generation CUDA architecture, code-named **Fermi**, adds powerful new features for general-purpose computing. Fermi processors will continue shouldering the graphics workloads in PCs and videogame consoles, but they are taking the largest step yet toward becoming equal-partner coprocessors with CPUs [6]. With high-performance computing becoming an increasingly essential part of modern life, efficiently delivering improvements in computing performance is the defining challenge for our industry. The rapid growth in the number of connected devices is creating orders-of-magnitude more data, requiring greater levels of computer to generate actionable insights for personal use and businesses. At the same time, supercomputers have become critical to enable research breakthroughs in fields including climate change, renewable energy development, infectious disease research, complex life science modeling, and much more [7].

Parallelism is the driving force of computer design [3][5], with energy and cost being the constraints Classes of parallelism in applications including Data-Level Parallelism (DLP): many data items operated at the same time; Task-Level Parallelism (TLP): tasks operated independently and in parallel. In Table 2, we summarize Classes of architectural parallelism.

Parallelism	**Parallelism in application**
Instruction-Level Parallelism (ILP):	Data-Level Parallelism (DLP)
Vector architectures/Graphic Processor Units (GPUs):	DLP
Thread-Level Parallelism:	DLP or TLP
Request-Level Parallelism	Task-Level Parallelism (TLP)

Table 2. Parallelism to Drive Computer Designs.

III. NO MOORE's LAW IN SOFTWARE

There are no Moore's laws in software. But the White House advisory report cited research, including a study of progress over a 15-year span on a benchmark production-planning task. Over that time, the speed of completing the calculations improved by a factor of 43 million. Of the total, a factor of roughly 1,000 was attributable to faster processor speeds, according to the research by Martin Groeschel, a German scientist and mathematician. Yet a factor of 43,000 was due to improvements in the efficiency of software algorithms.

Professor Edward Lazowska at the University of Washington pointed out that
First, the rapid pace of software progress is harder to measure in algorithms performing nonnumeric tasks. Second, the progress of recent years in AI Applications like language understanding, speech recognition and computer vision as evidence that the story of the algorithm's ascent holds true well beyond more easily quantified benchmark tests. Third, there is a lot to the notion of the yin and yang of computing, software and hardware inextricably linked [4].

Neural Processing Units (NPUs) refer to on-chip accelerators dedicated to neural network computations, often integrated into system-on-chip (SoC) designs for smartphones, tablets, and edge devices. For instance, Apple's Neural Engine (introduced in iPhone chips since 2017) and Qualcomm's Hexagon DSP / AI Engine are NPUs designed to run AI inference tasks on-device (like face recognition, speech recognition, and image enhancement) with high efficiency.

Therefore, an AI PC is a computer with NPU designed to run AI tasks locally on the device instead of relying on the cloud. This architecture results in faster performance, enhanced privacy, and better power efficiency. Regular PCs may not support these features or may rely heavily on cloud-based processing.

Furthermore, AI PCs power features like real-time transcription, intelligent noise cancellation, background blur, predictive typing, and personalized performance tuning which enhance productivity and creativity. The weakness of NPU is that
less versatile than CPUs and GPUs, and not as good at general-purpose computing.

IV. CONCLUSIONS

The concept of post-Moore's Law computing, which focuses on alternative approaches to sustain and enhance computing, is still evolving; At CES 2026, industry leaders continue projected from approximately 100 zettaflops in early 2026 to more than 10 yottaflops with five years [8]. Note that 1 ExaFlop = a billion calculations every second. 1 ZettaFlop = 1,000 exaflops, 1 yottaflop = 1,000,000 exaflops.That is, reaching yottascale AI compute would require the equivalent of millions of today's frontier exaflop-class supercomputers operating together – a scale that would have been unimaginable just a few years ago. In this paper, we discussed some key aspects of post-Moore's Law computing to sustain and advance computing capabilities beyond the limits of traditional transistor scaling. By leveraging these innovations, researchers and engineers aim to unlock new possibilities and drive future advancements in computing power, efficiency, and functionality. Moore's Law only stops when innovation stops.

References

[1] G. E. Moore, Cramming more components onto integrated circuits, *Electronics*, **38** (1965). https://doi.org/10.1109/n-ssc.2006.4785860

[2] D. A. Patterson, J. L. Hennessy, *Computer Organization and Design - The Hardware / Software Interface*, Morgan Kaufmann, 2020.

[3] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 6th Edition, 2017.

[4] S. Lohr, Software Progress Beats Moore's Law, NY Times, (2011).

[5] M. J. Fynn Very high-speed computing systems, *Proceedings of the IEEE*, **54** (12) (1966), 1901-1909. https://doi.org/10.1109/proc.1966.5273

[6] T. R. Halfhill, NVIDIA's Next-Generation CUDA Compute and Graphics Architecture, Code-Named Fermi, Adds Muscle for Parallel Processing, White Paper: Looking Beyond Graphics, NVIDIA, (2009).

[7] L. Su, S. Naffziger, Innovation for the Next Decade of Compute Efficiency, the IEEE International Solid-State Circuits Conference (ISSCC), 19-23, (2023). https://doi.org/10.1109/isscc42615.2023.10067810

[8] L. Su, From CES 2026 to Yottaflops: Why the AMD Keynote Highlights a Turning Point for AI compute, AMD (2026).

Received: January 11, 2026; Published: January 27, 2026