Check for updates

International Journal of
HIGH PERFORMANCE
COMPUTING APPLICATIONS

Survey Paper

The International Journal of High
Performance Computing Applications
2025, Vol. 0(0) 1–60
© The Author(s) 2025

Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/10943420251348799
journals.sagepub.com/home/hpc

Technology trends in computing hardware and their impacts on high-performance scientific computing Part I: General-purpose processors and hardware accelerators

Bagus Hanindhito[1,*], Arash Fathi[2,*], Dimitrios Gourounas[1], Dimitar Trenev[2], Andreas Gerstlauer[1,3] and Lizy K John[1,3]

Abstract

Computing technology has evolved significantly during the past five decades. As semiconductor scaling is reaching physical and technological limits, it is driving many transformative changes in computing hardware. This has led to computing systems that rely heavily on multi-core processors and GPUs, and resulted in the development of specialized hardware for applications in machine learning and scientific computing. While modern hardware provides significant computing power, and therefore opportunities, it challenges many established algorithms and workflows in scientific computing: these algorithms may not be able to fully leverage modern hardware. Often times, effective use of modern hardware entails revised algorithms, and even rewriting a considerable portion of an existing code. Understanding technology trends in computing hardware is necessary for designing next-generation algorithms for scientific computing. This paper reviews these trends, along with their drivers, in a language that is accessible to computational and data scientists, and applied mathematicians. In this paper (Part I), we review technology evolution in general-purpose microprocessors and hardware accelerators, along with background material. In Part II (Hanindhito et al., 2026), we consider memory systems, inter-device communication, heterogeneous computing and system integration, energy consumption, and how these trends impact scientific computing.

Keywords

Scientific computing, computing hardware, Moore's law, Dennard scaling, GPU, specialized hardware

Introduction

Motivation

Computing hardware has experienced a lot of changes, especially during the last two decades: multi-core microprocessors are common; GPU computing is gaining traction among a broader group; multi-node computing is more routine in industry and academia due to interest in solving larger and more complex problems; there are several examples of specialized hardware that are being used for scientific computing; and there is renewed interest in more exotic forms of computing, such as quantum and neuromorphic computing.

Some of these changes are stimulated by the end of Moore's law and semiconductor technology scaling: it is harder to double the processing power of modern chips every 2 years by doubling the number of transistors on a chip, at the same cost; moreover, power consumption is limiting the compute capacity of modern chips, as they typically have a higher power density, and cooling them is becoming increasingly more challenging.

A forward-looking computational scientist, applied mathematician, algorithm developer, or an industry that relies heavily on scientific computing must carefully

[1]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA
[2]ExxonMobil Technology and Engineering Company, Research, Spring, TX, USA
[3]Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA

*Authors contributed equally.

Corresponding author:
Arash Fathi, ExxonMobil Technology and Engineering Company, Research, 22777 Springwoods Village Parkway, Spring, TX 77389.
Email: arash.fathi@exxonmobil.com

examine the impact of changes in computing hardware on their algorithms and workflows: should different algorithms be designed to better harness emerging hardware? Is it feasible to design specialized hardware for some compute-intensive workflows? What are possible consequences of inaction, i.e., using current algorithms and workflows on emerging hardware? What problems will become tractable a decade from now due to projected trends in computing hardware? Answering these questions, and quantifying their impact, are quite challenging, and may need a multi-year, multi-disciplinary effort.

Generally speaking, computational and data scientists, and applied mathematicians, are not trained in computer architecture and technology trends in computing hardware at a level of detail that may guide strategic decision-making on future algorithms and workflows. Literature that provides more detail is often too technical, hard to read, not self-contained, or may not clearly outline the interaction between different parts and technologies.

We attempt to bridge this gap: through a collaborative effort between hardware engineers, computational scientists, and applied mathematicians, we provide a holistic view of technology trends in computing hardware,[1] and highlight interactions between different components. We venture into how these trends may impact different algorithms that are commonly used in high-performance scientific computing. This effort is distinct from similar works, by providing a deep level of technical detail, while attempting to keep it accessible to a general computational scientist. By providing numerous examples and illustrations to clarify key trends, we help readers form their own judgement. To improve readability, helping readers see the big picture without getting lost in details, and keeping the paper self-contained, we provide certain details and examples in footnotes.

We believe this paper provides a clear and detailed perspective about technology trends in computing hardware to our targeted audience, along with their impacts on algorithms. These trends, along with domain knowledge, provide insights to computational scientists, and could position them to make informed decisions about designing algorithms and workflows that are expected to perform well on modern hardware, especially in industries that rely heavily on high-performance scientific computing, such as oil and gas, aerospace, defense, automotive, pharmaceutical, and finance.

Outline and summary

We begin each section by reviewing historical and current trends, along with future projections, based on publicly-available industry roadmaps. We discuss how different technologies interact, and how they impact the overall performance. We end each section with takeaways, possibilities, and key challenges, along with our own opinions that are supported by the trends. In what follows, we provide a high-level summary and refer to different parts of the text for details. Acronyms we used throughout the text are listed in Appendix 1.

Background material on key concepts in computing hardware, along with general trends are covered in *General technology trends and concepts in computing hardware*. They are repeatedly referred to throughout the rest of the paper, and therefore, we suggest readers review it before reading the rest of the text. Specifically, we highlight key characteristics of transistors, which are the fundamental building blocks of computer chips. These include the clock frequency, which impacts the switching speed and therefore runtime performance, as well as power consumption of transistors. The transistors have consistently become smaller over the years. This has allowed placing more of them in the same area, which typically results in more powerful chips. Moore's law described the rate of transistor miniaturization at the same cost, and thus, it played a key role in setting expectations and guiding industry roadmaps. While some believe Moore's law is losing steam, others believe it is no longer alive. Semiconductor scaling is facing physical and technological limits. Dennard scaling, which described how computer chips could keep their power consumption under control as transistors become smaller does not hold anymore. Consequently, modern computers often need more energy to operate and therefore, energy efficiency has become a central issue in modern chip design.

As transistors become smaller, more of them are placed on the same silicon die area. This increases the possibility of defected chips as they become larger, and therefore results in lower yield. However, advances in packaging technology have resulted in larger and more powerful chips without impacting the yield, as well as keeping R&D and manufacturing costs under control. The small size of transistors also increases their vulnerability, which makes maintaining reliability for advanced chips increasingly more challenging. Not only modern chips have smaller transistors, they also have thinner wires for internal connectivity. Wire scaling is challenging as it increases resistance and impacts signal integrity, and thus, has motivated the development of optical on-chip interconnects. In parallel, as semiconductor technology scaling is reaching its limits, chip designers need to find new ways to improve performance, resulting in the widespread adoption of diversification in architecture and implementation of modern chips.

Computer chips often need to communicate with off-chip components (e.g., other processors or memory) to perform computations. The communication is enabled through pins. Over the years, the number of transistors on a computer chip has grown much faster compared to the number of its pins. This has led to communication bottlenecks, and thus contributed to substantial alleviation efforts through both hardware-centric and algorithmic approaches. The hardware-centric techniques often attempt to increase the communication bandwidth though using new technologies,

or using the communication signals more efficiently, e.g., through signal modulation. The algorithmic approaches often attempt to reduce the communication at the expense of increasing local computations.

In *General-purpose microprocessors*, we review their evolution during the past several decades. As transistor scaling enabled placing more transistors on a chip, a significant portion of these transistors supported instruction-level parallelism, leading to more powerful, single-core chips. This strategy showed diminishing returns on performance in the early 2000s, and led to the development of multi-core processors. Since then, adding more cores to a chip generally has a greater impact on the overall performance. Indeed, sophisticated units within compute cores do not benefit many applications that enjoy less complexity (e.g., dense linear algebra). The rise of these applications led to the development of many-core processors. Future general-purpose microprocessors are becoming increasingly diverse to meet specific performance needs of various applications. This includes processors that enable a faster runtime, or processors that may not be as fast, but are more energy-efficient, as well as processors that have access to a significant amount of on-package memory.

In *Hardware accelerators*, we examine how these devices provide performance improvements for select applications. These applications often enjoy a lot of regularity and structure, afforded by linear algebra, and enjoy a high computation-to-data-movement ratio. These conditions are typically observed in machine learning, and, sometimes, in scientific computing. Accordingly, the application's structure can be exploited to improve how a chip utilizes silicon area and energy: more silicon can be dedicated to perform the actual computations, as opposed to units that manage on-chip flow control and data movement. In layman's terms, well-behaved applications require less management, and therefore, the real estate that performs management can shrink to accommodate more workers. This specialization results in chips, such as GPUs that have significantly more processing power, and consume less energy, compared to general-purpose microprocessors. Availability of well-developed and well-documented software stacks[2] has been a key enabler for the adoption of GPUs by a large group of computational and data scientists. Porting a CPU code into GPUs is typically a tedious task. Often times, the underlying algorithms need to be revised, and a considerable portion of the code needs to be rewritten to maximize performance. Not all workloads and algorithms are suited for GPUs. General-purpose microprocessors are still needed for many applications. GPUs are still programmable hardware, and while efficient for some computations, they do not deliver the best performance for many applications. Given that technology scaling is reaching its limits, more aggressive hardware specialization remains among the very few options to further improve performance for the foreseeable future. Typically, a large market for the specialized hardware is needed to justify research and development costs, as well as the software support. A specialized hardware may be implemented on different fabrics (e.g., FPGA, CGRA, or ASIC), depending on performance requirements and the maturity of the targeted applications. We provide several examples of publicly-known specialized hardware that are used in machine learning and scientific computing.

General technology trends and concepts in computing hardware

Concepts outlined in this part are fundamental to understanding the underlying reasons behind computing technology trends, and are repeatedly referred to in the paper. They are briefly explained to provide context, improve readability, and to keep the paper self-contained.

Clock frequency

Clock frequency has been used as a proxy for chip performance (Agarwal et al., 2000; Henning, 2000). The switching activity of the transistors is governed by a periodic pulse signal, called clock (Xiu, 2019). The clock synchronizes the switching time of millions of transistors inside the chip (Friedman, 2001; Messerschmitt, 1990). This synchronization is vital for the chip to perform its operations correctly, including fetching data from peripheral devices (e.g., main memory (Cristal et al., 2005)), executing a stream of instructions (e.g., program code (Choi et al., 2004; Crawford, 1990)), and storing back the results. Therefore, the clock frequency, measured in Hz, determines how fast the transistors switch states between on and off, roughly translating into how many operations a chip can do per second (Messerschmitt, 1990). Several factors impact the highest possible clock frequency that a chip can run at, while maintaining the integrity of the data that flows throughout the circuitry. These are: a) intrinsic properties of the transistors, which depend on the process node[3] and manufacturing technologies (Geppert, 2002; Shahidi, 2007); b) operating voltage of the transistors (Liu and Svensson, 1993; Meijer et al., 2004); and c) the microarchitecture implementation of the chip itself (Boggs et al., 2004; Marculescu and Talpes, 2005).

Clock frequency may not always be a representative metric for performance. For instance, consider two chips with identical integer units, A and B: chip A runs at a clock frequency that is twice as fast as B, and B is equipped with a more advanced floating-point unit that can perform floating-point operations in 75% less clock cycles than A (e.g., 2 clock cycles for B vs 8 clock cycles for A). In a workload where 99% of the instructions are floating-point operations, chip B will perform nearly twice as fast as A, despite its lower clock frequency. However, in a workload where there

is minimal floating-point instructions, chip A will perform nearly twice as fast as B. Therefore, the clock frequency is not an accurate metric for comparing the performance of chips with different manufacturing technologies, operating conditions, and microarchitectures, which run diverse workloads.

Transistor power consumption and heat dissipation

Reducing the power consumption of modern transistors is arguably the biggest challenge in modern chip design (*Dennard scaling*). Electrical power consumed by transistors can be grouped into two major categories: dynamic power and static power.

CMOS[4] (transistors) consume dynamic power when switching states (Kaxiras and Martonosi, 2008; Kocanda and Kos, 2015). The majority of dynamic power consumption is due to charging and discharging of the parasitic capacitance[5] while a small portion is due to the short-circuit current.[6] The dynamic power (P_d) can be estimated through $P_d = C.f.V^2$, where C is the (parasitic) capacitance that depends on the process node and manufacturing technologies, f is the clock frequency, and V is the operating voltage of the transistors. The operating voltage at a transistor gate should be greater than a threshold voltage (V_t) to turn it on (Weste and Harris, 2010). While increasing the clock frequency directly increases the dynamic power, a higher operating voltage is also required for the transistors to sustain faster switching activity and maintaining the correctness of the operations (Le Sueur and Heiser, 2010). Therefore, increasing clock frequency increases the dynamic power significantly.

Static power is a consequence of current leakage (Butts and Sohi, 2000; Elgharbawy and Bayoumi, 2005). Ideally, when a CMOS transistor holds its state (either on, or off), no current should flow from the voltage source applied to the transistor. In reality, a small amount of current flows due to leakage. The leakage power increases as transistor size becomes smaller.

Power consumed by the transistors is converted into heat (Brooks et al., 2007; Gonzalez and Horowitz, 1996), which then must be pulled away to prevent damage. Cooling systems, including heat spreaders, heat sinks, and active cooling (e.g., fans) work together to remove the heat from an area in the order of squared centimeters. High power density, and small surface area for thermal contact, make heat removal challenging (Mahajan et al., 2006a; Wei, 2008). For instance, the Intel Pentium 4 Netburst P68 microarchitecture (Boggs et al., 2004) was expected to reach 10 GHz by 2005 (Ogasawara, 2002). However, even at 3.8 GHz, its power density reached 105 W/cm^2, which is the same power density induced by the core of a nuclear reactor (Gelsinger, 2001).

Supplying power to chips is bounded (Aygün et al., 2005; Radhakrishnan et al., 2021) due to physical limits on the amount of current that can be carried by wires and pins, from a power supply and power delivery system[7] to a chip (Lidow and Sheridan, 2003; Yazdani et al., 1997). The use of aggressive power management features, such as dynamic voltage and frequency scaling (DVFS) (Le Sueur and Heiser, 2010; Papadimitriou et al., 2019), play important roles in keeping the chip power consumption under control. The term typical power is often used to indicate the sustained power that a modern microprocessor consumes under a long heavy load,[8] and is used to design the cooling system needed for heat removal (Ganapathy and Warner, 2008; Guermouche and Orgerie, 2022).

Transistor size and Moore's law

Transistors are the building blocks of computer chips. They operate like a switch representing a binary value of 0 and 1 (i.e., off and on, respectively). Generally speaking, placing more transistors on a chip improves its processing power.

Gordon Moore, the co-founder of Intel, has made several predictions about transistor miniaturization trends due to technology improvements in Moore (1965) and in a 1975 paper reprinted in Moore (2006). His best-known forecast (1975), which is widely known as Moore's law, predicted shrinking of transistor sizes that corresponds to a doubling of transistor density on chips every 2 years. When Moore made his prediction, the design and manufacturing cost of a chip was proportional to its area. Therefore, Moore's law was intrinsically an economic forecast, stating computer chips becoming twice as powerful every 2 years, with the same price tag.

Consequences of Moore's law were significant: it guided technology roadmaps and became a self-fulfilling prophecy; furthermore, many applications enjoyed performance improvements proportional to transistor scaling rates while making minimal changes to their algorithms.

While chip manufacturers still go to great lengths to shrink transistor size according to Moore's law, the economic aspects of Moore's forecast are no longer sustained. For instance, Jensen Huang, CEO of NVIDIA, believes Moore's law is dead (Jain and Murugesan, 2021). Transistor scaling has been challenged by physical[9] and technological limits. We highlight the technological challenges in *Transistor power consumption and heat dissipation*, *Dennard scaling* (the need for more power), *Transistor count and yield, and manufacturability* (lower yield due to higher transistor density on a chip), and *Cost of designing and manufacturing semiconductors* (skyrocketing costs of designing and manufacturing of modern chips).

The end of Moore's law has far-reaching impacts. Since performance can no longer be improved through technology scaling, new strategies are being pursued. These are highlighted in *Advanced packaging technologies* (to improve yield and reduce cost), *Hardware accelerators* (increasing hardware diversity to maximize performance of various

application groups), and *Part II: Near-memory processing (NMP) and processing-in-memory (PIM)* (Hanindhito et al., 2026) (moving away from von Neumann architecture).

Dennard scaling

The end of Dennard scaling has made power consumption the central issue in modern chip design. As a result, modern chips need more power to run, and their cooling has become more difficult. It has created unprecedented challenges in supplying power to supercomputing and data centers (*Part II: Energy consumption of large computing centers and its implications* (Hanindhito et al., 2026)).

Dennard et al. (1974) published MOSFET scaling rules, which are widely known as Dennard scaling. Dennard described scaling relationships between transistor density, switching speed, and power dissipation due to MOSFET miniaturization. Dennard scaling states that by scaling down the transistor size by a factor of κ, the voltage (V), electrical current (I), and capacitance (C) will be scaled by a factor of $\frac{1}{\kappa}$. The most important consequence of Dennard scaling was that power density per square area of a chip remains constant due to transistor miniaturization. Specifically, transistor density increases by a factor of κ^2, and power requirement per transistor decreases by a factor of $\frac{1}{\kappa^2}$, due to $P = V \times I$, leading to no change in power density. Other consequences of Dennard scaling include: a) parasitic capacitance, which is related to the area, becomes smaller (Kim et al., 2003); b) circuit performance is improved[10] as delay is scaled down by a factor of $\frac{1}{\kappa}$, along with transistors being able to operate at higher clock frequencies[11]; and c) voltage to drive transistors (V) and threshold voltage (V_t) become smaller (Geppert, 2002; Rieger, 2019).

Dennard scaling did not take into account the existence of a physical lower limit for the operating voltage (V), imposed by the threshold voltage (V_t) (Taur, 1999a), as well as implications of scaling down the threshold voltage (V_t) (Stillmaker and Baas, 2017; Taylor, 2013). It assumed the operating voltage (V) and the threshold voltage (V_t) could continue to scale down as transistors become smaller. However, decreasing the threshold voltage (V_t) in smaller transistors increases the current leakage[12] (Ahmed and Schuegraf, 2011; Kim et al., 2003), which leads to increased static power consumption (Kao et al., 2002). This makes both static and dynamic power equally important[13] when transistor size becomes very small (Sylvester and Kaul, 2001). At nanometer-scales, power density per square area can no longer be held constant. Power density increases as the transistors become smaller, causing the "power wall" (Wang and Skadron, 2013). This marks the end of Dennard scaling, which occurred in the mid-2000s (DeBenedictis, 2017; Taylor, 2013).

Transistor count and yield, and manufacturability

Figure 1 shows historical data on the advancement of semiconductor process nodes and a projection for the next few years. When transistor size becomes a few nanometers, it approaches physical and technological limits (Naveh and Likharev, 2000; Taur, 1999b). This makes further shrinking of the size challenging, as indicated by the slowdown in the advancement of the process node.[14] Due to the slowdown in transistor scaling, larger silicon dies became popular, enabling higher transistor counts. Larger dies increase the possibility of defects, thus lowering the manufacturing yield and increasing manufacturing costs (Mack, 2015; Sun et al., 2020).

However, the maximum die size that can be manufactured for future process nodes is decreasing. The reticle limit, currently at 26 mm × 33 mm (858 mm^2) (Lai, 2021; Suzuki, 2020), acts as the upper limit on how large a silicon die can be manufactured. The reticle limit is expected to be halved at 26 mm × 16.5 mm (429 mm^2) due to the amorphous lens array used in the upcoming process node. Advances in packaging technologies allow the compartmentalization of a chip, by using multiple smaller dies called chiplets. This improves the yield and reduces costs, while conforming to the reticle limit of future process nodes.

Advanced packaging technologies

Advances in silicon packaging[15] technologies (Figure 1) (Su et al., 2017) will play an important role in increasing the number of transistors on a package in the near future. We highlight two key technologies: Multi-Chip Modules (MCMs) (Naffziger et al., 2021) and chiplets.

An MCM uses multiple (often similar) silicon dies to form a large chip (Arunkumar et al., 2017; Burd et al., 2019; Su et al., 2017). While MCM has been around since the 1980s, its popularity has increased in recent years to overcome the low-yield challenge and reticle limit

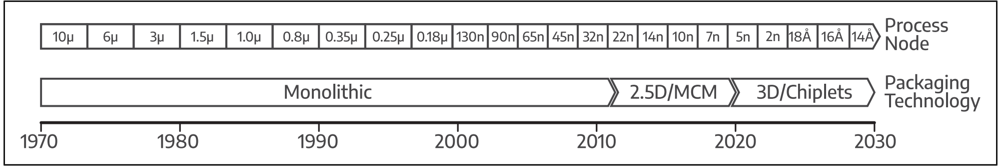

Figure 1. Semiconductor process node and packaging technology trends since the 1970s, with projections up to 2030.

(*Transistor count and yield, and manufacturability*). The silicon dies communicate through wires on the packaging substrate.[16] Traditionally, the dies are connected to the wires on the package substrate using wire-bond or flip-chip technology, which is referred to as 2D packaging. However, the wires in the package substrate are orders of magnitude larger than those in the silicon dies. The thicker wires and low wire density lead to routing congestion and therefore limit the attainable die-to-die bandwidth.

To improve connectivity between the silicon dies, an interposer layer is placed between the dies and the packaging substrate. In addition to bridging the connection between the silicon dies and the packaging substrate through vias, the interposer also acts as a conduit for connections between the silicon dies. Interposers are often manufactured using silicon,[17] hence the name silicon interposer. Silicon interposers provide significantly higher wire density, allowing implementation of high-bandwidth connection between dies. This is referred to as 2.5D packaging technology (Lenihan et al., 2013; Zhang et al., 2013), and the silicon dies are placed side-by-side. Further advancement of this technology is referred to as 3D stacking, which allows multiple dies to be stacked on top of each other (Agarwal et al., 2022; Su et al., 2017) and connected through vias.[18] While the 3D stacking technologies are promising, they face many challenges, such as the thermal issues[19] (Gomes et al., 2020; Su et al., 2017).

Specialization of silicon dies is becoming more common. Accordingly, a silicon die may perform only a specific function and thus manufactured using the best process node suited for that function. This type of specialization and compartmentalization is called chiplet (Loh et al., 2021; Naffziger et al., 2020). Just like in a puzzle, multiple chiplets with various functions[20] are used to build a modern System-on-Chip (SoC) MCM, which has continuously grown in popularity over the past decades. The use of chiplets is the recipe to keep Moore's law (perhaps nominally) alive, and is expected to result in chips with trillion transistors by 2030 (Gelsinger, 2022; Kelleher, 2022; Moore and Schneider, 2022). Note that communication interfaces between chiplets consume additional silicon area and power. Finally, the development for future scalable interconnect technologies are critical to allow for lower latency and high bandwidth communication between chiplets (Chirkov and Wentzlaff, 2023).

Reliability and availability of computing systems

Transistor size is getting smaller due to advanced manufacturing, leading to increased transistor density on advanced chips, which enables chips to carry increasingly more complex functions. The smaller size of transistors and rise of complexity make computer systems more susceptible to reliability issues. In this sense, reliability may be defined as a measure of success, where a computing system's behavior conforms to its specifications over a given operating period (Shooman, 2002). Failure happens when the behavior of the system deviates from its specifications. The relative proportion of time the system meets its specifications is called availability, which depends on the duration of failures as well as time needed to fix them.

An error is defined as an incorrect state of information stored in the system, whereas a fault is the cause of the error. The fault sources include component failure, equipment damage, interference (cross-talk) between wires, power disturbance, induction due to lightning, electromagnetic fields, electrical noise, and radiation (Randell et al., 1978). Radiation is a common cause of fault, especially for chips that are manufactured through smaller process nodes (Schrimpf et al., 2008), as they become more sensitive to it. Cosmic rays and alpha particles are common sources of radiation. They can cause soft errors (e.g., bit flips) in computing systems, particularly in logic and memory. A soft error is a non-permanent and non-recurring error, which corrupts information while the device itself may still function properly (Karnik and Hazucha, 2004). On the other hand, a hard error is often permanent (e.g., due to hardware failure) and may be repairable (Wang and Agrawal, 2008).

There are two approaches for achieving reliability in computing systems: fault-intolerance and fault-tolerance (Avižienis, 1975). Fault intolerance seeks to eliminate sources of unreliability. Since elimination of all possible sources of fault is not possible, fault intolerance reduces the probability of fault occurrence to an acceptable low level, and devises maintenance procedures should a fault occur. However, maintenance will impact system availability significantly, which may not be preferable for some critical computing systems. On the other hand, fault tolerance tolerates sources of unreliability, and seeks to counteract the consequences by using protective redundancy. Accordingly, systems that adopt fault tolerance can continue to operate despite the existence of errors, either at full or reduced capacity and capability, until the fault is addressed. For instance, aircraft computer systems use the Multiple-instruction Single-Data (MISD) approach, where multiple computer systems operate on the same data simultaneously. Next, the outputs are compared through a majority-voting mechanism. We remark that achieving reliability raises the cost of increasingly more expensive modern computing systems.

Wiring, connectivity, and signal integrity

In this part, we discuss the medium through which data is transmitted, which impacts the achievable bandwidth (Bogatin, 2022; Cho et al., 2007; Saraswat et al., 2008) as it affects signal integrity. The widely-used medium to carry electrical signals is metal-based wires, such as aluminum and copper. Carbon Nanotubes (CNT) are also briefly discussed as a potential replacement for metal-based wires.

Optical interconnects, which have gained more traction in recent years and serve as alternatives to electrical interconnects are explored in *Optical interconnects*.

A widely-used metric to measure the performance of wires for carrying electrical signals is propagation delay. There are several approaches for modeling the propagation delay, based on physical characteristics of the wires, and their interaction with other materials[21] (Seckin and Yang, 2008; Zhou et al., 1988). A simple but common approach is the RC model,[22] which relies on resistance (R) and capacitance (C) of a wire to estimate the propagation delay (Qian et al., 1994; Sakurai, 1993). The RC model is also referred to as RC delay (Ciofi et al., 2016; Sylvester and Keutzer, 1998). The resistivity of a wire relies on its material property and geometry (Ciofi et al., 2016; Savage, 2002). Capacitance of a wire is influenced by interactions between adjacent wires, and dielectric materials[23] (Duan et al., 2001; Ruehli and Brennan, 1975; Zhao et al., 2009).

As the number of transistors on a chip grows, the number of wires that provide connectivity between them has to increase as well (Edelstein et al., 1995). Most of the chip area has already been covered by wires, which limits space for laying out new wires. Therefore, more metal layers are being used (Gelatos et al., 1994; Koyanagi et al., 1998) to construct local, intermediate, and global on-chip interconnects.[24] Increasing the number of metal layers complicates the manufacturing processes, and has negative impacts on the capacitance and cross-talk of the wires (Duan et al., 2001; Sim et al., 2003). In principle, making the wires smaller increases wire density. However, unlike transistors, metal interconnects do not benefit from further scaling (JM Veendrick, 2017; Koo et al., 2007). Scaling negatively impacts metal interconnects: smaller wires have higher resistance, and carry less electrical current (Edelstein et al., 1995; Srivastava and Banerjee, 2004). Increased wire density also exacerbates cross-talk and capacitance effects between wires (Naeemi et al., 2006), affecting signal integrity.

Early process nodes used aluminum (Al) for on-chip wires. However, aluminum could not meet interconnect performance requirements for process nodes beyond 180 nm (Staff, 2019; Sylvester and Keutzer, 1998). The move from aluminum (Al) to copper (Cu) in the late 1990s for metal wires (Andricacos, 1999; Theis, 2000), and the use of low-κ dielectric materials in the 2000s (Beyne, 2003), provided a one-time opportunity to improve propagation delay.[25] Despite their more complicated manufacturing process (Andricacos et al., 1998; Gelatos et al., 1994, 1996), copper wires have significantly smaller resistance,[26] and higher durability, compared to aluminum wires. For a while, this allowed copper wires to be made smaller, keeping pace with transistor scaling. Eventually, however, technology scaling became more challenging for copper, due to complications in electrical conductivity, reliability,[27] latency, and power dissipation (Kapur et al., 2002; Tőkei et al., 2016). Copper saw its limitation[28] at 40 nm (Kaushik et al., 2007). IBM estimated new metal materials[29] are needed for constructing on-chip wires beyond 15 nm (Bonilla et al., 2020; Huang et al., 2020; Staff, 2019).

Replacing copper with carbon nanotubes (CNT) for on-chip wires has been proposed since the early 2000s (Joshi and Soni, 2016; Xu et al., 2022). CNT has a higher reliability than copper, due to its mechanical and thermal stability. It reduces delay for intermediate and global on-chip interconnects by about 30% (Naeemi et al., 2006; Nieuwoudt and Massoud, 2008). However, integrating CNT onto a chip is challenging, due to immature manufacturing techniques, imperfect metal to CNT contacts, and high growth temperature[30] of CNT, which results in higher probability of defects and low achievable wire density (Kaushik et al., 2014; Pasricha et al., 2010; Xu et al., 2022).

Optical interconnects

Some researchers are considering the possibility of moving away from electrical signals to optical signals and optical interconnects for on-chip communication, thus limiting the amount of on-chip metal wires. Optical interconnects operate at the speed of light. Therefore, they provide significantly higher bandwidth for data transmission with lower latency compared to electrical signals. Optical interconnects have also proven to be reliable, performant, and efficient, for long-distance inter-node communication (*Part II: Inter-node communication* (Hanindhito et al., 2026)). Indeed, research on optical interconnects and silicon photonics[31] dates back to the 1980s (Soref and Bennett, 1987; Soref and Larenzo, 1986). Earlier studies showed using optical signals for on-chip interconnects may not be effective due to silicon area and power requirements needed by electrical-optical conversion devices (Kobrinsky et al., 2004; Sato et al., 2015). Nevertheless, optical interconnects have several merits, compared to copper wires, which make them attractive for process nodes beyond 10 nm.

Co-existence of electrical interconnects and optical interconnects on future chips is likely: electrical interconnects may be used to realize local on-chip connections, whereas optical interconnects can be used to realize intermediate and global on-chip connections (Chen et al., 2006; Cheng et al., 2016; Kapur and Saraswat, 2002; Saraswat et al., 2008). The minimum distance for which using an optical interconnect becomes more efficient than a corresponding electrical interconnect is referred to as the critical length, which gets shorter as more advanced process node are used (Chen et al., 2006; Kaushik et al., 2007). The critical length is estimated by using multiple metrics, which measure the performance[32] of an optical interconnect, compared to a corresponding electrical interconnect (Miller, 2009; Rakheja and Kumar, 2012).

An on-chip optical interconnect comprises several components: light-sources, modulators, wave-guides, and

photo-detectors. Light-sources can be either on-chip, or off-chip lasers; they are being actively studied in the silicon photonics field (Li et al., 2022; Zhou et al., 2023). On-chip lasers, such as VCSEL,[33] are more efficient than off-chip lasers: they can be modulated at GHz frequencies, at the expense of increased power consumption, and reduced thermal dissipation (Amann and Hofmann, 2009). Off-chip lasers have low efficiency: they must be turned on almost all the time, to avoid light-generation delays; their integration is also more complex, since waveguides must be constructed to distribute the lasers to each on-chip optical modulator (Bai et al., 2011; Cadien et al., 2005; Peng et al., 2010). The modulator modulates light to encode data[34] (Section 2.14). The modulated lights then travel through the waveguides (Mekawey et al., 2022; Ryu et al., 1999) across the chip. Constructing on-chip waveguides to distribute the optical signals is not an easy task. It entails utilizing low-loss materials (Cadien et al., 2005), minimizing power consumption and efficiency loss (Bashir et al., 2019; Rahman et al., 2008), and finding an efficient topology and protocol for the waveguides (Le Beux et al., 2014; Werner et al., 2017).

While on-chip optical interconnects face challenges (Bashir et al., 2019; Chen and Segev, 2021), they have advantages over electrical interconnects (Cho et al., 2008). Optical interconnects enjoy minimal cross-talk and interference, allowing them to maintain signal integrity over long distances (Turkane and Kureshi, 2017). Moreover, due to wavelength division multiplexing (WDM),[35] optical interconnects can achieve significantly higher bandwidth compared to copper wires. With further improvements in light sources, modulators, and wave-guides, higher efficiency can be achieved, resulting in significantly lower energy requirements for data movement, compared to electrical interconnects (Karabchevsky et al., 2020). In addition to being used for on-chip interconnects, optical interconnects may also become candidates for connecting chiplets (Ayar Labs Inc., 2021; Hao et al., 2021; Wade et al., 2020), replacing copper wires on PCB to bridge multiple devices with massive bandwidth (Aleksic, 2017; Sharma et al., 2021), and also becoming the backbone of disaggregated infrastructure (*Part II: Disaggregated infrastructure* (Hanindhito et al., 2026)).

Cost of designing and manufacturing semiconductors

Design cost[36] of next-generation chips is expected to increase significantly, as the industry moves to smaller process nodes (Li et al., 2020). Both International Business Strategy Corporation (IBS) (Hruska, 2018) and McKinsey (Bauer et al., 2020) estimated the chip design cost to be: $28.5 M for a chip in a 65 nm process node, $51.3 M for a 28 nm process node (doubled), $106.3 M for a 16 nm technology (doubled), and $297.8 M for a 7 nm process node (tripled). The design cost of a chip in the latest 5 nm process node is estimated to be around $542 M, and the design cost for a future 3 nm process node is expected to reach $1 B. Moreover, the investment needed to build a semiconductor manufacturing facility (foundry) has also increased for smaller process nodes: $0.4 B for 65 nm, $0.9 B for 28 nm, $1.3 B for 16 nm, $2.9 B for 7 nm, $5.4 B for 5 nm, and $15 B to $20 B for 3 nm (Hruska, 2018).

The significant cost of designing and manufacturing modern chips may impact the economic feasibility of deploying custom-chips for certain scientific computing applications, especially those that are attractive to smaller markets. An alternative is to design and build chips in older and more cost-effective process nodes, especially when the absolute highest performance is not required.

Execution model, architecture, and implementation style

Since transistor scaling can no longer be a major driver of performance, there is growing interest in designing different classes of chips that are performant for specific or a group of applications. In this part, we comment on the execution model, architecture, and implementation style of different types of chips, their significance, and how they impact performance.

Execution model. The top layer of Figure 2 shows the classification of chips based on their execution model, according to Flynn's Taxonomy (Flynn, 1966, 1972). Single-instruction single-data (SISD) is used in single-core processors that execute a single instruction stream to operate on data, which was popular before the 2000s. Multi-core and many-core processors adopted the multiple-instruction multiple-data (MIMD) execution model, where each core can run different instructions and operate on different data. The single-instruction multiple-data (SIMD) execution model can be found in the vector units of CPUs, where each vector unit runs the same operations, and accesses a set of contiguous data to enable parallelization (Hassaballah et al., 2008; Raman et al., 2000). GPUs modify this concept, and use the single-instruction multiple-threads (SIMT) (Fung and Aamodt, 2011; Habermaier and Knapp, 2012), where each thread executes the same instructions and execution flow. With SIMT, any difference in branch outcomes will result in thread divergence. Accordingly, each group of threads with a different branch outcome will be executed sequentially, resulting in reduced computational efficiency. Nevertheless, SIMD or SIMT are often more efficient for explicitly parallel applications, compared to MIMD, since they reduce the overhead for processing of instructions per data item. The multiple-instruction single-data (MISD) model is less common, and is typically used when there are reliability concerns.[37]

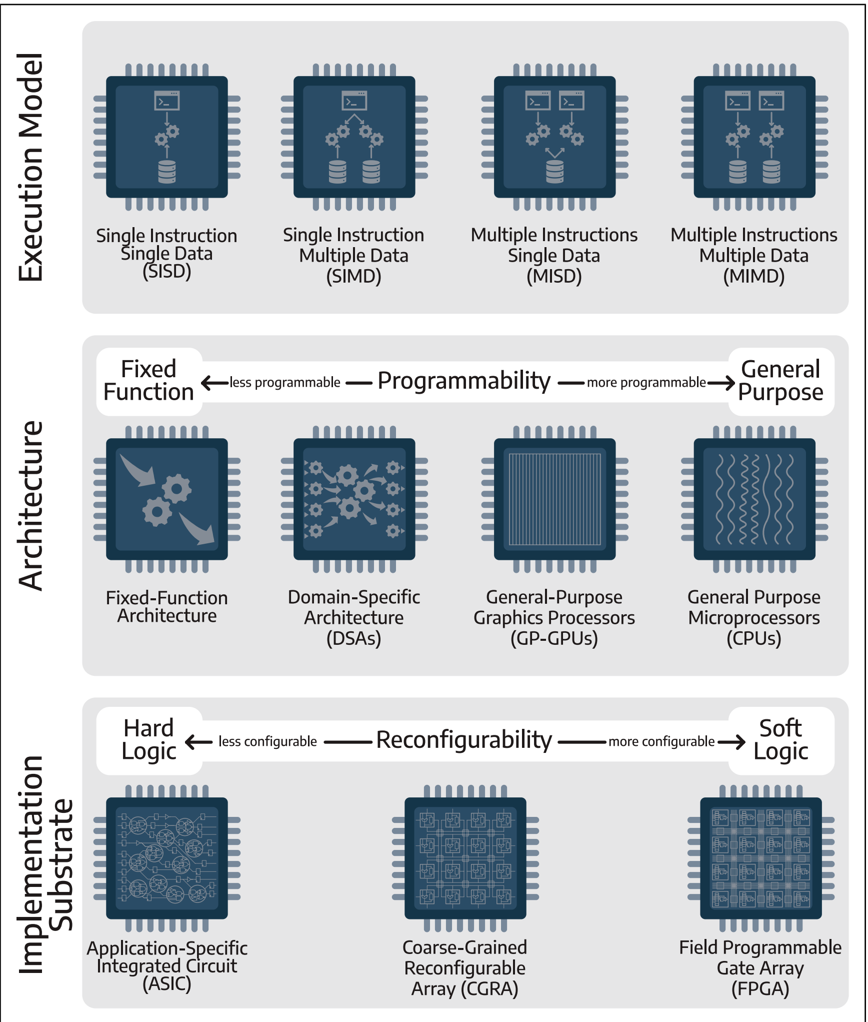

Figure 2. Execution model (top), architecture design of a chip (middle), and the substrate used for implementation (bottom). At the execution level, chips can have four types of models, according to Flynn's taxonomy: SISD, SIMD, MISD, and MIMD. At the architecture level, chips can be grouped according to their programmability. A fixed-function chip only operates on a specific input to produce a specific output, based on what operation it is designed to do. A general-purpose chip can perform various operations, based on what program it is running, and hence, is programmable. Some chips exist in between. For instance, a domain-specific architecture chip targets a wider class of applications that share common operations. At the substrate level, chips can be implemented as hard-logic, soft-logic, or somewhere in the middle. Hard logic substrates, such as ASICs, have a fixed structure of logic gates and their connections, which cannot be changed once manufactured. On the other hand, soft logic substrates, such as FPGAs, provide reconfigurable logical functions and connections even after manufacturing.

Chip architecture and programmability. The middle layer of Figure 2 shows the classification of chips at their architectural level. The architecture of a chip defines the structure of logic blocks inside the chip, how they are connected to each other, and how software interacts with them to achieve the desired functionality. A chip can be fully programmable, fixed-function, or somewhere in the middle.

A programmable architecture can interpret different instructions[38] to perform different operations (Peccerillo et al., 2022). Programmable architectures target a wide range of applications and are suitable for workloads whose algorithms or protocols constantly change (Iyer, 2012). While programmability brings versatility, it comes with performance, energy, and silicon area overheads associated with interpreting and decoding instructions (Dally et al., 2020; Hennessy and Patterson, 2019). An example of a fully programmable architecture is the general-purpose microprocessor (CPU) (Section 3), which can be used for virtually any application.[39] By contrast, a fixed-function architecture is designed to perform specific computations for a specific set of inputs, which typically provides the best performance, highest energy efficiency, and smallest silicon area footprint, at the expense of losing flexibility (Tong et al., 2006). Fixed-function architectures are suitable for applications that are mature or have a short lifetime, such as cryptographic processors (Anderson et al., 2006), analog-to-digital or

digital-to-analog converters, and multimedia encoding or decoding processors (Harrand et al., 1995; Tamitani et al., 1992), among many others (Peccerillo et al., 2022).

Some architectures may provide a limited degree of programmability for a specific class of applications, to balance performance, energy efficiency, and silicon area. These architectures may not be as versatile as a fully programmable architecture. However, they target a wider range of applications, compared to fixed-function architectures. Some examples include Graphics Processing Units (GPUs), which were originally designed for graphics applications and have become more programmable (Aamodt et al., 2018; Peddie, 2023b) to target a wide range of applications that have significant inherent parallelism, and Digital Signal Processors (DSP) (Lee, 1990), which are designed for signal processing applications, such as audio (Han et al., 1996), image (Chen and Chien, 2008), video (Kim et al., 2005), and sensor data (Gao et al., 2020). Domain-specific architectures (DSAs) (Section 4.2) (Fujiki et al., 2021; Halawani and Mohammad, 2024) target multiple applications within the same domain (Huang et al., 2017).

Implementation fabric. The chip architecture is then implemented on a fabric, where it can be a hard-logic fabric or a soft-logic fabric (Gindin et al., 2007; Paulin, 2004; Rose, 2004) (bottom layer of Figure 2). Hard logic fabric implements logical functions by using logical gates made from transistors on the silicon die, which cannot be modified or altered after the chip is manufactured. Hard logic fabric provides the best performance, energy efficiency, and mass-product cost (Abdelfattah and Betz, 2012; Gindin et al., 2007). Since a hard-logic chip is not reconfigurable after manufacturing, it may also be called an application-specific integrated circuit (ASIC). Examples include CPUs released by Intel and AMD, GPUs released by Intel, AMD, and NVIDIA, and many hardware accelerators, some of which are discussed in Section 4.

A soft logic fabric permits reconfigurability after manufacturing, by modifying the logical functions of the logic elements and their interconnections (Peccerillo et al., 2022). Field-programmable gate arrays (FPGAs) (Mencer et al., 2020) are commonly used as soft-logic fabrics that provide fine-grained reconfigurability. This allows users to reconfigure the programmable logic elements and their interconnection through a hardware description language (HDL) (Riesgo et al., 1999), such as Verilog (Dubey, 2009), VHDL (Shahdad et al., 1985), or using high-level synthesis (HLS) from C code (Nane et al., 2016). The reconfigurability comes with a price: architectures implemented in FPGAs can not compete with ASICs in terms of performance, energy efficiency, and silicon area footprint[40] (Boutros et al., 2018; Kuon and Rose, 2006; Zahiri, 2003). Soft-logic fabrics (e.g., FPGAs) are suitable for implementing architectures that are likely to be changed in a short timeframe (Gandhare and Karthikeyan, 2019; Leong, 2008), performing architectural validation before manufacturing them into ASICs (e.g., pre-silicon verification and prototyping) (Huang et al., 2011; Ray and Hoe, 2003), or manufacturing low-volume, fast-time-to-market, and short-lived products (Gandhare and Karthikeyan, 2019; Marquardt et al., 2000).

Besides hard-logic (e.g., CPUs) and soft-logic (e.g., FPGAs) fabrics, Coarse-Grained Reconfigurable Arrays (CGRAs) lie in the middle to provide some degree of reconfigurability for a specific class of applications (Prabhakar et al., 2020). In contrast to FPGAs, whose reconfigurability is performed at the lowest level logical functions (i.e., boolean algebra), CGRAs provide hard-coded arithmetic and logic building blocks that are optimized for specific domains of applications and can be configured to operate in a specific arrangement to perform a larger functionality (Peccerillo et al., 2022). This allows CGRAs to provide increased hardware efficiency, energy efficiency, faster reconfiguration times, and increased performance, through more specialized functional units (Tan et al., 2021; Wijerathne et al., 2022), bringing CGRAs closer to the spectrum of a hard-logic fabric in terms of power, performance, and silicon area (Liu et al., 2019; Niu and Anderson, 2018). We highlight some chips that use CGRA in *Examples of custom and specialized hardware*.

Off-chip interfaces and pin limitations

A modern chip uses hundreds to thousands of pins (Figure 3) as conduits for power delivery and data exchange with (off-chip) peripheral devices when it is mounted[41] on a motherboard. Additional pins are typically needed if a chip requires more memory bandwidth,[42] or needs more inter-device bandwidth to connect to peripheral devices[43] (Burger et al., 1996; Chen et al., 2017). Moreover, due to increasing power requirements, more pins are needed to deliver the electrical current to the microprocessor (Stanley-Marbell et al., 2011). In modern chips, about half of the pins are used to supply power, and the remaining pins are used for data exchange.[44] While the transistors have become much smaller, the pins have not enjoyed the same level of miniaturization due to physical limitations for carrying electrical current and maintaining physical strength. Adding more pins is expensive. Since both the pin and the pitch[45] are not getting significantly smaller anymore, it increases the size of the chip package. A larger package size entails a more complicated mounting mechanism, as it becomes more difficult to maintain uniform pressure for providing sufficient contact for all the pins to the socket (Chow et al., 2006).

Architecture of communication interfaces

A communication interface enables communication of a microprocessor with other microprocessors, accelerators, or

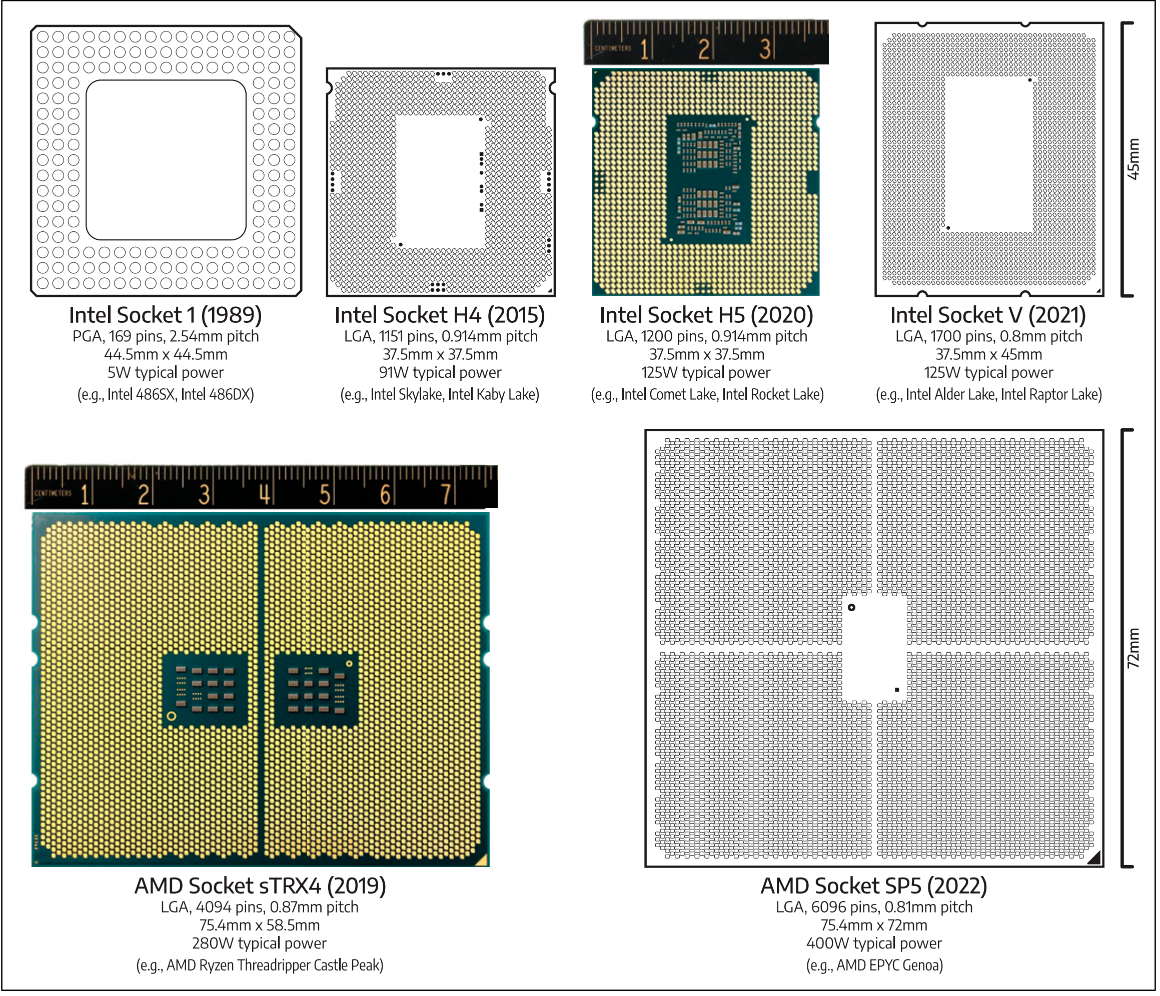

Figure 3. Evolution of microprocessor socket and pin (back of the processor), drawn to scale. The Intel socket 1 is a pin-grid array (PGA) socket, containing 169 pins, with 2.54 mm pitch (i.e., the distance between the center of pins). Launched in 1989, it was designed to accommodate Intel 486 SX, Intel 486 DX, and Intel 486 DX2 microprocessors, with 1.2 million to 1.6 million transistors, operating at typical power of 5 W. Almost three decades later, Intel socket H4, which is a land-grid array (LGA) socket, containing 1151 pins, spaced at 0.914 mm, was released to support Intel Skylake and Kaby Lake microprocessors, with 1.75 billion transistors, operating at a typical power of 91 W. Intel socket H4 was followed by Intel socket H5 5 years later. It retains the same socket dimension, but has 49 more pins to deliver more power (125 W vs 91 W of typical power) to Intel Comet Lake and Rocket Lake microprocessors, which have double core counts compared to their predecessor. A year later, Intel socket V, containing 1700 pins, spaced at 0.8 mm in a land-grid array (LGA), was released to support Intel Alder Lake and Raptor Lake microprocessors, operating at a typical power of 125 W. It has more connectivity than its predecessors. AMD Socket sTRX4 has 4094 pins in a land-grid array (LGA), which has the same physical dimension as AMD Socket SP3. It was launched in 2019 to support AMD Ryzen Threadripper Castle Peak, with up to 64 cores, 30 billion transistors, and 280 W of typical power. AMD Socket SP5 is a land-grid array (LGA) socket, containing 6096 pins, spaced at 0.81 mm. It was released in 2022 to support 96-core AMD EPYC Genoa microprocessors, with 90 billion transistors, 400 W typical power, 12-channel DDR5 memory, and 128 PCI Express 5.0 lanes.

off-chip memory. Parallel interfaces were popular early on. The need for higher bandwidth resulted in the development and adoption of serial interfaces. We review both next, and also highlight implementation aspects.

Parallel interface. Early interfaces were implemented by using a parallel architecture (Figure 4(a)) (Giuma and Hart, 1996; Shanley et al., 1995), which transferred data by dividing it simultaneously over multiple wires (Dawoud and Dawoud, 2020; Roman, 1998). For instance, a 32-bit-wide parallel interface needed at least 32 wires to transfer 32 bits (4 bytes) of data in one cycle, with additional wires for controls.[46] To correctly receive the full data, the receiving end must wait until all signals have arrived (Bandyopadhyay and Cases, 2000; Lee et al., 2013), before assembling them into the complete 4-byte data. While building a wider parallel interface or raising the clock frequency to increase the bandwidth may seem

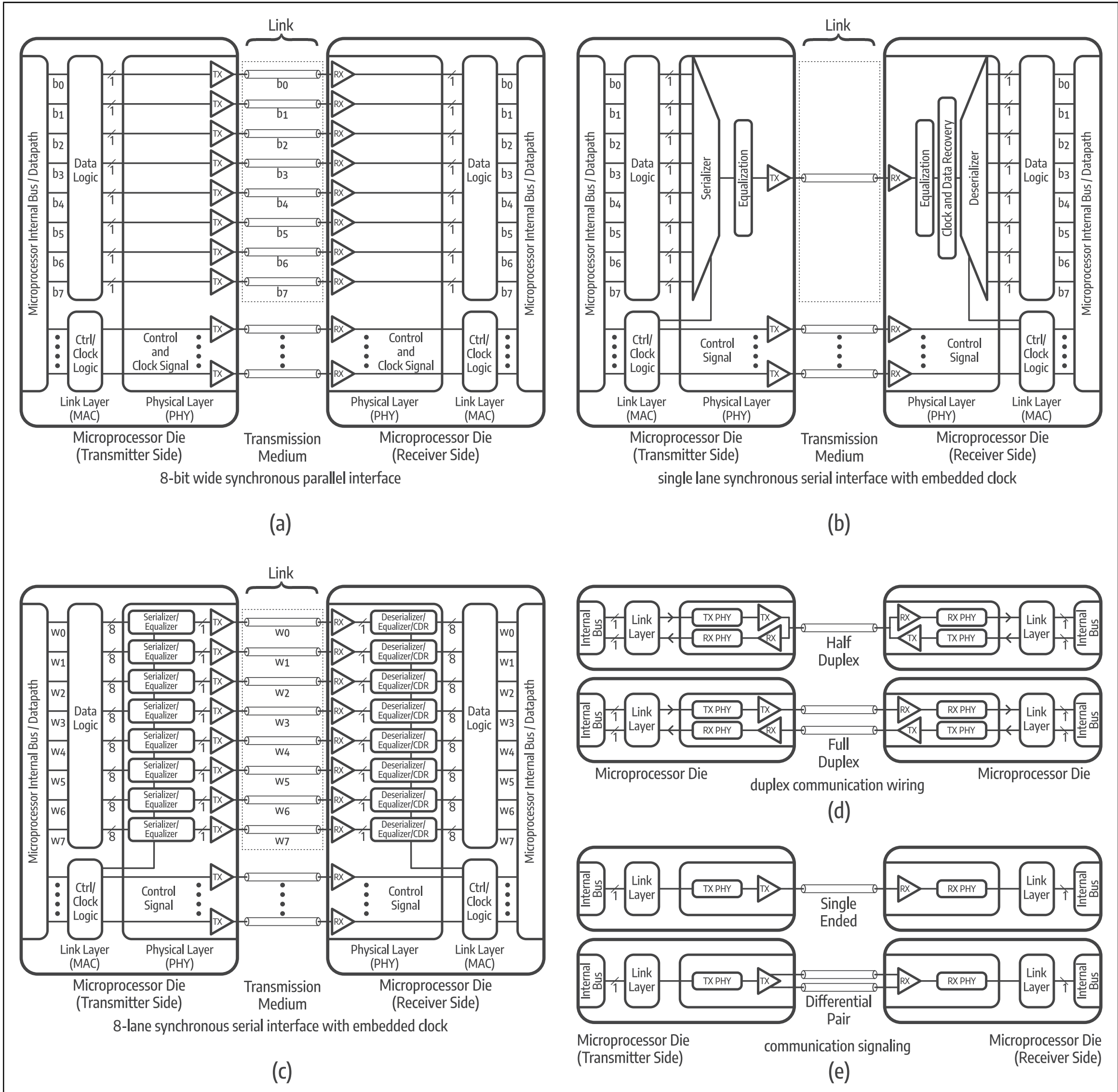

Figure 4. Simplified high-level overview of communication interfaces: (a) assume a word w consists of 8 bits of data b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, which can be transmitted in parallel at the same time (i.e., during the same clock cycle) using eight data wires along with dedicated clock signal and control signal wires on the 8-bit wide synchronous parallel interface; (b) in serial interface, a serializer takes each bit of the word and transmits it one by one per clock cycle (i.e., eight clock cycles are needed to transmit a word) using one wire along with control signal wires (i.e., in this case, the clock signal is embedded into the data signal). Although it needs more cycles to transmit the word, the serial interface can run at a significantly higher frequency compared to the parallel interface; (c) aside from increasing the clock frequency, the serial interface can have multiple lanes to increase its bandwidth. In this case, there are eight lanes, each transmits a word at the same time (w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7). Each word is 8-bit of data (b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7); (d) a two-way communication system, either parallel or serial, can be implemented as a half-duplex or a full-duplex interface. In half-duplex, only one wire is used for sending and receiving a bit, and thus both sending and receiving cannot happen at the same time. On the other hand, in a full-duplex, two wires are used for sending and receiving a bit, and thus both sending and receiving can happen at the same time, improving bidirectional bandwidth at the expense of more wires; (e) the communication link can use single-ended or differential pair signaling. Single-ended signaling is usually used for short-distance communication since it is prone to noise while differential pair signaling can be used for longer distances as it is more immune to noise.

intuitive, it is challenging in practice (Bandyopadhyay and Cases, 2000; Dawoud and Dawoud, 2020). A wider parallel interface entails more pins on the (already pin-limited) microprocessor (Sarmah and Azeemuddin, 2017), and uses more wires (Dawoud and Dawoud, 2020; Roman, 1998), which then makes PCB routing more difficult. That is also why early interfaces were half-duplex[47] (Figure 4(d)). Full-duplex interfaces require dedicated wires for sending and receiving data at the same time (Figure 4(d)) (Dawoud and Dawoud, 2020;

Summerville, 2009), doubling the number of wires, further complicating the routing and pin allocation.

Increasing the clock frequency to enable higher data transfer rate introduces signal integrity issues (Bogatin, 2011; Wu et al., 2013), including timing skew[48] (Hu and Yuan, 2009; Lee et al., 2013), and electromagnetic interference[49] (Frenzel, 2007; Karstensen et al., 2000). When wires have different lengths,[50] or are subjected to different noises and cross-talks, they may have different arrival times (skew). Accordingly, implementing a wider parallel interface becomes more challenging at higher clock frequencies: the tolerance window in which the receiver can wait for all signals to arrive becomes shorter, whereas the skew increases due to higher parasitic capacitance, noise, and cross-talk. These challenges led to the development and adoption of serial interfaces.

Serial interface. Modern communication interfaces rely on serial architectures to limit the number of pins on a processor, as well as wiring on PCB. Since the 2000s, interfaces were implemented through a serial architecture,[51] where each bit-line was implemented by using either one wire for a single-ended interface, or two wires for a differential pair interface[52] (Figure 4(e)) (Chen and Katopis, 2004; Mechaik, 2001). To increase the bandwidth, multiple serial lanes are used to form a wider serial interface (Figure 4(c)) (Sarmah and Azeemuddin, 2014, 2015) and multi-lane synchronization can be done by using a special control byte (Wu et al., 2016). Unlike the parallel interface, each serial lane is individually synchronized, and multi-lane synchronization can be performed through the use of a control byte. However, adding more lanes would then need more wires and pins on the microprocessor, leading to increased routing complexity, and higher implementation costs (Na et al., 2017; Sreerama et al., 2018). Moreover, adding more lanes also necessitates using a dedicated physical layer for each lane, which consumes area on the silicon die, and increases energy consumption (Abdennadher et al., 2020; Rashdan et al., 2020).

The physical layer of the serial interface. Serializer-Deserializer (SerDes) implements the physical layer[53] of the serial interface,[54] and its performance directly impacts communication speed. SerDes converts a parallel stream of data into a serial stream of data; once the serial stream of data has been transmitted and obtained by the receiving end, SerDes (on the receiving end) converts it back into the parallel stream[55] (Figure 4(b)) (Frenzel, 2007; Ko, 2022). SerDes resides inside the chips, and is the heart of a serial interface (Rashdan et al., 2020). SerDes also supports inter-node communication interfaces, such as Ethernet (Law et al., 2013), and Infiniband (*Part II: Inter-node communication* (Hanindhito et al., 2026)).

The need for higher bandwidth has pushed for significant improvements in SerDes data rate: a factor of 200 during the last 20 years (Table 1). These improvements have been enabled by transistor scaling and using denser data modulation schemes. Specifically, transistor miniaturization allows integration of more advanced Digital Signal Processing (DSP) blocks into SerDes (Fujimori, 2014; Tonietto, 2022; Yue and Shekhar, 2022). The advanced DSP then supports high-order modulation schemes, resulting in higher SerDes data-rate. Using a more advanced modulation scheme can possibly push the data rate beyond 224 Gbps by 2030 (Che and Chen, 2023; Hecht et al., 2022; Yue and Shekhar, 2022). However, this could be challenging since it increases the complexity of SerDes circuitry, and results in more power consumption (Rashdan et al., 2020).

Signal coding and modulation

We highlight techniques and technologies that are used to encode and modulate data for transmission through a medium (e.g., metal wires or optical interconnect). They play a fundamental role in future bandwidth improvements by allowing a signal to carry more data over a long distance. Advanced data encoding and modulation are especially important as they are among the very few options that can alleviate communication bottlenecks.

Before data could be transmitted through a medium that carries either electrical or optical signals, data is encoded through encoding schemes. This improves spectral efficiency[56] (Rysavy, 2014; Winzer, 2012), and enables error

Table 1. SerDes data rate based on optical internetworking forum (OIF).

Standard	Year	Data rate	Example interfaces
SPI-3	2000	0.1 Gb/s	SONET 622 Mb/s
SPI-4.2	2001	0.8 Gb/s	HyperTransport 1.0
SxI-5	2002	3.1 Gb/s	SATA 2.0, SAS-1, InfiniBand SDR
CEI-6G	2004	6 Gb/s	SATA 3.0, SAS-2, 4G/8G Fibre Channel, HyperTransport 3.1, Infiniband DDR
CEI-11G	2005	11 Gb/s	SAS-3, 16G Fibre Channel, InfiniBand QDR
CEI-28G	2008	28 Gb/s	SAS-4, 32G Fibre Channel, InfiniBand EDR
CEI-56G	2017	56 Gb/s	64G Fibre Channel, InfiniBand HDR, 50G/100G/200G Ethernet (802.3 cd)
CEI-112G	2022	116 Gb/s	InfiniBand NDR, 100G/200G/400G Ethernet (802.3ck)
CEI-224G	TBD	232 Gb/s	Terabit Ethernet (802.3dj)

detection and error correction (Alyaei and Glass, 2009; Fair et al., 1991; Imai and Hirakawa, 1977), which are needed for maintaining signal integrity (Mercier et al., 2010; Seshadri et al., 1993; Tzimpragos et al., 2016). Then, modulation is performed to transform the data into suitable signals for transmission over long distances. The primary goal of encoding and modulation is to increase the data rate[57] while reducing the signal rate.[58] We focus on digital modulation techniques (Smithson, 1998; Xiong, 2006) since they are widely used in intra-node and inter-node communication (*Part II: Inter-node communication* (Hanindhito et al., 2026)).

Digital baseband[59] modulation, also known as line coding (Guri et al., 2015; LoCicero and Patel, 2018; Matin, 2018; Rezaei et al., 2023; Teixeira and Zaharov, 2007), is used to encode data into a pattern of voltage, current, or photons for short-distance communication through metal wires or optical cables (Loan, 2007; LoCicero and Patel, 2018). The choice of line coding depends on several considerations, such as timing and synchronization capability, power efficiency and electrical characteristics, error detection and correction capability, probability of error and noise tolerance, and complexity of transmitter and receiver design (Couch, 1994; Lathi and Ding, 2022). Line coding comprises two parts: pulse shaping and block coding.

Pulse shaping. Pulse shape defines the pattern of voltage, current, or photons that transmit information at the bit level. Commonly-used pulse shapes can be grouped into five categories: unipolar, polar, bipolar, multi-transition, and multi-level (Madhow, 2008; Rezaei et al., 2023). The unipolar, polar, and bipolar pulse can be either non-return to zero (NRZ) or return-to-zero (RZ), resulting in several combinations.[60] Herein, we only highlight line coding techniques that are used by communication technologies that are mentioned in later sections. Interested readers are suggested to consult (Madhow, 2008) for a detailed comparison between pulse schemes.

In NRZ, the non-zero voltage pulse maintains its voltage level throughout the bit-time.[61] With this scheme, NRZ does not provide enough signal transition to help distinguish each bit during long consecutive transmissions of the same bit value, leading to synchronization problems (Anand and Razavi, 2001; Song and Soo, 1997). RZ tries to make sufficient transition, by making the non-zero voltage pulse return to zero in the middle of the bit-time. This allows the receiver to distinguish each bit, in the case of long consecutive transmission of the same bit value.[62] In addition to greater complexity, the RZ data rate is half that of NRZ for the same signal rate.

Due to its simplicity, NRZ is often used for bit rates up to 40 Gbps (Chen et al., 2023; Van Kerrebrouck et al., 2019). However, implementing higher-bandwidth communication interfaces (Section 2.13) necessitates finding denser pulse shapes, such as those that use Pulse-Amplitude Modulation (PAM), which uses multi-level voltage to represent the digital symbols. For instance, PAM-4 uses four levels of voltage to represent 2-bit symbols, in order to increase the modulation density. This choice increases the complexity of implementing transmitter and receiver, and raises signal vulnerability to noise and cross-talk[63] (Chen et al., 2023; Forghani and Razavi, 2022; Van Kerrebrouck et al., 2019), resulting in vastly higher bit error rate.[64] Increasing the operating voltage[65] to address the issue is not preferred due to higher energy consumption (Garcia et al., 2007; Müller et al., 2015) Higher-order PAM, such as PAM-6 or PAM-8, are currently being developed to fulfill the needs for future data rates (Che and Chen, 2023; Hecht et al., 2022; Yue and Shekhar, 2022).

Block coding. Although pulses are sufficient for transmitting information at the bit level, finding a more efficient scheme is necessary to fulfill the demand for high-bandwidth communication interfaces. This is where block coding can help. In addition to higher bandwidth, block coding also provides signals with higher power efficiency[66] and self-synchronization capabilities, which NRZ and PAM alone cannot achieve. Block coding is used with both NRZ and PAM, and is being used in recent high-bandwidth communication interfaces.[67]

Block coding divides the data into fixed-length blocks, and encodes each block with slightly longer blocks, according to predefined coding schemes. Popular block codings include 4B/5B (Robe et al., 1993), 2B1Q (Sugimoto et al., 1989), 8B6T (Buchanan, 1999), and 8B/10B (Wang et al., 2010). For instance, the 8B/10B block coding, used in many communication interfaces,[68] encodes 8-bit blocks (or 8-bit words) into 10-bit symbols. This increases power efficiency and provides self-synchronization capability, at the expense of 2-bit overhead for every 8-bit of transmitted data. Denser block codings have lower overhead. These include 64B/66B[69] (Balasubramanian et al., 2011; Mohapatra et al., 2017), 128B/130B[70] (Mhaboobkhan et al., 2019; Weng et al., 2021), 242B/256B, and 512B/514B (Cideciyan et al., 2013; Teshima et al., 2008), and are used for communication interfaces that require even higher bandwidth.

General-purpose microprocessors

Central Processing Units (CPUs) are designed to support various instructions in diverse workloads, and hence, are called general-purpose microprocessors (Gelsinger, 2001) (Figure 2). Through using multiple complex hardware structures, they are designed to maximize average performance[71] for a wide range of applications. While this is expensive in terms of implementation[72] and energy consumption (Bhandarkar, 1997; Blem et al., 2013), it makes programming CPUs easier for software developers.

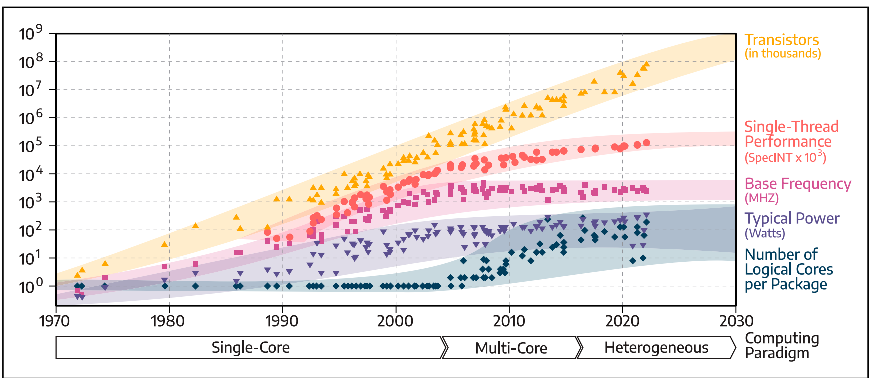

Figure 5. Technology trends of general-purpose microprocessors since the 1970s, with projections up to 2030.

We review how technology trends in computing are impacting CPUs. In this vein, Figure 5 exhibits the evolution of key characteristics of general-purpose microprocessors during the past five decades, along with future projections.[73] These characteristics include transistor counts, single-thread performance, base clock frequency, typical power consumption, and the number of logical cores. Both the transistor counts and the number of logical cores are given per package. Moreover, computing paradigm (Hennessy, 2021) is also shown in the figure to indicate how they evolved over the years. Among these five characteristics, the clock frequency (Leiserson et al., 2020; Xiu, 2017) is perhaps the most widely-known feature, due to its publicity in marketing for decades.

We discuss the key drivers of these trends. The end of Moore's law and Dennard scaling significantly impacted the evolution of general-purpose microprocessors. Therefore, we partition the discussion of the trends to before and after the end of Dennard scaling, followed by future possibilities.

Trends between 1980s and early 2000s

The clock frequency increased significantly between the 1980s and the early 2000s, as both Intel and AMD were competing to develop their speed-demon chips, in which higher clock frequency was the main figure of merit a microprocessor (Ronen et al., 2001) used for marketing (Olukotun and Hammond, 2005). Meanwhile, transistor sizes continuously decreased with more advanced process nodes (Bohr, 2007; Gargini, 2017), from 10 µm during the 1970s to 32 nm in late 2000s (Figure 1). During this period, Dennard scaling (Bohr, 2007; Dennard et al., 1974) still held, allowing manufacturers to raise the clock frequency and reduce power costs per transistor switching, thus keeping the overall power under control. Although the increase in the clock frequency improved single-thread performance, measured through SPEC Integer benchmark[74] (Dujmovic and Dujmovic, 1998), this design strategy came with costs. It increased dynamic power consumption (Liu and Svensson, 1994) and made thermal dissipation more challenging (Gurrum et al., 2004; Kish, 2002).

The advancement in the process node also allowed designers to pack more transistors in the same area, resulting in the steady increase of the number of transistors per package of microprocessors between the 1980s to early 2000s. These transistors were used to implement more complex functional units, which improved single-thread performance[75] through the out-of-order execution engine,[76] vector extensions,[77] branch predictor,[78] and improvements in the memory system (Peleg and Weister, 1991). During this period, the increase in performance and transistor count followed Moore's law.

The end of Dennard scaling and emergence of multi-core processors

In the mid-2000s, both clock frequency and typical power started to plateau (Figure 5) (Theis and Wong, 2017). Clock frequency lost its significance as the key figure of merit in microprocessor design. Power efficiency became important (Zyuban and Kogge, 2000, 2001) due to the end of Dennard scaling (Esmaeilzadeh et al., 2011a; Wang and Skadron, 2013). Parallelism became fundamental to delivering higher performance. Instead of increasing the clock frequency or adding more features to improve single-thread performance, which came with disproportional increases in overhead and hence decreases in performance and power efficiency, designers built multi-core processors whose cores were architected to provide a good balance between performance and power consumption. This trend has continued, and the number of cores in a microprocessor has increased since then (Geer, 2005; Gepner and Kowalik, 2006; Parkhurst et al., 2006). Table 2 shows the increase in the number of cores in recent datacenter class microprocessors. The paradigm shift into multi-core microprocessors (Figure 5) made parallel programming crucial for fully utilizing the compute power offered by each core (Blake et al., 2009; Marowka, 2011). Applications that have inherent parallelism,[79] as is

Table 2. Comparison of recent, commercially-available datacenter class microprocessors.

CPU name	Year	# of cores	# Transistor (bn)	# Pins	Mem. Bw. (GB/s)	Bw. per core (GB/s)	TDP (W)
Intel E7-8894 v4	2017	24	7.2[a]	2011	85	3.54	165
IBM POWER9	2017	22	8	3899	170	7.72	190
AMD EPYC 7601	2017	32	19.2[b]	4094	170	5.31	180
Intel Xeon 8280	2019	28	8[c]	3647	131	4.68	205
AMD EPYC 7742	2019	64	39.5[d]	4094	205 s	3.2	225
Ampere M128-30	2021	128	38[e]	4926	204[f]	1.59	250
AMD EPYC 9654	2022	96	90[g]	6096	461	4.80	360
Intel Xeon 8490H	2023	60	48[h]	4677	307	5.11	350

[a]Intel Broadwell-EX features up to 24 cores, using a High Core Count (HCC) die, which measures 456.12 mm^2 in area and contains 7.2 billion transistors.
[b]Each multi-chip module (MCM) has 4.8 billion transistors, for a total of 19.2 billion transistors (Naffziger et al., 2021).
[c]Estimated from its predecessor, Intel Xeon Platinum 8180 (Skylake-SP), with similar core count and process node.
[d]Each core chiplet has 3.9 billion transistors. The (I/O die) IOD chiplet has 8.3 billion transistors (Naffziger et al., 2021).
[e]Number is estimated from the approximate die size of each ARM Neoverse N1 core, with 1 MB L2 cache (1.4 mm^2), L3 cache (Part II Section 2.1.2 (Hanindhito et al., 2026)), integrated memory controller, and PCIe interface, for a total area of 400 mm^2, along with TSMC 7 nm transistor density of around 95 million transistors/mm^2.
[f]Approximate memory bandwidth from memory configuration, with 8 channels of DDR4-3200 ECC (Ampere Computing LLC, 2022).
[g]Each core chiplet has 6.57 billion transistors. The IOD chiplet has 11 billion transistors.
[h]Each chiplet has an area of 400 mm^2, containing approximately 11–12 billion transistors (Nassif et al., 2022).

often the case in high-performance computing, and machine learning, are well-suited to exploit multi-core processors.

While increasing the number of cores seems to be an intuitive way of improving the performance of microprocessors, several factors limit the number of cores that can be placed into a microprocessor. These are:

Memory bandwidth bottlenecks. Increasing the number of cores elevates pressure on the memory subsystems (Ahn et al., 2009; Borkar, 2007; Mandal et al., 2010; Sancho et al., 2010), as higher bandwidth would be needed to supply the required data to the cores. Insufficient bandwidth may cause cores to wait for memory accesses (Cristal et al., 2005). Microprocessors with a high number of cores usually have multiple memory channels to accommodate the demand for memory bandwidth (Sancho et al., 2010). Adding more memory channels is costly because it needs additional pins (Figure 3), which are already limited, and it necessitates more memory controllers that consume area on the silicon die. Moreover, a more sophisticated motherboard design will be needed in order to house more memory modules, and maintain signal integrity.

Resource contention and core synchronization. Typically, a core needs to communicate with other cores to share data, share resources, and perform synchronization barriers. Increasing the number of cores in a microprocessor makes synchronization and resource sharing more difficult (Blagodurov et al., 2010; He et al., 2017; Zhuravlev et al., 2010). Extensive inter-core communication and resource contentions can limit the attainable performance of multi-core processors (Hood et al., 2010; Xu et al., 2010).

Cache Coherency. As the number of cores increases, maintaining coherency across multiple levels of cache becomes more difficult (*Part II: Registers* (Hanindhito et al., 2026)).

Transistor count, die size, and manufacturing yield. Adding more cores to a microprocessor increases the transistor count. With the slowdown of transistor scaling, a larger die size is then needed, which lowers the yield and increases manufacturing costs (Mack, 2015; Sun et al., 2020) (Section 2.5). To overcome these challenges, since 2017, microprocessors started to use multi-chip module (MCM) technology[80] (Naffziger et al., 2021), followed by chiplet[81] (Loh et al., 2021; Naffziger et al., 2020) and 3D stacking technologies[82] (Agarwal et al., 2022; Beyne et al., 2021; Ingerly et al., 2019; Su et al., 2017). Figure 6 shows the evolution of packaging technologies used by modern microprocessors.

Power consumption and heat dissipation. Packing more transistors on a silicon die, and the end of Dennard scaling (Esmaeilzadeh et al., 2011b; Wang and Skadron, 2013), has resulted in higher power consumption, which is one of the major limiting factors of adding more cores (Horowitz, 2014; Tiwari et al., 1998). To overcome this difficulty, manufacturers typically lower the clock frequency of higher-core-count microprocessors (Gepner and Kowalik, 2006), which resulted in a slower increase in power consumption since 2005 (Figure 5). The use of dynamic voltage and frequency scaling (DVFS) (Herbert and Marculescu, 2007; Le Sueur and Heiser, 2010; Papadimitriou et al., 2019) makes up the performance loss due to the lower clock frequency, by allowing the microprocessor to momentarily increase its power

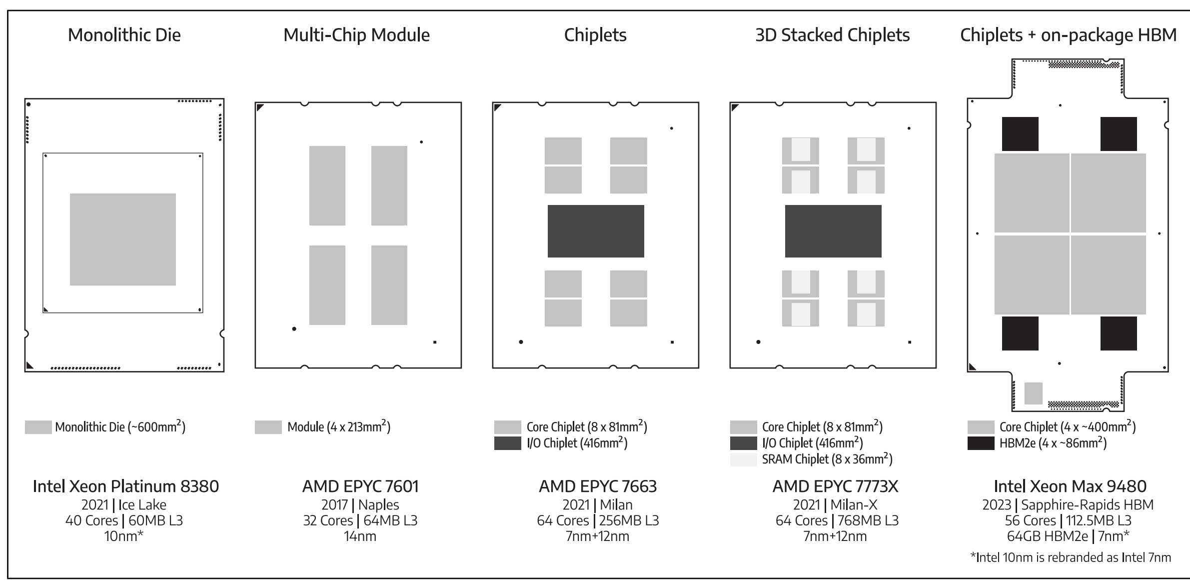

Figure 6. Evolution of microprocessor packaging technology, drawn to scale. Until recently, microprocessors had a single silicon die on a package. Nowadays, more transistors and more cores are placed on a microprocessor die. The slowdown in the advancement of process nodes and transistor shrinking has resulted in larger silicon die sizes, aimed at fitting more transistors on a die. For instance, the Intel Xeon Platinum 8380 is a 40-core microprocessor, implemented by using a single monolithic die, whose size is approximately 600 mm^2. A larger silicon die raises the possibility of manufacturing defects, which lowers the yield, and increases the manufacturing cost (Mack, 2015; Sun et al., 2020) (Section 2.5). To improve the yield, AMD uses multiple identical silicon dies (each single die is able to function as a stand-alone die) within a single package, referred to as a multi-chip module (MCM). For instance, the 32-core AMD EPYC 7601 microprocessors have four silicon dies, each containing eight cores, and have a size of 213 mm^2, for a total of 852 mm^2 per package (Naffziger et al., 2021). The next advancement in packaging technology uses multiple silicon dies, referred to as chiplets. Each chiplet may have a different functionality and process node. For instance, the 64-core AMD EPYC 7663 consists of 33 million transistors, implemented in an eight-core complex die, manufactured in a 7 nm process node (8 × 81 mm^2 in size), and one I/O die manufactured in a 12 nm process node (416 mm^2 in size), for a total of 1064 mm^2 of silicon die in a package (Naffziger et al., 2021). Chiplets can be stacked on top of each other by using 3D packaging technologies. An example includes AMD EPYC 7773X, where SRAM cache chiplet (in light-grey) is stacked on top of each core complex die, providing triple the capacity of the last-level cache (256 MB on AMD EPYC 7663 vs 768 MB on AMD EPYC 7773X) (Agarwal et al., 2022). The 56-core Intel Xeon Max microprocessors have four chiplets (each is presumed to have a size of 400 mm^2) and integrate HBM2e memory (in dark grey) on the same package (Sanca and Ailamaki, 2023).

consumption to boost an individual core's clock frequency to achieve higher single-thread performance, as long as it is within its power and thermal envelopes.[83] This feature is specifically useful for legacy applications that cannot take advantage of all the available cores (Cochran et al., 2011). Another method to balance power consumption is through heterogeneity, by combining high-performance and low-power cores. This approach reduces power when running workloads that do not heavily stress the CPU. Notable examples are the ARM big.LITTLE and Intel Alder Lake architectures (Rotem et al., 2022). Nevertheless, high-core-count microprocessors will soon require a liquid cooling system for effective heat dissipation.

Pin Limitation. With the increase in power consumption, higher memory bandwidth requirements, and the need for faster connectivity to off-chip peripherals, modern microprocessors need more pins, which are expensive[84]. Figure 3 shows the pins and the package size used by several microprocessors, as well as their evolution over time. Specifically, it shows that, while the number of transistors on a chip has increased by about three orders of magnitude during the past three decades, the number of pins have increased by only about an order of magnitude, exacerbating communication bottlenecks.

Many-core processors

Applications that have inherent parallelism, which are abundant in high-performance computing and machine learning, typically have simple execution flows[85] (Mittal, 2020a; Véstias and Neto, 2014). Advanced branch predictors, aggressive speculative execution engines or features like simultaneous multi-threading (SMT) do not typically benefit these workflows. These applications can enjoy larger performance improvements if area on a silicon die is allocated to build a large number of simple cores, instead of fewer but more sophisticated ones (Carter et al., 2013; Narayanan et al., 2015). More cores allow these applications to improve performance through parallelization (Schmidl et al., 2013; Silva et al., 2019).

Manycore processors, followed by hardware accelerators, such as GPUs, have been a response to this need. We highlight the basic design philosophy of manycore processors through an example.

As summarized in Table 3, about the same amount of silicon die and pins were used to produce two different processors: Knights Landing has more simple cores (many-core processor), and is suited for highly-parallel and low-execution-complexity workloads; Skylake-SP has small number of sophisticated cores (multi-core processor), and is more suited for applications that may have unpredictable behavior. Specifically, the Intel Xeon Phi[86] Knights Landing has up to 72 cores, implemented with 7.1 billion transistors, on a 682 mm^2 silicon die. Each core uses the Airmont micro-architecture (Farrell et al., 2017), which has a simpler design when compared to the Skylake micro-architecture, used in Intel Xeon codenamed Skylake-SP (Tam et al., 2018), which is a multi-core processor introduced in 2017. Both Knights Landing and Skylake-SP share the same LGA-3647 socket (i.e., they have roughly the same off-chip memory bandwidth), and are manufactured on the same 14 nm process node. However, Skylake-SP can only have up to 28 cores, which are implemented with 8 billion transistors, on a 694 mm^2 silicon die. Both Knights Landing and Skylake-SP can execute x86-64 instruction sets, which implies applications that run on Skylake-SP can also run on Knights Landing; this eases the migration of applications between the two microprocessors (Wang et al., 2014). On the memory side, Knights Landing features on-package, 16 GB Multi-Channel Dynamic Random Access Memory (MCDRAM) (Pohl and Sattler, 2018; Salehian and Yan, 2017), which is 3D stacked DRAM (*Part II: High-Bandwidth Memory (HBM) and variants* (Hanindhito et al., 2026)). This on-package DRAM provides more than 400 GB/s of memory bandwidth (Sodani, 2015) in addition to the off-chip 6-channel DDR4 memory, which provides 115.2 GB/s bandwidth. Despite seamless code migration, optimization would still be needed to achieve higher performance on Knights Landing (Fang et al., 2014; Mittal, 2020b) and for taking advantage of its high-bandwidth MCDRAM (Butcher et al., 2018; Peng et al., 2017).

The last generation of Intel Xeon Phi was Knights Mills[87] (2017), which was specifically designed for accelerating AI and ML workloads (Domke et al., 2019; Georganas et al., 2018). Intel Xeon Phi offers substantial performance gains for HPC, AI, and ML workloads, due to abundant data-level parallelism and simple execution flows (Mittal, 2020b; Shao and Brooks, 2013). However, it faced strong competition from GPUs (Mittal, 2020a), which forced Intel to discontinue Xeon Phi product lineup in 2020. Nevertheless, the spirit of many-core computing is still alive in other areas, such as cloud computing clusters[88] that host small applications and micro-services in containerized forms (Pahl et al., 2019; Singh and Singh, 2016) for multiple tenants. These micro-services that are hosted in the cloud are typically not as computationally demanding as HPC or ML applications. Therefore, a simpler and more energy-efficient core is often preferred: many-core processors allow the cloud provider to achieve higher compute density through higher aggregate number of cores per server rack, and improve energy efficiency, which decrease the total cost of ownership (TCO). This has lead to the development of microprocessors designed specifically for cloud computing clusters, such as AMD EPYC Bergamo (2023), which has 128 Zen4c cores[89] that are optimized based on performance-per-watt. Intel is expected to release (2024) a new product lineup for cloud computing with its Sierra Forest processor, which is estimated to have 288 (energy) efficiency-oriented cores.

Several semiconductor companies are increasingly adopting ARM-based processors due to their lower power consumption, making them a strong alternative to Intel and AMD's x86 architectures. This shift has long been evident in consumer devices—Apple's M-series (MacBooks) and A-series (iPhones), as well as Qualcomm's Snapdragon processors, all use ARM microarchitectures. More recently, major cloud providers have extended this trend to their infrastructures, integrating ARM processors like Amazon AWS Graviton (Loghin, 2024), Microsoft Azure Cobalt, and Google Cloud Axion. Ampere Computing also develops ARM-based many-core server solutions like AmpereOne, incorporating up to 192 cores, with future models expected to reach 256 and 512 cores. NVIDIA develops Grace CPU based on ARM architecture (Evans, 2022) as part of their CPU-GPU heterogeneous system (*Part II: System integration and heterogeneous computing* (Hanindhito et al., 2026)).

Table 3. Comparison of multi-core (Intel Xeon) and many-core (Intel Xeon Phi) processors.

	Multi-core	Many-core
Product Name	Intel Xeon 8180	Intel Xeon Phi 7290
Year	2017	2017
Code Name	Skylake-SP	Knights Landing
Core Architecture	Skylake	Airmont
Socket	LGA-3647	LGA-3647
Number of Transistors	8 Billion	7.1 Billion
Silicon Die Area	694 mm^2	682 mm^2
Process Node	14 nm	14 nm
Number of Cores	28 Cores	72 Cores
Base Clock Frequency	2.50 GHz	1.50 GHz
On-package DRAM	N/A	MCDRAM
Bandwidth	N/A	400 GB/s
Off-package DRAM	DDR4-2666	DDR4-2400
Bandwidth	119.21 GB/s	115.2 GB/s
Typical Power	205 W	260 W

Programmability

CPU is the most flexible hardware platform, as the Instruction Set Architecture (ISA) allows them to execute any

workload and application with relative ease. CPUs can be coded through several programming languages: from machine level (e.g., x86 or ARM assembly), to mid-level (e.g., C/C++), and to high-level (e.g., Python). Compiler tools (e.g., gcc or LLVM) are responsible for converting high-level software code to machine binary code. There exist several tools to extract parallelism from multi-core CPUs. For instance, in C, programmers may use either pthread, a low-level execution model compliant with most operating systems, or OpenMP, a library that offers an easy-to-use API.

While highly flexible and easy to use, extracting the maximum amount of performance out of a CPU is not trivial. Although the compilers offer many optimization techniques, their generated machine code is typically sub-optimal. In order to efficiently utilize a CPU, the programmer needs to be aware of hardware-related features of the underlying CPU architecture.[90]

What comes next?

Since chip designers have limited options to improve performance further, the next generation of general-purpose microprocessors is expected to have features that are purpose-built and optimized for specific use cases:

- *Single thread performance.* General-purpose microprocessors will only see slight improvements in individual core performance due to the diminishing returns of adding more transistors to implement a more complex core. With slight performance improvements between generations, manufacturers will, once again, rely on (slightly) increasing the clock frequency to improve performance across generations.[91] Due to the performance improvement stagnancy, it is expected that manufacturers will include more on-chip accelerators to offload popular workloads, such as machine learning (Khaldi et al., 2021; Tukanov et al., 2022), data analytics (Sanca and Ailamaki, 2023), data encryption (Biswas, 2021), and network applications (Nassif et al., 2022) (Section 4).
- *Number of cores.* The number of cores is expected to increase to improve aggregate performance for highly parallelizable applications.
- *Core architecture and heterogeneity.* Manufacturers will continue to develop multiple variants of core architecture: performance-oriented cores, to achieve the highest performance-per-core; and efficiency-oriented cores, to achieve the highest performance-per-watt. This allows manufacturers to develop different product-lines with different core architectures to address specific needs[92] (Sideco, 2023) or integrate both of the different core architectures on the same die or in the same package (Rotem et al., 2022; Vasilakis et al., 2017).
- *Memory bandwidth.* Improvements have been made by integrating high-bandwidth memory (HBM) into the same package as the CPU (Sanca and Ailamaki, 2023; Shipman et al., 2022), through the use of higher-bandwidth memory modules (DDR5), and by adding more memory channels (*Part II: Memory systems* (Hanindhito et al., 2026)). HBM will provide significant speed-up for applications that fit inside the memory, such as machine learning inference, where models typically fit in HBM (*Part II: High-Bandwidth Memory (HBM) and variants* (Hanindhito et al., 2026)).
- *Packaging.* Manufacturers will employ even larger package sizes to increase the number of pins in order to deliver more power and provide more connectivity, while continuing to use chiplets,[93] along with new substrate material, such as glass substrate (Kudo et al., 2021; Vanna-Iampikul et al., 2023), to improve connectivity between the chiplets in a package. For instance, Intel's next-generation data center processors, code-named Granite Rapids and Sierra Forest, are expected to have an LGA 7529 socket, which has 7529 pins (60% increase over the current LGA 4677, used by Sapphire Rapids and Emerald Rapids).

Summary and remarks

When Moore's law was alive, CPUs were becoming steadily faster at the same price tag. Improvements in performance were primarily due to the implementation of more complex features, thanks to transistor miniaturization. In this era, improvements in hardware would typically impact application performance directly, sometimes making it difficult to justify algorithmic modifications.

The end of Moore's law significantly impacted this trend. Adding more complex features to the chip resulted in minor impacts on performance. Moreover, power density on a chip increased due to the end of Dennard scaling, making energy efficiency a central issue in modern chip design. In this era, better performance could be realized by building multiple simpler and more efficient cores within a chip, as opposed to a single powerful but inefficient core. Multi-core processors became popular for complex parallelizable workflows, and many-core processors could provide significant performance gains to parallelizable applications that enjoyed considerable regularity and structure. Parallel algorithms became fundamental to harness the performance offered by multi- and many-core processors.

CPUs are here to stay. They need to run the operating system and control other hardware, such as GPUs. Algorithms that are hard to parallelize, or legacy software that do not get financial and technical support for modernization will continue to run on CPUs.

With more transistors needed to realize the higher number of cores, next-generation CPUs will rely on

advanced packaging to improve yield and decrease manufacturing costs. On the other hand, the number of CPU pins has grown much slower than the number of on-chip transistors, resulting in performance degradation when off-chip communication is significant. On-package high-bandwidth memory alleviates this bottleneck when the application is small enough to fit into that memory. Algorithms that have reduced off-chip communication could be valuable when considerable off-chip communication occurs. Lastly, energy considerations will result in different classes of CPUs integrated into heterogeneous systems to support different application needs: those cores that are very fast but consume a lot of energy (performance-oriented), and those that are energy-efficient and thus are slower (efficiency-oriented).

Hardware accelerators

The design philosophy of general-purpose microprocessors is to maximize average performance for a wide range of applications (Küçük et al., 2013; Smith and Sohi, 1995). To this end, their hardware strives to extract parallelism from applications through: a) out-of-order execution engine (Eyerman et al., 2009; Peleg and Weister, 1991) (instruction-level parallelism (Davidson and Jinturkar, 1995; Jouppi and Wall, 1989; Rau and Fisher, 2003; Wall, 1991)); b) speculative execution by using branch prediction (Chang et al., 1996; Modi et al., 2005), to hide instruction latency (Brekelbaum et al., 2002; Gronowski et al., 1998); and c) caching (Iyer et al., 2021; Juan et al., 1997) and prefetching (Jiménez et al., 2012; Vanderwiel and Lilja, 2000), to minimize off-chip memory access latency. These mechanisms are generally hidden from the programmer and help reduce the burden of optimizing applications. However, the hardware that is needed to implement these mechanisms is expensive, and consumes a significant amount of power.[94] They also occupy the majority of the silicon die, increasing the manufacturing cost of the chip (Dally et al., 2020; Hameed et al., 2010a). Table 4 shows how these features occupy more than 95% of the silicon die area on modern general-purpose chips, leaving the rest (around 5%) of the silicon die for implementing the integer and floating-point execution units that actually perform the sought-after computations.

Remarkably, many classes of applications may not substantially benefit from the above-mentioned exotic features[95] (Giles and Reguly, 2014) and may not even get the best execution efficiency by using them[96] (Brooks et al., 2000; Zyuban and Kogge, 2000, 2001). For these applications, which typically enjoy a lot of regularity and parallelism, the silicon area can be used more efficiently. Many-core processors and hardware-accelerators have been a response to this reality. It is worth noting that while typically a key objective for hardware specialization (D'Arnese et al., 2023; Peccerillo et al., 2022; Qasaimeh et al., 2019; Chong et al., 2014) through the use of hardware accelerators has been to improve energy efficiency (*Part II: Energy consumption of large computing centers and its implications* (Hanindhito et al., 2026)), the specialization often improves other performance metrics as well (Altaf and Wood, 2017; Hameed et al., 2010b).

A hardware accelerator (accelerator, for short) can be defined as a separate compute structure[97] that has an architecture specifically developed for the needs of a particular application or a class of applications (Hwu and Patel, 2008), and is typically connected to a general-purpose microprocessor for execution of other code that does not fit on the accelerator. An accelerator provides significant improvements in many metrics[98] when the supported applications, referred to as accelerated workloads (Nowatzki et al., 2017), are offloaded from the general-purpose microprocessor to the accelerator. There are four primary techniques that accelerators exploit to deliver performance and efficiency, compared to general-purpose microprocessors (Dally et al., 2020; Hennessy and Patterson, 2019): a) using specialized functional units to operate on particular data-types, which allows for fast execution with low overhead[99]; b) exploiting parallelism at several levels that are more efficient for particular applications, along with using optimized hardware structure[100]; c) tailoring the memory hierarchy to the application[101]; and d) reducing overhead associated with fetching and decoding instructions through specific and simplified control flow.[102] In the remainder of this section, we discuss GPUs, which are among the most well-known hardware accelerators, followed by custom-made hardware for specific applications.

Graphics processing units (GPUs)

In this part, we discuss how GPUs have evolved during the past three decades, their compute unit, memory system, and recent additions, such as matrix accelerators, to make GPUs attractive to a wider class of applications and markets.

Evolution of GPUs. In what follows, we discuss why and how GPUs evolved from a rigid hardware, which was made to only process graphics-related workloads into a computing unit capable of processing a wider class of applications.

Fixed-function pipeline era. Before the 2000s, GPUs were a fixed-function microprocessor, solely responsible for processing graphics. They had a fixed graphics pipeline for performing 2D and 3D transformations, and computing lighting equations (Blythe, 2008; Lindholm et al., 2008). The graphics pipeline includes vertex,[103] primitive,[104] fragment,[105] and pixel[106] generation and processing units. These operations are highly parallel since the vertices, primitives, fragments, and pixels are independent, and can be processed in parallel during each stage of the graphics pipeline (Blythe, 2008). While this hardware could support basic graphics processing tasks, it did not have general-purpose computing capability.

Table 4. Estimated silicon die area (mm^2) for AMD Zen architecture based on silicon die image analysis[a]. Silicon dedicated to arithmetic operations constitutes less than 5% of the total die area, whereas 95% of the silicon die is used to improve efficiency and manage data-movement.

Architecture	Zen 1	Zen 2	Zen 3	Zen 4
Product Segment	Ryzen 7	Ryzen 9	Ryzen 9	Ryzen 9
Product Model	1800X	3950X	5950X	7950X
Number of Cores	8	16	16	16
Year	2017	2019	2020	2022
Total Die Area	212	277	293	267
I/O Die (IOD) Area[b]	212[c]	125	125	125
(%)		45.13%	42.66%	46.82%
Core Die (CCD) Area[d]	212[c]	2 × 76	2 × 84[e]	2 × 71[e]
(%)		54.87%	57.34%	53.18%
Aggr. Core Complex (CCX)	2 × 44.11	4 × 31.39	2 × 84	2 × 71
(%)	41.61%	45.33%	57.34%	53.18%
L3 Cache for all CCXs	2 × 16.32	4 × 16.82	2 × 35.52	2 × 25.1
(%)	15.40%	24.29%	24.25%	18.80%
L2 Cache for all Cores	8 × 1.65	16 × 0.81	16 × 0.77	16 × 1.03
(%)	6.23%	4.68%	4.20%	6.17%
Aggr. Core without L2	8 × 5.29	16 × 2.83	16 × 3.09	16 × 2.66
(%)	19.96%	16.35%	16.87%	15.94%
L1 instruction cache	8 × 0.6	16 × 0.13	16 × 0.08	16 × 0.09
(%)	2.26%	0.75%	0.44%	0.54%
L1 data cache	8 × 0.64	16 × 0.23	16 × 0.23	16 × 0.15
(%)	2.42%	1.33%	1.26%	0.90%
Fetch & Decode	8 × 0.88	16 × 0.47	16 × 0.48	16 × 0.45
(%)	3.32%	2.71%	2.62%	2.70%
Branch Prediction	8 × 0.72	16 × 0.48	16 × 0.44	16 × 0.47
(%)	2.72%	2.77%	2.40%	2.82%
Out-of-Order Scheduler[f]	8 × 0.79	16 × 0.38	16 × 0.35	16 × 0.29
(%)	2.98%	2.19%	1.91%	1.74%
Integer Exec. Unit	8 × 0.27	16 × 0.14	16 × 0.19	16 × 0.12
(%)	1.02%	0.81%	1.04%	0.72%
Float. Point Exec. Unit[g]	8 × 0.53	16 × 0.40	16 × 0.40	16 × 0.35
(%)	2.00%	2.31%	2.18%	2.10%
Load Store Unit	8 × 0.64	16 × 0.41	16 × 0.52	16 × 0.41
(%)	2.42%	2.37%	2.84%	2.46%

[a]Silicon die images are obtained from Fritz (2019a, 2019b), Fritz (2020), and Killian (2023) for Zen 1, Zen 2, Zen 3, and Zen 4, respectively. Analysis of the die images is based on Locuza (2020) for Zen 1 and Zen 2, and Locuza (2022) for Zen 3 and Zen 4. Percentage is based on the total area of all dies within a package.
[b]I/O Die (IOD) (Suggs et al., 2020) contains the InfinityFabric interface for inter-die communication, PCI Express interface, Memory Controller, Memory Interface, and Integrated GPU (for Zen 4 only). It is manufactured by using 12 nm (Zen 2 (Naffziger et al., 2021) and Zen 3 (Burd et al., 2022)) and 6 nm (Zen 4 (Munger et al., 2023)) process nodes.
[c]Zen 1 does not have a stand-alone I/O die. It has the Zeppelin Multi-Chip Module (Beck et al., 2018; Burd et al., 2019), where Core and I/O are implemented as one die.
[d]Core Complex Die (CCD) contains the Core Complex (CCX), L3 Cache, and other functional units. It is manufactured by using 14 nm (Zen 1 (Naffziger et al., 2021)), 7 nm (Zen 2 (Suggs et al., 2020) and Zen 3 (Evers et al., 2022)), and 5 nm (Zen 4 (Munger et al., 2023)) process nodes.
[e]Zen 3 and Zen 4 feature a unified CCD, where a single CCD has a single 8-core CCX (Burd et al., 2022; Munger et al., 2023) (as opposed to two 4-core CCX in Zen 1 and Zen 2 (Naffziger et al., 2021; Suggs et al., 2020)), allowing all eight cores to share L3 cache.
[f]Out-of-Order scheduler includes both integer and floating-point.
[g]Floating-point execution unit includes the SIMD units (i.e., SSE/AVX), but excludes floating-point registers.

Programmable shaders. In the early 2000s, due to the need for creating more complex computer-generated imagery, GPUs became increasingly programmable (Blythe, 2008; Elliott, 2004). Programmability was initiated with the introduction of programmable shaders (Peddie, 2023a), through the graphics-focused application programming interfaces (APIs), such as Direct3D by Microsoft and OpenGL by Khronos Group. More parts of the graphics pipeline became programmable as GPUs and graphics APIs advanced. For instance, Direct3D version 9 (2002) features

programmable vertex and fragment processing (Rodriguez et al., 2011), which was implemented by dividing the GPU hardware into several hardware pipeline stages (Goodnight et al., 2005) that resemble the programmable graphics pipeline (Angel and Shreiner, 2011; Owens et al., 2007). It was difficult for hardware designers to determine the area of the silicon die that should have been dedicated to each hardware pipeline stage across a wide range of graphics applications, as well as for software designers to identify bottlenecks in graphics applications in order to balance the performance of each graphics pipeline stage across different GPU architectures[107] (Chen et al., 2005; Peddie, 2023a). Moreover, the graphics APIs were also constantly changing, and, at times, the number of graphics pipeline stages was not known during the design of the hardware.

Unified shaders. A unified graphics architecture was introduced in 2006 to overcome the difficulties of balancing the hardware resources across different types of shaders (Peddie, 2023a). A unified hardware structure within a GPU, referred to as a Streaming Multiprocessor (SM) in NVIDIA GPUs, a Compute Unit (CU) in AMD GPUs, or a Core (XC) in Intel GPUs, can run all vertex, fragment, and geometry tasks, without distinction, depending on the program (i.e., GPU kernel[108]) that was given to it. The introduction of computing libraries (e.g., NVIDIA CUDA (Buck, 2007), AMD ROCm (Sun et al., 2018), Intel OneAPI (Aktemur et al., 2020)) made developing non-graphics applications to take advantage of GPUs easier, which marked the birth of general-purpose graphics processing units (GP-GPUs) (Figure 2). While we employ the terminology that is used by NVIDIA GPUs, the concepts generally apply to AMD and Intel GPUs as well. Table 5 lists the terminology equivalency between NVIDIA, AMD, and Intel GPUs.

General-purpose graphics processing units (GP-GPUs). Tables 6 and 7 summarize the evolution of NVIDIA's datacenter general-purpose GPUs. It started with the introduction of the Tesla architecture (2007) (Lindholm et al., 2008), a ground-breaking architecture that became the foundation of the NVIDIA GPU architecture for about two decades. The trend depicted in Tables 6 and 7 is similar to that shown in Figure 5 for general-purpose microprocessors, especially for transistor count, typical power, and number of cores. The base clock of the GPU shows only a small increase to manage thermal dissipation. Starting with the Kepler architecture (2013), NVIDIA introduced the GPU Boost feature, which can increase the GPU clock frequency for a short time, as long as it stays within the thermal and power envelope. From a manufacturing perspective, the

Table 5. Terminology equivalency between NVIDIA, AMD, and Intel GPUs.

NVIDIA	AMD	Intel	Description
Software Perspective			
Thread	Work-item	Work-item	Single stream of instruction and data
Warp	Wavefront	Sub-group	Group of threads that execute the same instruction stream in lock-step fashion and operate on different data
Thread Block	Work-group	Work-group	Group of warps executed by single SM/CU/XC and share synchronization barrier and on-chip memory
Grid	ND-Range	ND-Range	Collection of thread-block or work-group of a kernel executed by GPU.
Hardware Perspective			
CUDA Cores (CC)	Stream Processors (SP)	Vector Engines (XVE)	Arithmetic unit that processes one data item and executes portion of SIMT instruction stream
Tensor Cores (TC)	Matrix Cores	Matrix Engines (XMX)	Specialized unit to accelerate matrix-matrix operations (e.g., GEMM)
Subpartition (SMSP)	SIMD Unit	Execution Unit (EU)	SIMT processor to execute warp/wavefront/sub-group in a lock-step manner
Streaming Multi-processor (SM)	Compute Unit (CU)	Xe Core (XC)	Unit capable of executing one thread-block/work-group of a kernel. It consists of subpartitions sharing on-chip memory
Texture Processing Cluster (TPC)	Workgroup Processor (WGP)	Render Slice or Compute Slice (SLC)	GPU super-clusters, consisting of several SMs/CUs/XCs to perform texture operations
Graphics Processing Cluster (GPC)	Shader Engine		GPU mega-clusters, consisting of several super-clusters to perform graphics operations
Graphics Processing Die (GPD)[a]	Graphics Complex Die (GCD)	Stack (STK)	GPU dies, consisting of several megaclusters; used for chiplet-based GPU.

[a]Unofficial name, given for completeness.

Table 6. Evolution of NVIDIA datacenter-class general-purpose GPUs.

Year	2007	2009	2011	2013	2015	2016
Name						
Architecture	Tesla	Tesla 2	Fermi	Kepler	Maxwell	Pascal
Product	C870	C1080	M2090	K40[a]	M40[a]	P100
Manufacturing						
Process node[b] (nm)	90	55	40	28	28	16
Transistor count[b]	681 M	1400 M	3000 M	7080 M	8000 M	15.3 B
Die area[b] (mm^2)	484	470	520	561	601	610
Compute						
Base clock (MHz)[b]	600	610	651	745	948	1328
Boost clock (MHz)[b]	–	–	–	876	1112	1480
Total GPDs	1	1	1	1	1	1
Total GPCs	1	1	4	5	6	6
TPC per GPC	8	10	4	3	4	5 or 4[c]
Total TPCs	8	10	16	15	24	28
SM per TPC	2	3	1	1	2	2
Total SMs	16	30	16	15	24	56
CUDA Cores per SM	8	8	32	192	128	64
Total CUDA Cores	128	240	512	2880	3072	3584
Vector FP64 (TFLOP/s)[b]	–	0.08	0.67	1.68	0.21	5.3
Vector FP32 (TFLOP/s)[b]	0.35	0.62	1.32	5.05	6.83	10.6
Vector FP16 (TFLOP/s)[b]	–	–	–	–	–	21.2
On-chip Memory						
Register/SM (kB)	32	64	128	256	256	256
L1 Cache/SM (kB)	–	–	64[d]	64[e]	48[f]	24[g]
Shared/SM (kB)	16	16			96[f]	64[g]
L2 Cache (kB)	–	–	768	1536	3072	4096
Off-chip Memory						
Size (GB)	1.5	4	6	12	12	16
Technology	GDDR3	GDDR3	GDDR5	GDDR5	GDDR5	HBM2
Interface (bit)	384	512	384	384	384	4096
Clock (MHz)[b]	800	800	924	1502	1502	715
Bandwidth (GB/s)[b]	76.8	102	177	288	288	732
Physical Properties						
Form Factor	PCIe	PCIe	PCIe	PCIe	PCIe	SXM2
TDP[b,h] (Watts)	171	188	250	245	250	300

[a]NVIDIA Tesla K40 and NVIDIA Tesla M40 were manufactured using the same 28 nm process node, although with different architectures. The M40 with Maxwell architecture has a higher single-precision performance but much lower double-precision performance, compared to K40 with Kepler architecture. NVIDIA marketed the Tesla M40 as a deep learning training accelerator, where most deep learning applications only need single precision.
[b]Data is obtained from TechPowerUp GPU Specs Database: C870, C1080, M2090, K40, M40, P100.
[c]To improve manufacturing yields, some GPCs can have all of their TPCs active, while others can have fewer TPCs enabled. For example, the H100 can have two GPCs with 9 TPCs per GPC and 6 GPCs with 8 TPCs/GPC.
[d]The GF110 chip in NVIDIA Tesla M2090 features a unified L1 cache and shared memory of 64 KB per SM, which can be configured as 16 KB:48 KB or 48 KB:16 KB (L1-cache:shared-memory).
[e]Just like its predecessor, the GK180 chip in NVIDIA Tesla K40 features 64 KB of unified L1 cache and shared memory per SM. It can be configured as 16 KB:48 KB, 48 KB:16 KB, or a split 32 KB:32 KB (L1-cache:shared-memory).
[f]The GM200 chip in NVIDIA Tesla M40 has a dedicated 96 KB shared memory and 48 KB of unified L1 cache and texture cache. In its predecessors, the texture cache was a dedicated unit separate from L1 (e.g, Kepler has a 48 KB texture cache and 64 KB unified L1 cache and shared memory).
[g]The GP100 chip in NVIDIA Tesla P100 has the same implementation of on-chip memory as its predecessor. Although GP100 has reduced L1 cache and shared memory per SM, due to the half number of CUDA cores in each SM compared to GM200, the total L1 cache and shared memory in P100 are 42% larger than M40 because of the double number of SMs compared to M40.
[h]Thermal design power (TDP) specifies the maximum amount of heat that can be generated by a chip, which a cooling system must handle (Ganapathy and Warner, 2008; Guermouche and Orgerie, 2022). Since heat is a consequence of dissipated power, TDP indicates the power a chip consumes during a sustained, long operation. A chip may consume higher power than TDP for a short period of time (e.g., due to the boost clock) as long as it is still within its thermal and power envelopes.

Table 7. Evolution of NVIDIA datacenter-class general-purpose GPUs-continued.

Year	2018	2018	2020	2022	2022/**2024**	2024/**2025**
Name						
Architecture	Volta	Turing	Ampere	Ada Lovelace	Hopper	Blackwell
Product	V100	Quadro RTX 6000	A100	RTX 6000 Ada Gen	H100/H200	B200/B300
Manufacturing						
Process node[a] (nm)	12	12	7	4	4	4
Transistor count[a]	21.1 B	18.6 B	54.2 B	76.3 B	80 B	2 × 104 B
Die area[a] (mm^2)	815	754	826	609	814	2×∼820
Compute						
Base clock (MHz)[a]	1290	1440	1275	915	1590	1665
Boost clock (MHz)[a]	1530	1770	1410	2505	1980	1837
Total GPDs	1	1	1	1	1	2
Total GPCs	6	6	7	12	8	16
TPC per GPC	7 or 6	6	8 or 7	6	9 or 8	9 or 8
Total TPCs	40	36	54	71	66	132
SM per TPC	2	2	2	2	2	2
Total SMs	80	72	108	142	132	264
CUDA Cores per SM	64	64	64	128	128	128
Total CUDA Cores	5120	4608	6912	18176	16896	33792
Vector FP64 (TFLOP/s)[a]	7.8	0.51	9.7	1.42	33.5	62.1
Vector FP32 (TFLOP/s)[a]	15.7	16.3	19.5	91.1	67	124.1
Vector FP16 (TFLOP/s)[a]	31	32.6	78	91.1	134	248.2
Tensor Cores[b] per SM	8	8	4	4	4	4
Total Tensor Cores	640	576	432	568	528	1056
Tensor FP64 (TFLOP/s)[a]	–	–	19.5	-	67	37/1.2
Tensor TF32 (TFLOP/s)[a]	–	–	156	182	495	1100
Tensor FP16 (TFLOP/s)[a]	125	130.5	312	364	989	2250
Tensor INT8 (TOP/s)[a]	–	261	624	728	1979	4500/0.14
On-chip Memory						
Register/SM (kB)	256	256	256	256	256	256
L1 Cache/SM (kB)	128[c]	96[d]	192[e]	128	256[f]	256[g]
Shared/SM (kB)						
L2 Cache (kB)	6144	6144	40960	98304	51200	102400
Off-chip Memory						
Size (GB)	32	24	80	48	80/141	192/288
Technology	HBM2	GDDR6	HBM2E	GDDR6	HBM3[h]/3E[h]	HBM3E[h]
Interface (bit)	4096	384	5120	384	5120/6016	8192
Clock (MHz)[a]	877	1750	1593	2500	1310/1600	∼2000
Bandwidth (GB/s)[a]	900	672	2039	960	3353/4812	∼8000
Physical Properties						
Form Factor	SXM2	PCIe	SXM4	PCIe	SXM5	SXM6
TDP[a] (Watts)	300	260	500	300	700	1000/1200

[a]Data is obtained from TechPowerUp GPU Specs Database: V100, Quadro RTX 6000, A100, RTX 6000 Ada Generation, H100, H200, and B200.
[b]The Tensor Cores are specialized compute blocks optimized for tensor operations. They were first introduced with the V100 GPU.
[c]The GV100 chip in NVIDIA V100 features a unified L1 cache, texture cache, and shared memory, for a total of 128 KB per SM. The shared memory can be configured to use up to 96 KB of the unified memory, while the rest is used for both the L1 and texture cache.
[d]The TU102 chip in NVIDIA Quadro RTX 6000 features a 96 KB L1 data cache/shared memory. Traditional graphics workloads can partition it as 64 KB of dedicated graphics shader RAM and 32 KB for texture cache and register file spill area. Compute workloads can divide it into 32 KB shared memory and 64 KB L1 cache, or 64 KB shared memory and 32 KB L1 cache.
[e]The GA100 chip in NVIDIA A100 features a unified L1 cache, texture cache, and shared memory, for a total of 192 KB per SM. The shared memory can be configured to use up to 164 KB of the unified memory, while the rest is used for both the L1 and texture cache.
[f]The GH100 chip in NVIDIA H100 features a unified L1 cache, texture cache, and shared memory, for a total of 256 KB per SM. The shared memory can be configured to use up to 228 KB of the unified memory, while the rest is used for both L1 and texture cache. The preliminary specification of H200 retains the same compute configuration with increased memory bandwidth and capacity, which is useful for handling large language models (LLMs).
[g]Blackwell is the first chiplet-based NVIDIA GPU consisting of two dies connected through 10 TB/s NVIDIA High Bandwidth Interface (NV-HBI). Its preliminary specification is based on publicly available information. The Blackwell Ultra B300 is fine-tuned specifically to accelerate large language models by adding more memory and a higher power envelope than the Blackwell B200, resulting in 50% higher Tensor FP4 performance at the cost of lower performance on other precisions.
[h]HBM3 and HBM3E can double the effective bandwidth compared to HBM2E at roughly the same memory clock by utilizing higher number of memory channels to increase parallelism (i.e., eight 128-bit channels in HBM2E and sixteen 64-bit channels in HBM3); See Part II Section 2.2.6 (Hanindhito et al., 2026).

advancement of process nodes played an important role in the evolution of GPUs. The number of transistors is doubled every 2 years, except from 2013 to 2015, due to process node stagnancy. As it becomes more difficult to shrink the transistor size further, the die area has grown slowly,[109] which then lowers the yield, and increases manufacturing costs (Arunkumar et al., 2017; Mack, 2015; Sun et al., 2020). Just like general-purpose microprocessors, future GPUs will use advanced packaging technologies, such as chiplets[110] (Loh et al., 2023; Pratheek et al., 2022) to improve yield and reduce manufacturing costs (Section 2.10).

GPU compute unit. GPUs are popular for running and accelerating non-graphics applications that exhibit significant parallelism, such as high-performance computing, and machine learning. Abundant parallelism in these applications can be extracted through hundreds of GPU cores (e.g., CUDA Cores in NVIDIA GPUs, Stream Processors (SP) in AMD GPUs, and Vector Engines (XVE) in Intel GPUs) with single-instruction multiple-threads (SIMT) execution model (Section 2.11). While the design philosophy of GPU is similar to that of many-core microprocessors, the term "core" in a GPU is significantly different than that in many-core and general-purpose microprocessors. In both general-purpose and many-core microprocessors, each individual core has a front-end to fetch and decode instructions, a back-end with multiple functional units to execute the instructions, and dedicated registers and first-level cache to store recent and commonly used data. Each core can run software threads independently, until they arrive at a synchronization barrier, if any.

On the other hand, GPU cores are much simpler. They are grouped into SMs/CUs/XCs, where each core shares commonly-used hardware structures, such as instruction schedulers, L1 cache, and register files, roughly leaving each core only to contain arithmetic-logical units (e.g., floating-point or integer units). As seen in Table 4, these commonly-used hardware structures typically occupy more than 95% of the silicon die area in general-purpose microprocessors, and thus, sharing them across hundreds of cores makes it possible for GPUs to have thousands of (simple) cores, while keeping the power consumption and silicon die area under control. Figure 7 illustrates terminologies that are used in GPU hardware and software.

While GPUs can execute thousands of threads, they struggle to handle irregular execution patterns. A group of threads, called a warp, is executed by a single SIMT processor. Due to sharing a large portion of hardware structures, each thread within a warp is executed in a lock-step fashion; i.e., any difference in branch outcomes[111] between threads within a warp must be serialized, reducing GPU execution efficiency.

Tables 6 and 7 show the total number of CUDA cores has increased from generation to generation, as they are the ultimate workhorse to extract as much parallelism as possible from an application. The hierarchical organization has also changed between generations, either by changing the number of CUDA cores per SM, by adding more features to each SM, or by adding more SMs to a GPU. For the same total number of CUDA cores per GPU, it is generally preferred to have more SMs with less number of CUDA cores per SM. This allows for better resource[112]-sharing within an SM, and reduces the complexity of thread scheduling. In addition, each SM can run a different kernel, which is beneficial for applications that cannot utilize all the available SMs on a single GPU.

GPU memory system. A GPU's memory system consists of large on-chip and high-bandwidth off-chip memory, in order to keep up with data demands from its extremely large number of cores.

On-chip memory. A GPU's on-chip memory consists of a large number of registers (*Part II: Registers* (Hanindhito et al., 2026)) and various types of caches: texture cache, constant cache, and data caches (*Part II: Caches* (Hanindhito et al., 2026)). Moreover, GPUs provide user-managed, on-chip memory, which is called shared memory (*Part II: User-managed vs. compiler-managed scratchpad memory* (Hanindhito et al., 2026)).

The L1 cache started to appear on the Fermi (2011) architecture in addition to the shared memory. Both of them were implemented as a unified, on-chip memory per SM, where users have control over how the unified on-chip memory is divided between the L1 cache and the shared memory. This organization was changed in Maxwell (2015) and Pascal (2016) architectures, where the L1 cache and the texture cache were implemented as unified on-chip memory, whereas the shared memory was implemented as standalone (i.e., fixed-sized) on-chip memory. The organization of the on-chip memory changed again in the Volta (2018) architecture, where the L1 cache, texture cache, and shared memory were implemented as a unified, on-chip memory. Its successors, Ampere and Hopper still retain this organization. The L2 cache is shared across SMs, and provides a way for SMs to share data; it is directly connected to the off-chip memory controller.

Off-chip memory. Due to limited on-chip memory capacity, GPUs need to access data from off-chip memory. Accessing data from off-chip memory has high latency, which can stall execution, resulting in reduced execution efficiency of GPUs. GPU tries to hide the memory access latency by aggressive context switching (Lee and Wu, 2014; Lin et al., 2018): it schedules another thread block in case a thread block is stalled and is waiting for memory access. This keeps the SMs busy. On the other hand, to fulfill the off-chip bandwidth requirements, GPUs tend to use off-chip memory that has significantly higher bandwidth due to using wider memory interfaces (Kim et al., 2014; Li et al., 2018), and higher memory clock frequency[113] (Cho et al., 2012;

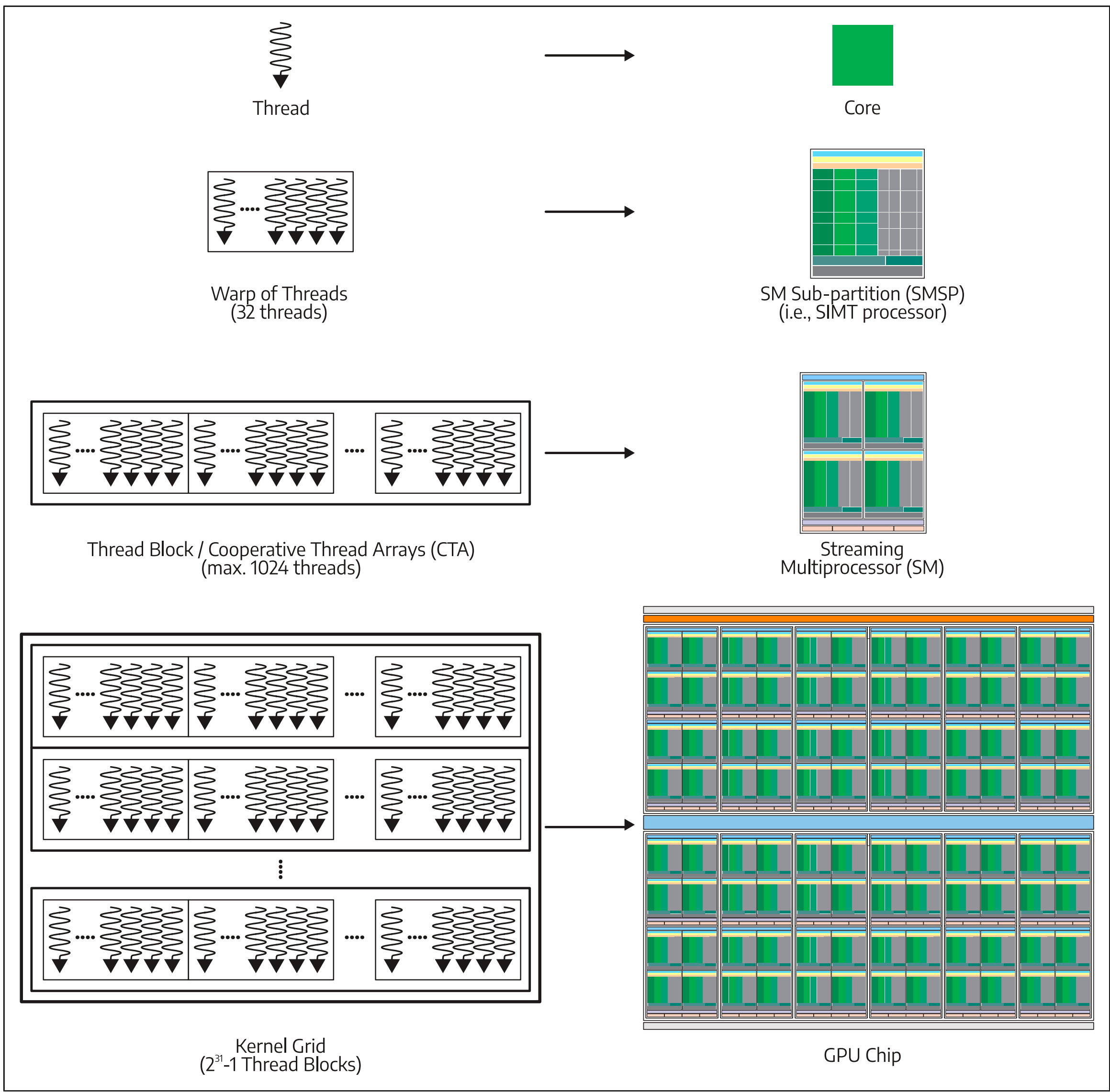

Figure 7. A GPU die floor plan (right) and execution model (left). CUDA Cores execute a thread and perform operations on data. A group of threads, called warp, is scheduled into SM Sub-partitions (SMSP), where they are executed in a lock-step manner. A thread block or cooperative-thread-arrays consists of up to 1024 threads (32 warps) and is mapped into a Streaming Multiprocessor (SM). A GPU kernel may use tens of SMs.

Zheng et al., 2008). This led to the development of a specialized version of such memory dedicated to graphics in 1997, which was called SGRAM (Glaskowsky, 1997; Hitachi, Ltd, 1997), and its successor, GDDR-SDRAM (Foss, 1997; Prince, 1999), which will be discussed in detail in *Part II: Graphics DRAM* (Hanindhito et al., 2026).

Even with these specialized memory systems, meeting the growth in bandwidth demand due to increased number of cores on a GPU is challenging (Hwu and Patel, 2018). In 2015, HBM found its way to the GPUs to fulfill the needs of even higher memory bandwidth (Hu et al., 2018; Macri, 2015). Since then, HBM and its successors have been integrated into GPUs, especially data-center-class GPUs (Tables 6 and 7). HBM and its competing product, HMC, are discussed in *Part II: High-Bandwidth Memory (HBM) and variants* (Hanindhito et al., 2026).

Specializations for machine learning. Machine-learning applications have become very popular during the last decade. Many of these applications involve convolution, which translates into matrix operations. Machine-learning's large market has motivated GPU designers to add more specialized features to improve efficiency across popular applications.

General matrix-matrix multiplication (GEMM) operations are abundant in machine learning and few high

performance computing applications (Juan et al., 2021; Zhuang et al., 2023b). Starting from the NVIDIA Volta architecture (2018), GPUs have been equipped with Tensor Cores, which are specialized cores for performing GEMM with different levels of precision. The first-generation of Tensor Cores were able to perform half-precision GEMM operations to support mixed-precision machine learning training (Micikevicius et al., 2018) with the theoretical peak performance of 125 TFLOP/s, which is roughly four times the peak performance provided by CUDA Cores for half-precision computation within the same chip (Markidis et al., 2018). Therefore, GEMM operations can be performed more efficiently through (specialized) Tensor Cores, as opposed to (more general) CUDA cores. This makes GPUs a popular hardware accelerator for many machine-learning applications (Cass, 2020; Jordà et al., 2019).

Subsequent versions of Tensor Cores added more levels of precision and operations. Second generation Tensor Cores in NVIDIA Turing architecture (2019) added 8-bit, 4-bit, and 1-bit integer data-types to support quantized machine learning inference (Kim et al., 2021; Li et al., 2021) and binary neural networks (Li and Su, 2021). Third generation Tensor Cores in NVIDIA Ampere architecture (2020) added new BFloat16 (Wang and Kanwar, 2019) and TensorFloat32 (Choquette et al., 2021) to accelerate machine learning workloads with better dynamic ranges. They also added new FP64 support, which opens the possibility of Tensor Cores being used in HPC and scientific computing applications (Gallet and Gowanlock, 2022; Lee et al., 2022). Support for sparse matrix-matrix multiplication for applications that have abundant sparsity was also added, which improves performance and reduces bandwidth utilization (Anzt et al., 2020; Sun et al., 2022b). Fourth generation Tensor Cores in NVIDIA Hopper architecture (2022) added new quarter precision, FP8, with a built-in transformer engine that can dynamically choose the appropriate FP8 format to improve performance, while maintaining model accuracy for large language models (Zhuang et al., 2023a). The new Tensor Memory Accelerators (TMA) improve asynchronous data staging for the Tensor Cores. Both AMD and Intel followed NVIDIA to integrate their own version of Tensor Cores in their GPUs: AMD introduced Matrix Core with their CDNA GPU architecture in 2020, and Intel introduced XMX Matrix Engine into their Ponte Vecchio GPU architecture in 2022 (Jiang, 2022).

Programmability. NVIDIA and AMD provide CUDA (Buck, 2007) and ROCm (Sun et al., 2018), respectively, for programming GPUs. High-level software stacks, such as TensorFlow (Abadi et al., 2015) and PyTorch (Paszke et al., 2019), have been specifically developed for machine learning; these frameworks provide intuitive and practical Python APIs, which efficiently map user-defined network models to the underlying GPU architecture, and therefore, maximize productivity.

Despite the existence of these frameworks, developing efficient GPU code for a new application is challenging, and requires extensive understanding of the GPU architecture. For instance, developers must be aware of performance degradation that is caused by several factors, such as: a) thread divergence, which happens when threads in a warp follow different paths in conditional statements; b) register spilling (*Part II: Registers* (Hanindhito et al., 2026)), which can lead to significant delays and underutilization of the compute units; c) high-kernel-launch overhead, which occurs when kernels are not assigned enough work; and d) lack of vectorization, i.e., not taking advantage of the SIMD capabilities of the vector units.

Specialized and custom hardware

Specialized hardware can be used when desired performance or efficiency metrics cannot be met by using CPUs and GPUs. In essence, hardware accelerators epitomize the art of balancing flexibility (e.g., range of supported applications) and efficiency (e.g., cost, performance, energy consumption) (Sze et al., 2017; Verbauwhede et al., 2004), as illustrated in Figure 2, and discussed in *Execution model, architecture, and implementation style*. We outline typical challenges that arise during the development of specialized hardware, followed by possibilities for implementing the specialized accelerator. We end this section by providing several examples of specialized hardware that have been used in machine learning and scientific computing.

Challenges in developing specialized hardware accelerators. The development and adoption of specialized accelerators face two major challenges: cost and software support, which we highlight next.

Development and manufacturing of an accelerator is expensive, compared to using off-the-shelf products. The total cost of ownership (TCO) (Cui et al., 2017; Martens et al., 2012) of the off-the-self products, compared to a specialized accelerator, is often used as the key metric for making strategic decisions (Khazraee et al., 2017; Magaki et al., 2016). For large-scale[114] deployments, the cost per chip of a specialized accelerator may be lower than that of a corresponding off-the-shelf hardware (Wu and Tsai, 2004; Zahiri, 2003). By contrast, if only a small number of units is to be expected, the baseline cost of designing an accelerator may not be amortized across a large enough number of fabricated instances. This implies development of specialized accelerators is economically viable only if many of them are going to be produced. Moreover, specialized accelerators often have longer development and implementation cycles compared to off-the-shelf components. A common way of reducing the cost of an accelerator is for it to target a broad class of applications within the same domain, instead of a single application. This is often referred to as domain-specific architecture (DSA) (Dally et al., 2020; Fujiki et al., 2021; Halawani and Mohammad, 2024;

Krishnakumar et al., 2023). We provide several examples of DSAs.

Furthermore, availability of robust software ecosystems[115] that help users map and migrate their existing applications to target specific accelerators (Cong et al., 2016; Koeplinger et al., 2018; Koul et al., 2023) is critical. Without adequate software support, users will face difficulties in migrating their existing workloads to accelerators, limiting their adoption rates (Cascaval et al., 2010; Hawick and Playne, 2014; Ikarashi et al., 2022).

Implementation choices. There are several options for implementing specialized hardware accelerators. The fabric in which a chip architecture can be implemented is discussed in *Execution model, architecture, and implementation style*. Next, we highlight key characteristics of popular fabrics, such as FPGAs, CGRAs, or implementation as full-custom ASICs.

Field Programmable Gate Arrays (FPGAs) have been used as the substrate to implement many hardware accelerators, including machine learning (Chen et al., 2019b; Roorda et al., 2022), linear algebra (Hu et al., 2021; Matteis et al., 2020), graph processing (Besta et al., 2019; Zhou et al., 2019), cryptography (Chelton and Benaissa, 2008; Chen et al., 2019a; Kumar et al., 2020), data analytics (Hoozemans et al., 2021; Kara et al., 2020), and high-performance scientific computing (Belletti et al., 2009; Tan et al., 2014), among many others (Gandhare and Karthikeyan, 2019; Shahzad et al., 2021). Aside from programmable logic elements, FPGAs can contain memory cells (on-chip static random access memory (SRAM); see *Part II: Static random access memory* (Hanindhito et al., 2026), and specialized blocks (e.g., digital signal processing (DSP) blocks) (Langhammer and Pasca, 2015; Ronak and Fahmy, 2016)). Recently-released FPGAs are also equipped with High-Bandwidth Memory (HBM) (*Part II: High-Bandwidth Memory (HBM) and variants* (Hanindhito et al., 2026)) to tackle the bandwidth demands of many applications (Holzinger et al., 2021; Wang et al., 2020).

Intel and AMD, the leading FPGA vendors,[116] have recently developed FPGA architectures enhanced to more effectively support the high computational demands in ML (Boutros et al., 2024). In particular, Intel employs specialized in-fabric processing blocks, known as tensor blocks, comprising multiple dot-product engines (Gribok and Pasca, 2024; Langhammer et al., 2021). On the other hand, AMD introduced the AI engine (AIE) (Ahmad et al., 2019), which is an out-of-fabric array of programmable vector processors, integrated next to the FPGA fabric. Another notable AI-optimized FPGA is Achronix's Speedster7t, which contains up to 2560 Machine Learning Processor blocks (Cairncross et al., 2023). These devices have been used in many works to accelerate AI workloads, significantly outperforming GPUs and traditional FPGAs (Boutros et al., 2020; Taka et al., 2023, 2024). Moreover, academic researchers have proposed in-fabric blocks that employ a 2D systolic dataflow (i.e., FPGA-version of Tensor Cores; see Section 4.1.4) (Arora et al., 2021; Taka et al., 2025), or even processing-in-memory (*Part II: Near-memory processing (NMP) and processing-in-memory (PIM)* (Hanindhito et al., 2026)) capabilities (Arora et al., 2022), to provide better performance for popular applications. As a result, both Intel and AMD see opportunity in integrating FPGAs into their general-purpose microprocessors. This gives customers flexibility to implement accelerators for specific use cases, especially in data-center and cloud computing workloads.

The hardware designs targeted for implementation on FPGAs are written using low-level hardware description languages (HDL), such as Verilog, SystemVerilog and VHDL. Compared to commonly-used programming languages for CPUs and GPUs, these languages are much more complicated. This leads to longer development and debugging cycles. Major challenges in implementing hardware design on FPGAs include the routing congestion, and maximizing the attainable clock frequency. AMD and Intel provide High-Level-Synthesis (HLS) tools, namely Vitis HLS, and Intel HLS, respectively, which are C/C++ libraries that convert high-level software code to HDL. While HLS tools can greatly boost productivity, they often come with severe limitations that affect design choices and attainable performance.

Coarse Grained Reconfigurable Arrays (CGRAs) are used in many domains, including signal processing (Mei et al., 2008; Park et al., 2009), high-performance scientific computing (Charitopoulos et al., 2021; Käsgen et al., 2018), machine learning (Geng et al., 2020; Wei et al., 2023), and near-data processing (Gao and Kozyrakis, 2016). As highlighted in *Execution model, architecture, and implementation style*, their main advantage over FPGAs are faster reconfiguration time, as well as increased performance and energy efficiency, which bring them closer to ASICs. Mapping applications to CGRAs is a challenging and heavily-studied area. Improved tools, compilers, and frameworks over the years are expected to make it easier for users to adopt CGRAs for their applications (Chin et al., 2017; Martin, 2022; Wijerathne et al., 2022).

Finally, Application-Specific Integrated Circuits (ASICs) provide higher performance and energy efficiencies compared to FPGAs and CGRAs. ASICs have a higher design and manufacturing cost, and longer development time (Wu and Tsai, 2004; Zahiri, 2003). ASIC is a better choice for implementing a mature accelerator architecture that targets popular workloads.[117] Due to the high cost of developing ASICs, most ASICs sacrifice some level of efficiency for programmability, in order to support several similar applications for a given domain.[118]

Examples of custom and specialized hardware. In this part, we provide several examples of popular and publicly-known hardware accelerators. Many of these accelerators are targeted for processing of deep neural networks and are often

referred to as Neural Processing Units (NPUs). The list is not exhaustive: our intention is to highlight various possibilities, and show readers that making specialized hardware is becoming more common. Table 8 summarizes key specifications of these devices, along with other popular chips.

Google's tensor processing unit (TPU). TPU is a machine-learning accelerator DSA that was developed by Google, and was first introduced in 2016 (Jouppi et al., 2017, 2018). The main advantage of TPUs over GPUs is that they are about an order-of-magnitude more energy-efficient. Due to the scale of Google's operations, these energy savings could be considerable, reducing their total cost of ownership (TCO).

TPUs are programmable through TensorFlow (Abadi et al., 2015), and are available for public use through the Google Cloud Platform (GCP).

Its first generation (TPUv1) was primarily used for inference and was manufactured by using 28 nm process node technology. It had matrix-multiply-accumulate units arranged in a systolic array fashion (Jouppi et al., 2018), which is capable of multiplying 8-bit integers and accumulating the results in 32-bit. Even with its specialized architecture, the matrix-multiply-accumulate unit – the arithmetic unit that performs the sought-after computations – is only 24% of the whole silicon die area, compared to less than 5% in general-purpose microprocessors (Table 4).

The second generation of TPU (TPUv2), introduced in 2017, used 16 nm process node technology. It added training support and included High-Bandwidth Memory (HBM) to address the memory bottleneck problems of its predecessor (Jouppi et al., 2020).

The third generation of TPU (TPUv3), introduced in 2018, used 16 nm process node technology. Compared to its predecessor, it has twice the number of matrix-multiply-accumulate units, twice the HBM capacity, higher clock frequency, and higher memory bandwidth (Jouppi et al., 2021).

The fifth-generation of TPU (TPUv4[119]), introduced in 2021, uses 7 nm process node technology. It includes optical switches to allow reconfiguration of inter-chip interconnection topology. It supports 8-bit integers, in addition to BFloat16, and has hardware support for large language models and recommender systems (Jouppi et al., 2023).

The latest generation (TPUv5p) was introduced in 2023, which is their most powerful chip so far. While its process node technology has not been disclosed yet, it provides a significantly higher memory bandwidth of 2.76 TB/s, with a peak 918 TOPS/s INT8 performance, targeted for more efficient training and inference of large language models.

GraphCore's intelligence processing unit (IPU). IPU[120] is a machine-learning DSA, developed by GraphCore. The fundamental architecture of IPU is different than that of CPUs and GPUs. Contrary to the SIMD in the vector units[121] (Hassaballah et al., 2008; Raman et al., 2000) of CPUs, or SIMT of GPUs (Fung and Aamodt, 2011; Habermaier and Knapp, 2012), Graphcore IPU uses MIMD execution model (Berg and Siegel, 1991; Flynn and Rudd, 1996), which allows it to efficiently execute massive number of threads that have distinct codes, execution flows, and irregular or sparse data-access (Section 2.11) (Jia et al., 2019). Unlike GPUs and TPUs, which use High Bandwidth Memory (HBM) to provide adequate off-chip bandwidth

Table 8. Specifications of popular server-class custom ASICs.

	Google	Google	Amazon	Meta	Intel	Cerebras	Graphcore	Sambanova
	TPUv4	TPUv5p	Inferentia2	MTIA	Gaudi 3	WSE-2	MK2 IPU	SN40L RDU
Process Node (nm)	7	–[a]	–[a]	7	5	7	7	5
Peak FP8/INT8 (TOPs/s)	275	918	380	105.6	1835	7500[b]	560[c]	638[d]
DRAM Technology	HBM2	HBM2e	HBM2e	LPDDR5	HBM2e	DDR/Flash[e]	DDR[f]	HBM/DDR[g]
DRAM Capacity (GB)	32	95	32	64	128	1000/1500[e]	448[f]	64/768[g]
DRAM Bandwidth (GB/s)	1200	2765	820	176	3700	150[e]	–[a]	1600/100[g]
On-Chip SRAM size (MB)	160	–[a]	–	136	96	40000	900	520
TDP (W)	170	–[a]	–[a]	25	900	23000	300[h]	625[i]

[a]Information not yet publicly available.
[b]This corresponds to FP16 dense peak performance, while sparse peak performance is at 75 PFLOPs (Lie, 2022).
[c]Peak FP8 throughput in a C600 PCIe card comprising a single IPU chip.
[d]Refers to peak throughput for BFloat16. This is slightly lower than the 688 TOPs/s peak of SN30L.
[e]Refers to the MemoryX memory unit, comprising 12 MemoryX nodes, primarily used to store model weights and stream them into WSE-2 for processing. Each MemoryX node contains 1 TB of DRAM and 0.5 TB of Flash memory.
[f]DRAM is shared between the host CPU and all 4 IPUs in an M2000 IPU-Machine (Knowles, 2021). In a C600 PCIe card comprising a single IPU, the IPU relies on the PCIe bus for communicating with the host's DRAM.
[g]The SN40L comprises 64 GB of HBM and 768 GB of DDR memory, each offering 1.6 TB/s and 100 GB/s peak bandwidth, respectively.
[h]Refers to the TDP of each individual IPU chip in an M2000 IPU-Machine (Knowles, 2021). The TDP of a C600 PCIe card is 185 W.
[i]Refers to typical inference power per SN40L RDU chip in a Cerulean system comprising 16 SN40L RDU chips.

and memory capacity, IPUs use a large number of on-chip static random access memory (SRAM) (*Part II: Static random access memory* (Hanindhito et al., 2026)) to provide ultra-high bandwidth memory, at the cost of consuming more power: the Mk1 has 304 MB of on-chip memory, while the Mk2 has 896 MB of on-chip memory,[122] providing 62 TB/s on-chip memory bandwidth. IPUs share the DRAM memory with the host CPU, which can be used to provide additional memory capacity to the IPUs (Knowles, 2021).

IPU's architecture (i.e., MIMD) allows it to efficiently handle massively irregular computations that exhibit irregular memory access patterns, compared to GPUs. Moreover, IPU's large, on-chip memory allows some machine learning models to fit inside the chip. Such models can exploit the significantly higher bandwidth and lower latency of the on-chip memory, compared to off-chip HBM in GPUs. A larger model can be partitioned across IPUs, and one-time used data can be streamed from the host CPU through the PCI Express interface.

Graphcore provides Graphcore Poplar software stack and development tools (Bohl, 2022), along with integration with popular machine-learning frameworks, such as TensorFlow (Abadi et al., 2015), and PyTorch (Paszke et al., 2019). Among the applications that target Graphcore IPUs are machine-learning (Bohl, 2022) (e.g., regression (Balewski et al., 2022), text detection (Sumeet et al., 2022), neuromorphic (Sun et al., 2022a)), particle physics (Maddrell-Mander et al., 2021), and cosmology (Arcelin, 2021).

The first generation IPU (Mk1), introduced in 2017 (Trader, 2017), was manufactured via 16 nm process node technology, and had 23 billion transistors. The second generation (Mk2), introduced in 2020, relied on 7 nm process node technology, and had nearly 60 billion transistors (Knowles, 2021).

Cerebras' Wafer scale engine (WSE). WSE is a machine learning accelerator DSA developed by Cerebras. The accelerator occupies the whole area of the silicon wafer,[123] measuring an area of 46,225 mm^2 (Lauterbach, 2021), which is 56 times larger than the die size of an NVIDIA H100 GPU (Table 7). Key reasons behind this radical design philosophy are: a) machine-learning models are becoming increasingly large, necessitating the use of large compute clusters; this requires the programmer to distribute the model by navigating through the complex system hierarchy,[124] which could be challenging; and b) the conventional centralized memory system is not optimized for many machine learning applications due to its high latency and limited bandwidth (Cerebras Systems, 2019). The WSE provides fast on-chip interconnection between cores and distributed on-chip static random access memory (SRAM) (*Part II: Static random access memory* (Hanindhito et al., 2026)), at the expense of consuming more energy, to address these two issues. It also comes with a software stack to map the machine-learning workloads across the cores in WSE (Lie, 2022). The chip has also been used for solving partial differential equations, and has shown impressive performance as long as the problem fits into a single WSE (Groeneveld et al., 2021; Lin et al., 2022; Luo et al., 2023).

Cerebras manages the yield problem by implementing spare cores as a redundancy measure to replace any defective cores (Lauterbach, 2021). Power delivery and heat problems are managed through vertical power delivery to supply 20,000 amperes of electrical currents, and large water-cooled copper heat exchangers, respectively (Lauterbach, 2021).

The first generation (WSE), introduced in 2019, relies on 16 nm process node technology. It features 400,000 cores, provides 18 GB of on-chip memory, and has 1.2 trillion transistors (Cerebras Systems, 2019). The second generation (WSE-2), introduced in 2021, uses 7 nm process node. It features 850,000 cores, along with 40 GB of on-chip memory, and uses 2.6 trillion transistors (Lauterbach, 2021; Lie, 2022). The WSE-2 system is also paired with an external memory device named MemoryX (Lie, 2024). MemoryX is physically decoupled from WSE-2 and it is primarily used to store weights and to stream them into WSE-2 for processing. The third generation (WSE-3), released in 2023, uses a 5 nm process node and employs 900,000 cores with 44 GB on-chip memory. In total, it uses 4 trillion transistors and delivers up to 125 PFLOP/s.

SambaNova's reconfigurable dataflow unit (RDU). RDU, developed by SambaNova Systems, is a machine-learning accelerator, implemented as a CGRA. It consists of a network of programmable on-chip compute engines, called Pattern Compute Units (PCUs), as well as distributed on-chip memory systems, called Pattern Memory Units (PMUs). PCUs and PMUs are connected through the programmable on-chip interconnect, called Tile-level Switch Network (SWN) (Emani et al., 2021; Prabhakar et al., 2022). By providing flexible space- and time-scheduling, and flexible memory system and interconnect, this architecture allows constructing custom dataflow pipelines based on the dataflow graphs of the applications. This enables efficient execution and reduced off-chip memory access (Hosseini et al., 2023). A key characteristic of RDU is its decentralized compute and memory units, as opposed to the von Neumann architecture that is used in CPUs and GPUs.

SambaNova provides a compiler, called SambaFlow, which captures the dataflow graph of the applications, determines the communication patterns, and creates spatial dataflow to exploit data locality and parallelism (Prabhakar and Jairath, 2021). The PCUs consist of SIMD ALUs that support FP32, BFloat16, 32-bit Integer, 16-bit Integer, and bitwise operations, whereas the PMUs contain software-managed SRAM (*Part II: Static random access memory* (Hanindhito et al., 2026)). The first generation of RDU, SN10 (2019), was manufactured via 7 nm process node

technology, with a total of 40 billion transistors, to implement 640 PCUs and 640 PMUs (320 MB of on-chip memory), achieving BFloat16 peak performance of 320 TFLOP/s (Prabhakar et al., 2022). The second generation, SN30 (2022), is manufactured by using the same process node, with a total of 86 billion transistors to double the number of PCUs and PMUs (640 MB on-chip memory), achieving BFloat16 peak performance of 688 TFLOP/s. The latest generation, SN40L (2024), is built on 5 nm, with 520 MB on-chip memory and 638 TFLOP/s peak BFloat16 performance. It also incorporates 64 GB of HBM for a peak bandwidth of 1.6 TB/s.

General NPU trend of semiconductor companies. The aforementioned examples underscore a clear industry-wide trend, where major technology companies are increasingly investing in custom AI hardware to improve performance and efficiency while reducing operational costs. Many other companies have developed their own large-scale ASICs to accelerate AI workloads in datacenters, provide rentable cloud service, and reduce dependence on costly third-party solutions like NVIDIA GPUs. Notable examples include Amazon's Inferentia2 and Trainium (Fu et al., 2024), Meta's MTIA (Firoozshahian et al., 2023), Intel's Gaudi (Kaplan, 2024) and Tesla's Dojo (Talpes et al., 2022).

NPUs in consumer devices. NPUs are also increasingly integrated with CPUs in consumer devices like desktops, smartphones and tablets. They typically handle tasks such as image processing, voice recognition, and real-time translations, reducing the burden on CPUs and GPUs. Notable examples include Samsung's Exynos and Qualcomm's Snapdragon chips (used in laptops, tablets and smartphones), as well as Apple's M-series (M1–M4, used in MacBooks and iPads) and A16–A18 (used in iPhones), reflecting the growing demand for AI-optimized computing in general-purpose devices.

What comes next?

Continuing the trend shown in Tables 6 and 7, future GPUs will have more transistors, and thus, more compute (e.g., CUDA) cores. Since the die size of the latest NVIDIA GPUs has already reached the limit of reticle size (*Transistor count and yield, and manufacturability*), using chiplets for future GPUs is inevitable. Both AMD and Intel are already using chiplets in their latest GPUs. Using chiplets makes manufacturing of different GPU products easier. For instance, different classes of GPUs may emerge, each having optimized features for specific use cases.

Another implication of the growth of (CUDA) cores is increased power consumption. With current power consumption sitting at 700 W, it is expected that future GPUs will have power consumption exceeding 1000 W. This implies liquid-based cooling will become common for GPU-accelerated compute nodes. Accordingly, a cluster that expects to host high-end GPUs should be prepared to provide a liquid-based cooling system. Moreover, the associated datacenter needs to have an improved power delivery system since the power density of each node and each rack will be significantly higher.

While the compute performance of individual GPUs is expected to grow, memory capacity and bandwidth may not be able to keep up with that pace. Using cutting-edge DRAM technologies (e.g., HBM4 and GDDR7) can only partially fulfill the bandwidth requirements. Therefore, using algorithms that reduce data-movement can have a considerable impact on attainable performance. With the stagnancy in individual GPUs' memory capacity, these algorithms will become even more important, as multi-GPU, multi-node computing will become more common to enable larger aggregate memory capacity, which is often needed to handle larger problems sizes. Accordingly, high-end clusters, such as those for machine learning, or advanced scientific computing, will use the most advanced inter-node communication technologies (*Part II: Inter-node communication* (Hanindhito et al., 2026)) to provide inter-node bandwidth that is comparable to off-chip memory bandwidth.

Summary and remarks

In order to deal with diverse applications, CPUs have a complex design, where the vast majority of the silicon die is tasked with "management" as opposed to performing "actual work". Many applications may not fully leverage the level of complexity that CPUs offer. For these "simple" applications, the silicon die can be used more effectively, by doing less "management" and performing more "actual work". Hardware accelerators, such as GPUs, follow this design philosophy.

GPUs were originally developed for processing graphics in gaming applications. They were rigid, and could not be programmed. As the complexity of graphics processing increased, GPUs became programmable. Machine-learning and cryptocurrency-mining also created a significant market for GPUs. Architecture of GPUs evolved to support these markets, for instance, by including matrix-multiplication accelerators. Relative ease-of-use of CUDA language for programming NVIDIA GPUs was key to its adoption.

GPUs played a fundamental role in the success of machine-learning. Observation of this success, and increased availability of GPUs in government laboratories and academia, motivated computational scientists to attempt running some scientific computing applications on GPUs. This is still an ongoing endeavor, as many scientific computing applications in industry and academia do not have GPU-friendly algorithms. Porting a CPU code into GPU often involves redesigning algorithms and rewriting the code, which could become a multi-year, multi-disciplinary effort. Scientific machine-learning (SciML) is becoming a popular trend in scientific computing. It relies on using

machine-learning-based approaches to solve scientific computing problems. Popularity of SciML approaches may be attributed to: easy-to-use high-level programming languages, such as TensorFlow and PyTorch, enabling rapid progress and providing various functionalities, such as automatic differentiation; ability of SciML techniques to seamlessly incorporate observations; taking advantage of open-source culture, fostered by the machine-learning community; and considerable funding opportunities.

High-end GPUs typically enjoy high-bandwidth memory, which enables significant performance gains as long as the application fits into the memory, and off-chip communication is minimal. Advances in packaging technology will likely result in larger GPUs with larger on-package, high-bandwidth memory; however, off-chip communication, if considerable, continues to be a bottleneck.

With the end of Moore's law, one of the very few ways to improve performance (e.g., faster run-time, or reduced energy consumption) is hardware customization. The cost of research and development for a custom hardware is significant. It is often a multi-year effort, which involves a large, multi-disciplinary team of hardware engineers, software engineers, and algorithm designers. Therefore, economic viability of custom hardware relies on a large-enough market for it. This cost may decrease in the future due to automation and availability of open-source tools.

FPGAs are typically used as substrates for prototyping hardware design. They enable fine-grained implementation and testing. Configuring them is tedious, and hardware code synthesis process is slow, which may improve in the future as they become more popular. CGRAs may also be used for custom hardware implementation. Compared to FPGAs, reconfiguring them is faster, since they come with specialized, hard-logic building blocks, and only allow coarse-grained reconfiguration; they also enable higher efficiency and performance. ASICs offer the highest hardware performance. However, once built, their design cannot change. Typically, they are used to implement mature designs.

Hardware-aware algorithms have a fundamental role to fully harness GPUs and custom hardware. In most cases, an algorithm-hardware co-design approach is necessary to maximize performance.

Remarks

In this paper (Part I), we covered background material, and reviewed technology trends in general-purpose processors, as well as hardware accelerators (e.g., GPUs). In Part II of this paper (Hanindhito et al., 2026), we will consider different memory technologies, inter-device communication, system integration and heterogeneous computing, and energy consumption of large computing centers and its implications. We will also offer perspectives on how these technology trends impact scientific computing.

Acknowledgements

Arash Fathi and Dimitar Trenev are grateful to ExxonMobil for supporting this work, and permitting its publication. We would like to thank Laurent White for stimulating conversations and commenting on an earlier draft of this paper, which improved its quality. We are also grateful to Amir Gholami, Arben Jusufi, Ardavan Pedram, Brent Wheelock, Chirath Neranjena, Dakshina Valiveti, Dimitri Papageorgiou, Rahul Sampath, and Wenting Xiao, for insightful conversations and feedback.

ORCID iDs

Bagus Hanindhito https://orcid.org/0000-0002-8485-581X
Arash Fathi https://orcid.org/0000-0002-6809-9815
Dimitrios Gourounas https://orcid.org/0000-0002-5011-9646

Funding

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by ExxonMobil Technology and Engineering, agreement number EM10480.36, National Science Foundation (NSF) grant numbers 2326894 and 2425655 and Division of Computing and Communication Foundations (grant no. 1763848). Any opinions, findings, conclusions, or recommendations are those of the authors and not of the sponsors.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Notes

1. The scope of this paper is limited to hardware and algorithms that are commonly used in high-performance scientific computing.
2. e.g., CUDA, TensorFlow, and PyTorch.
3. Process node or technology node refers to the semiconductor manufacturing process technology (Arden, 2002; Geppert, 2002; Patton, 2009), which includes different characteristics or features of the transistors, such as interconnect pitch, gate length, and transistor type. With smaller process nodes, the size of transistors becomes smaller, which allows higher transistor density, and typically faster and more power-efficient transistors (Gargini, 2002; Mallik et al., 2019).
4. CMOS stands for Complementary Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET), which is constructed by using n-type and p-type MOSFETs. The n-type MOSFET will turn on and conduct electrical current when the operating voltage is higher than the threshold voltage. On the other hand, the p-type MOSFET will turn on and conduct electrical current when the operating voltage is below the threshold voltage (Chang et al., 2000; Weste and Harris, 2010). If only n-type or p-type MOSFET is used as a switch, current will flow through the internal resistance of the transistor when it is on, dissipating power as heat, thus reducing efficiency. By

using both p-type and n-type MOSFETs where the gates are connected (i.e., same V for p-type and n-type MOSFETs), one of the transistors must be turned off, which ideally cuts the flow of electrical current during a steady state.

5. When two electrical conductors (e.g., copper wires) with different voltages are in close proximity, a capacitor can be formed with capacitance $C = \epsilon \frac{A}{d}$, where ϵ describes the permittivity of the material (i.e., how well dielectric material, such as silicon dioxide, stores an electric field), A is the cross-section area formed by the two conductors, and d is the distance between the two conductors. Although this capacitor is unwanted (and hence the name is parasitic capacitance), it is usually unavoidable (Dai et al., 2017). The capacitor stores energy in the form of an electric field, where the electric current I is related to the voltage V through $I = C \frac{dV}{dt}$. This relation implies the voltage cannot change abruptly, which also impacts how fast the transistor can switch states. For example, consider a low voltage (e.g., 0 V) is used to represent logic 0, while high voltage (e.g., 1 V) is used to represent logic 1. Switching the state from 0 to 1 requires charging the parasitic capacitance (increasing the voltage level), while switching the state from 1 to 0 requires discharging the parasitic capacitance (decreasing the voltage level). The charging and discharging of the capacitor cannot happen instantaneously (since voltage cannot change abruptly), resulting in switching delays that degrades the transistor's switching performance.
6. It takes time for both p-type and n-type MOSFETs to switch states. During this period, both transistors momentarily conduct electrical currents, and dissipate the power as heat (Veendrick, 1984).
7. e.g., voltage regulator module (VRM).
8. Modern microprocessors can draw more power for a very short time to provide a performance boost, and can draw very little power when they are idle (Rotem et al., 2012).
9. e.g., transistor size is becoming as small as the size of a few atoms.
10. This is due to smaller parasitic capacitance; see Section 2.8 on RC model.
11. State switching is easier in smaller transistors (Geppert, 2002; Moore, 2020), making them amenable to higher clock frequencies (Taylor, 2013).
12. Using the same operating voltage (V), transistors with lower threshold voltage (V_t) can switch states faster (Miyake et al., 2001; Wei et al., 1998), albeit, with an increased leakage current (Roy et al., 2003). In other words, there is a trade-off between switching speed (performance) and power: to improve the switching speed, the threshold voltage should be decreased. However, decreasing the threshold voltage will increase the leakage current, raising the static power (Butts and Sohi, 2000). As a result of this trade-off, there are usually different flavors of transistors on a chip: Transistors with lower V_t are used for critical paths of circuitry, where faster-switching speed is needed; transistors with higher V_t are used in most of the circuitry to minimize leakage current (Kim et al., 2003; Wei et al., 1998). Some strategies to reduce the static power (Singhal et al., 2015) include: a) turning off the transistor, by removing the voltage source when not in use (e.g., power gating, transistor gating (Taylor, 2012)); b) using multiple transistors with different threshold voltages (Deepaksubramanyan and Nunez, 2007; Mutoh et al., 1995); and c) geometrical or material modification of the transistor's gate (e.g., through using a 3D gate, instead of a planar gate and through using high dielectric materials (Kim et al., 2003; Vidya et al., 2018)).
13. Before reaching the nanometer era, dynamic power was the major contributor to the total power consumption of semiconductor devices (Kim et al., 2003).
14. This slowdown is also observed by a revision in the International Technology Roadmap for Semiconductors (ITRS) projection for physical gate lengths, from a 3-year step to a 5-year step (Iwai, 2009).
15. Package is a container or case made from either metal, plastic, glass, or ceramic, which holds one or more semiconductor dies (Greig, 2007; Wong and Wong, 1999). The package provides the semiconductor dies with protection against corrosion (Lachance et al., 1997), physical damage (Kaufman, 1999), and electro-static discharge (ESD) (Pan et al., 2022). It also provides outside interface with external peripherals and power supply (e.g., pins) (Chang et al., 1998; Mahajan et al., 2006b), and allows for mounting on printed-circuit-board (e.g., through socket (Pellerite and Suhl, 1988), or surface mount (Jagt, 1998)). It also acts as a thermal interface between the dies and the heat-sinks (Chang et al., 1998; Kohli et al., 2001).
16. The packaging substrates for MCM can be categorized as MCM-L (e.g., laminated FR4 printed circuit boards), MCM-C (e.g., ceramic substrates), and MCM-D (e.g., thin film deposited patterns) (Barnwell and Wood, 1997). Laminated FR4 printed circuit boards are commonly used as packaging substrates due to their low cost.
17. Other types of material for interposers include organic and glass.
18. e.g., through-silicon-vias (Khan et al., 2010; Yoon et al., 2009), through-glass-vias (Shorey and Lu, 2016; Takahashi et al., 2013).
19. e.g., extracting heat from the bottom layer of the stack.
20. e.g., CPU chiplet, GPU chiplet, accelerator chiplet, memory chiplet.
21. e.g., epoxy on PCB, and interaction with other copper wires.
22. RC models are most reliable when evaluating the performance of the wires at low to medium frequencies. At higher frequencies, RLC models become more effective, since inductance (L) effects become more important. Either way, the parasitic resistance, capacitance, and inductance, limit the data rate that a wire can carry, due to their impact on data integrity (Albina and Hackl, 2007; Kleveland et al., 2002).
23. e.g., silicon dioxide for on-chip wires, or fiber epoxy compound for on-PCB wires.
24. On-chip interconnect comprises several metal layers stacked onto the top of each chip (Ho et al., 2001; Sylvester and Keutzer, 2000). The lower layer is used to construct local

interconnects, which consist of thin and short wires that connect transistors over very short distances. In the middle layer, intermediate interconnects are constructed using thicker wires, which provide connections between different blocks of a chip. The higher layer is reserved for global interconnects, which have even thicker and longer wires.

25. IBM reported 30% improvement in microprocessor speed (from 300 MHz to 400 MHz, using the same process node) by moving from aluminum to copper for on-chip wires (Staff, 2019).
26. i.e., 40% smaller than aluminum (Gelatos et al., 1994).
27. e.g., electromigration (Hu et al., 2009; Ryu et al., 1999).
28. Copper also has a limited bit-rate (Bogatin, 2022; Miller and Ozaktas, 1997), which caps the bus clock frequency for data transmission.
29. e.g., ruthenium (Ru), cobalt (Co), nickel (Ni), and other platinum-group metals (Bonilla et al., 2020; Huang et al., 2020; Staff, 2019).
30. Growth temperature is the temperature needed to manufacture and synthesize the carbon nanotube (Li et al., 2019; Venkataraman et al., 2019), ranging from 600°C to 1000°C. This temperature is not suitable for sensitive substrates, such as silicon. Further research is being conducted for developing a low-temperature, Si-compatible, CNT growth process (Chen et al., 2011; Duesberg et al., 2004; Haque et al., 2008; Todri-Sanial et al., 2017).
31. Silicon photonics is a research field that aims to place complex optical functions on a chip, by using CMOS process technology (Bogaerts and Chrostowski, 2018; Rahim et al., 2018).
32. e.g., bandwidth density (i.e., bandwidth per amount of silicon area used), latency, and energy dissipation metrics (Miller, 2009; Rakheja and Kumar, 2012).
33. Vertical-cavity surface-emitting laser.
34. e.g., by blocking or passing light, for bit 0 and bit 1, respectively.
35. WDM uses multiple wavelengths (light colors) to carry multiple signals, by using the same waveguide. The waveguide is a channel that allows passage of light. With WDM, a single waveguide can carry multiple different signals (i.e., more data), which needs dozens of metal wires in electrical interconnects (Mekawey et al., 2022; Ryu et al., 1999).
36. The majority of the cost of chip design is due to licensing of the electronic design automation (EDA) software, and verifying the chip functionality in each step of the development (Bauer et al., 2020; Hruska, 2018). The rest is spent on the acquisition of semiconductor intellectual property (IP), architecture and logic design, physical implementation, as well as prototyping and validation.
37. For instance, some flight computers in an airplane use multiple processors on the same data, and then use majority voting to decide the correct result (Spector and Gifford, 1984).
38. e.g., in the form of binary codes.
39. e.g., by developing different programs or instructions.
40. There exists CPUs and GPUs that are implemented on soft-logic fabric (e.g., FPGAs), which are referred to as soft-core. Examples include Altera NIOS CPUs, Xilinx MicroBlaze CPUs (Tong et al., 2006), and logi3D 3D Graphics Accelerators GPUs (Xylon d.o.o, 2021).
41. The pins are arranged in a grid array in different ways: PGA (Ghosal et al., 2001; Yeoh et al., 2000), LGA (Corbin et al., 2002; Pandey et al., 2005), or BGA (Chen et al., 2002; Shah and Mello, 2004). In pin grid array (PGA), the male contact is on the microprocessor side while the socket houses the female contact. As the pin gets smaller, the male contact on the microprocessor becomes more fragile, making it difficult to safely mount the microprocessor on the socket. The land grid array (LGA) alleviates this difficulty by placing the male contacts on the motherboard socket while the microprocessor holds the contact pads. The ball grid array (BGA) is used when the microprocessor is soldered directly into the motherboard, providing excellent connection at the expense of making it very difficult to replace (e.g., for upgrading or repairing).
42. e.g., through adding more memory channels.
43. Through a communication protocol, e.g., PCI Express.
44. For example, Socket SP3 in AMD EPYC Milan (2021) has 4094 pins with 2072 of them used for supplying 280 W power to the processor while Socket SP5 in AMD EYPC Genoa (2022) has 6096 pins with 3047 of them used for supplying 400 W power to the processor.
45. Pitch is the distance between the center of pins.
46. Control signals (Figure 4(a)) may include: a) clock signal to synchronize data transmission; b) device selector signal to choose which device should the host microprocessor communicate with (e.g., in the case of multiple devices sharing the same communication link); c) transaction type, i.e., whether it is a read transaction (e.g., data flow from device to the host microprocessor), or a write transaction (e.g., data flow from the host microprocessor to device); and d) other protocol-specific signals that regulate the data transmission (Fawcett, 1995; Wu et al., 2009).
47. A half-duplex interface cannot send and receive data at the same time, since the same wires are used for both sending and receiving (Figure 4(d)) (Dawoud and Dawoud, 2020; Summerville, 2009). For instance, two-way communication through a walkie-talkie is half-duplex, since only one person can talk at a time.
48. Each transmitted bit through a different wire must be synchronized and arrive on the receiving end at the same time, as they share the same clock signal. It becomes more difficult to maintain this synchronization as the number of wires increases, especially, when the wires are not straight, have different lengths, and are over a longer distance (Hu and Yuan, 2009; Lee et al., 2013).
49. As the data rate increases, higher switching activity of a wire can generate severe electromagnetic interference, which then affects adjacent wires (cross-talk), possibly causing a bit-flip

that compromises data integrity (Frenzel, 2007; Weng et al., 2018).

50. Due to routing on PCB.
51. The serial interface, serializes the data transfer, by sending each bit one by one, based on a clock signal. The clock signal can be sent separately (e.g., by using a dedicated lane) (Liu and Liu, 2017; Pandey et al., 2020), or, it can be embedded into the serial data itself through line coding (e.g., Manchester pulse shape or self-synchronizing block coding; Section 2.14) (Cheng et al., 2008; Gohel, 2012).
52. To transmit the data serially in each direction, the single-ended interface uses only one wire, while the differential-pair uses two wires (Figure 4(e)). In the differential-pair interface, both wires carry the same data, but with opposite voltage polarity. The receiver will extract the data by finding the voltage difference between the pair. The differential-pair interface is therefore more immune to electromagnetic interference: assuming both wires are subjected to the same electromagnetic interference (Chen and Katopis, 2004; Mechaik, 2001), the voltage difference between the pair remains constant, and thus, the original data remains intact.
53. Physical layer (PHY) is the bottom-most layer in the computing system model (e.g., Open Systems Interconnection (Li et al., 2011)). It directly interfaces with the transmission medium (e.g., copper wires), and provides mechanical, electrical, functional, and procedural characteristics for bit transmissions (McClelland, 1983).
54. e.g., PCI Express (PCIe), Universal Serial Bus (USB) (Halvorsen and Clarke, 2011), High-Definition Multimedia Interface (HDMI) (Eidson et al., 2003), and Serial AT Attachment (SATA) (Grimsrud and Smith, 2003).
55. For instance, a 64-bit microprocessor has a 64-bit-wide parallel internal bus. The serializer component of SerDes converts the parallel 64-bit data stream into a 1-bit serial data stream, before sending the data to the serial external bus. On the receiving end, the data is received through a serial external bus of a microprocessor. Then, the deserializer component of SerDes converts the serial data stream back into a parallel, 64-bit data stream.
56. i.e., data rate that can be transmitted over a medium.
57. Number of bits sent per second.
58. Number of signal elements sent per second.
59. There are two modulation types: baseband and passband (Alyaei and Glass, 2009; Sundareshan, 1992). Baseband modulation is performed without modifying the frequency with which the data is generated (i.e., no spectrum transformation). Passband modulation is usually used for wireless radio communication (e.g., Wi-Fi, LTE, 5G), and thus, is not explored here.
60. e.g., unipolar NRZ, unipolar RZ, polar NRZ-level, polar NRZ-invert, polar RZ, and bipolar NRZ (i.e., alternate-mark inversion or pseudoternary).
61. The length of the pulse defines the bit-time.
62. Except for unipolar RZ, which still suffers from synchronization problems when long consecutive zero voltage is transmitted.
63. Assuming an operating voltage of 1V, two levels of voltage (e.g., $0V$ and $1V$) in NRZ can be used to represent bit 0 and bit 1, with $1V$ difference. With the same operating voltage, PAM-4 will need four levels of voltage (e.g., $0V$, 333 mV, 666 mV, and $1V$) to represent symbols 00, 01, 11, and 10. The smaller difference between each level makes PAM-4 more susceptible to noise (Chen et al., 2023; Dikhaminjia et al., 2016).
64. Bit error rate (BER) is the probability of bit error (Jeruchim, 1984). For instance, an BER of 10^{-6} expects to have one wrong bit, over 1 million transmitted bits.
65. i.e., voltage swing.
66. i.e., DC-balanced.
67. e.g., PCIe 6.0, InfiniBand HDR, and InfiniBand NDR.
68. e.g., Serial ATA (Grimsrud and Smith, 2003), Serial Attached SCSI-1 to SCSI-3 (SAS 1.x to SAS 3.x) (Cai et al., 2005), Gigabit Ethernet (Frazier, 1998), InfiniBand SDR/DDR/QDR (Eddington, 2002), and PCI Express 1.x and 2.x.
69. e.g., 10 Gbps Ethernet (Cunningham, 2001), InfiniBand FDR, EDR, and HDR.
70. e.g., PCI Express 3.0, 4.0 and 5.0.
71. This will become clear later.
72. e.g., silicon area.
73. Data before 2010 were obtained from Horowitz et al. (2015); Danowitz et al. (2012b,a); Olukotun and Hammond (2005), while data between 2010 and 2021 were obtained from Rupp (2022). We augmented the above with data from 2022–2023, and with trend projections based on publicly available industry roadmaps and extrapolation of previous decades' trends.
74. Since clock frequency is not a reliable metric for measuring and comparing the performance of modern microprocessors (Section 2.1), instruction per cycle (IPC) is often used as a performance metric instead (Emma, 1997). However, IPC is not a reliable metric either (Alameldeen and Wood, 2006) since different workloads have different instruction mixes, and each micro-architecture can perform differently under different instruction mixes, resulting in a different IPC. Using benchmark suites, such as SPEC (Standard Performance Evaluation Corporation) (Bays and Lange, 2012; Dujmovic and Dujmovic, 1998; Henning, 2000), is a better way of measuring the performance of different microprocessors, as they establish a frame of reference through standardized micro-benchmarks that represent the majority of the workloads in the industry.
75. Modern microprocessors can execute more than one instruction per clock cycle by dispatching more than one instruction to keep multiple functional units busy (i.e., superscalar) (Palacharla et al., 1997; Smith and Sohi, 1995). Moreover, the execution path of the microprocessor is often divided into several stages (e.g., instruction fetch, instruction decode, execution, memory access, and write-back) to achieve higher throughput, by allowing each stage to work on different instructions simultaneously (e.g., super-pipeline) (Dysart et al., 2004; Srinivasan et al., 2002).

76. Early microprocessors executed instructions in the order they were written in an application. An instruction could take longer to finish (e.g., due to memory access, or, complexity of the operations), causing subsequent instructions to wait. In an out-of-order execution engine (Peleg and Weister, 1991), these subsequent instructions can be executed without waiting for the blocking instruction to finish, provided that the subsequent instructions are not dependent on the results of the blocking instruction and there are resources available (e.g., ALUs) to execute these instructions. At the end of the pipeline, a reorder buffer (ROB) (Eyerman et al., 2009) is responsible for re-arranging the results of the instructions, such that they are retired in the order of how they should have been executed by the application (i.e., out-of-order execution, in-order retirement).
77. General-purpose microprocessors are usually equipped with vector processing units to efficiently execute vector and matrix operations, which are common in many classes of applications. Although the development of vector processors started as early as the 1970s (Calahan and Ames, 1979; Kolodzey, 1981), the first vector extension in the x86 architecture, known as MMX, was introduced by Intel in 1997 (Peleg et al., 1997). This was followed by 3DNow! (1998) (Oberman et al., 1999), the Streaming SIMD Extensions (SSE) in 1999 (Raman et al., 2000), which was then succeeded by SSE2 (2001) (Mehis and Radhakrishnan, 2002), SSE3 (2004), Supplemental SSE3 (2006), and SSE4 (2006), the Advanced Vector Extension (AVX) in 2008, and then succeeded by AVX2 (2013), and AVX-512 (2015), and the newest Advanced Matrix Extension (AMX) in 2022 (Nassif et al., 2022) and AVX10 in 2023.
78. Branch instructions, such as if-then-else statements, make the microprocessor diverge from the current instruction sequence into possibly a different sequence. It takes time for the microprocessor to evaluate the outcome of the branch, and assess which conditional branch they should take, causing a stall on the execution pipeline. A branch predictor (Mittal, 2019) predicts which branch should be taken, based on the branch's history. Thereafter, the microprocessor can directly execute (the speculative) subsequent instructions, thus avoiding pipeline stall, which then improves the overall performance (Quinones et al., 2007). However, false predictions (Ball and Larus, 1993; Gandhi et al., 2004) are costly, as the microprocessor needs to undo the speculatively executed instructions and execute the correct instruction sequence.
79. Generally, there are three types of parallelism that can be exploited (Hill and Reddi, 2021): instruction-level parallelism (ILP) (Rau and Fisher, 2003; Wall, 1991), thread-level parallelism (TLP) (Blake et al., 2010; Keckler et al., 1998), and data-level parallelism (DLP) (Espasa and Valero, 1997). ILP attempts to find multiple instructions from a sequence of instructions (e.g., in an application) that can be executed at the same time, as long as there is no dependency between those instructions, and hardware resources to execute them are available (Pai et al., 1997; Rau and Fisher, 2003). The out-of-order execution engine discussed earlier tries to extract ILP from applications that are hidden from a programmers' perspective (Schlansker et al., 1997). TLP uses multiple threads to improve an application's performance, whether they run the same instructions but work on different data, or they run asymmetrically (e.g., master-slave paradigm); the operating system schedules the threads into available hardware threads, e.g., through the use of simultaneous multithreading (SMT) (Eggers et al., 1997; Magro et al., 2002) or multiple physical cores. However, SMT is becoming less common or is often disabled in datacenter CPUs, either to conserve chip area or to reduce complexity and avoid security-related issues in cloud environments. DLP (Espasa and Valero, 1997) refers to performing the same operations on different data at the same time, such as using vector extensions to perform SIMD operations (e.g., scaling a vector that contains thousands of elements).
80. For example, AMD EPYC Naples has four dies, each containing eight cores, two memory channels, and 32 PCI Express 3.0 lanes (Naffziger et al., 2020).
81. For example, the AMD EPYC Rome and AMD EPYC Milan has eight chiplets, manufactured in 7 nm process node. Each chiplet has eight cores, and an I/O chiplet that is manufactured in 12 nm process node (Mattioli, 2021).
82. For example, AMD EPYC Milan-X microprocessors implement a 3D vertical cache to add more capacity to the last-level cache (Agarwal et al., 2022).
83. For instance, while AMD EPYC Genoa has a typical power of 400 W, it can consume up to 700 W for a short time (1 ms) (Mujtaba, 2022), if it ensures both the power delivery and cooling systems can handle this burst in power consumption.
84. For instance, the SP5 socket for AMD EPYC Genoa (2022) has 6096 pins, which is 48.9% more than its predecessor, the SP3 socket (2021), with 4094 pins. Despite its smaller pitch (0.81 mm × 0.94 mm on SP5 vs 0.87 mm × 1.00 mm on SP3), the SP5 is 25% larger than the SP3. These extra pins are used for adding more memory channels (12 on SP5 vs 8 on SP3) and supplying more power to the microprocessor (400 W on SP5 vs 280 W on SP3). For reference, the highest-core-count microprocessor supported on the SP5 socket has 90 billion transistors, compared to 40 billion transistors in the highest-core-count processor on SP3, a 125% increase.
85. e.g., fewer branches, uniform computation patterns, and predictable memory accesses.
86. The Intel Xeon Phi product lineup was first introduced in 2010 as Knights Ferry (Duran and Klemm, 2012) which was a derivative of Intel GPGPU Larrabee architecture (Heinecke et al., 2012; Seiler et al., 2009) and Intel experimental architecture called Single-Chip Cloud Computer (SCC) (Grelck and Sarris, 2017). Although Knights Ferry was only made for a working prototype (Mittal, 2020a), its successor, Knights Corner (2012) (Chrysos, 2012) and Knights Landing (2016) (Sodani et al., 2016), found their way into HPC clusters, such as Stampede (Dubrow, 2015) and Stampede 2

(Stanzione et al., 2017) of the Texas Advanced Computing Center (TACC).

87. Prior to Knights Mills, Intel announced Knights Hills in 2014, based on 10 nm process node. It was expected to be used in a number of supercomputers, including the U.S. Department of Energy's exascale supercomputer Aurora (Mujtaba, 2015). However, delays in the 10 nm process node pushed Knights Hills to 2017, which was eventually canceled and was replaced by Knights Mills, manufactured in 14 nm (Mittal, 2020a).
88. e.g., Amazon AWS, Microsoft Azure, and Google Cloud Platform (Kaushik et al., 2021).
89. Zen4c cores are simpler and more power efficient than the fully-fledged Zen4 cores used in AMD EPYC Genoa (2023). Zen4c is 35% smaller than Zen4 in terms of silicon area, which allows Bergamo to house 128 cores, compared to Genoa with 96 cores on the same package (SP5) and the same power envelope (400 W).
90. e.g., cache configurations, memory hierarchy, etc.
91. This slow increase in clock frequency is primarily enabled by minor refinements of the process node used for manufacturing the chips.
92. For example, the 4th Gen AMD EPYC processors have four product lines with different core architecture (Sideco, 2023): a) Genoa with up to 96 cores for general-purpose computing; b) Bergamo with up to 128 cores for cloud applications (e.g., containers and micro-services); c) Genoa-X with 3D vertical cache, and up to 96 cores for HPC applications and technical computing (e.g., finite element simulations, electronic design automation, computational fluid dynamics); and d) Siena for telecommunication, and low-power edge Computing, with up to 64 cores in smaller SP6 package (e.g., 5G and network traffic management).
93. Interestingly, Intel Xeon Emerald Rapids, the successor of Intel Xeon Sapphire Rapids, is expected to have only two chiplets, as opposed to four chiplets. Compared to Sapphire Rapids, it has more cores (64 vs 60), and 2.84× larger L3 cache (320 MB vs 112.5 MB), while occupying relatively the same silicon area (1526 mm^2 vs 1575 mm^2). The decision to use two larger chiplets, instead of four smaller chiplets, is likely motivated by four reasons: a) Intel was able to improve the 7 nm process node (i.e., the same process node used for Sapphire Rapids), resulting in slightly better yield; b) Intel put an extra core per chiplet for a total of 66 cores across two chiplets to improve yield (e.g., one defective core can be replaced by the spare core); nevertheless, it will only have 64 active cores; c) Intel improved design and floor-planning to use silicon area more efficiently; and d) the reduced number of chiplets decrease the silicon area required for implementing chiplet-to-chiplet interconnect, resulting in more efficient silicon area usage. We remark chip design is about design trade-offs. Manufacturers strive to find a good balance between various metrics (e.g., performance, power consumption, area, cost, yield, etc.).
94. For instance, energy consumed to fetch and decode an instruction in general-purpose microprocessors is 10× to 4000× higher than the energy needed to perform simple arithmetic operations (e.g., addition, subtraction) (Dally et al., 2020).
95. To limit power consumption, transistors corresponding to these advanced features are turned off when an application does not use them, leading to underutilized silicon area.
96. e.g., due to the overhead of instruction fetch and decode, and branch misprediction penalty.
97. Either on the same die, on the same package, or as a separate peripheral.
98. e.g., performance, latency, or energy efficiency.
99. e.g., it may take an accelerator only a single clock cycle to perform an operation that may consume tens to hundreds of cycles on a general-purpose microprocessor. Moreover, specialized functional units can support new data types, such as BFloat16 (Wang and Kanwar, 2019), quarter precision (FP8) (Wang et al., 2018), and configurable FP8/FP16 (CFP8/CFP16) (Talpes et al., 2023), which are low-precision data formats that are typically used in machine learning.
100. For instance, using SIMD is often more efficient for explicitly parallel applications compared to MIMD (Section 2.11). Moreover, Very Long Instruction Words (VLIW) may be preferred over out-of-order execution for some classes of applications. VLIW relies on the compiler to schedule the instructions in order to achieve instruction-level parallelism, which is often easier and more efficient for explicitly parallel applications, compared to the hardware-managed out-of-order execution engine.
101. e.g., the size of local (on-chip) memory can be optimized for specific workloads and access patterns. This allows functional units to perform computations efficiently, and minimize performance penalties associated with accessing global (off-chip) memory.
102. e.g., simplified control flow may hardcode a particular execution sequence of operations, which avoids the need and overhead associated with interpreting more generic instructions.
103. Vertex is a point in 2D or 3D space where two or more lines are met. In computer graphics, a vertex is represented as a data structure that describes attributes of a point in 2D or 3D space, such as its position. A vertex shader is a part of the GPU that handles the transformation of raw geometry of the vertices, either through fixed-function hardware (Angel and Shreiner, 2011), or through a vertex program running in a programmable shader (Lindholm et al., 2001). The transformation includes the translation of triangle vertices from 3D space into 2D space for screen viewing (Peddie, 2023a).
104. A primitive is the simplest geometric shape in a computer graphics system with which more complex geometry can be built. An example of a primitive (Peddie, 2023c) includes a point (Alexa et al., 2004), line (Stoll et al., 2005), or triangle (Deering et al., 1988). The latter is widely-used in computer graphics due to its simplicity and efficiency for geometry generation and processing algorithms: surfaces of any shape

can be represented by using triangle meshes without complicated smoothness conditions (Kobbelt and Botsch, 2004). Primitives generated by the vertex shader are accepted by the geometry shader for further processing, which includes tessellation and shadow volume extrusion (Peddie, 2023a).

105. Fragment is a part of data that is needed to generate pixels, which includes color, texture, and transparency, among other features (Goodnight et al., 2005; Peddie, 2023a). A fragment shader or pixel shader performs the computation of color and other attributes of each fragment (Peddie, 2023c).
106. Pixel stands for Picture Elements, which represents the smallest unit of display that is handled by the computer (Peddie, 2023c). Pixel contains color information (e.g., RGB luminance or chrominance).
107. With vertex shader and pixel (fragment) shader implemented in separate hardware pipeline stages, there is a possibility of computation imbalance (Mochocki et al., 2006), where the vertex shader is idle while the pixel (fragment) shader is busy processing the backlogged operations (Peddie, 2023a). For instance, a scene from a graphics application may use large triangles to draw a massive distant sky which overwhelms the pixel (fragment) shader, leaving the vertex shader idle. A couple of milliseconds later, the scene switches to a more detailed view of nearby objects, using many smaller triangles that overwhelm the vertex shader, leaving the pixel (fragment) shader idle (Lindholm et al., 2008; Luebke and Humphreys, 2007).
108. A GPU kernel is a single stream of instructions that operates on large number of data-parallel works (Peddie, 2023b).
109. For instance, NVIDIA H100 has already reached a die area of 814 mm^2, which is close to the current reticle limit of 858 mm^2 (Lai, 2021; Suzuki, 2020) (Section 2.5).
110. For instance, AMD Mi300 GPUs (2023) have three types of chiplets: GPU dies (XCD), CPU dies (CCD), and High-Bandwidth Memory (HBM) stacks. The number of chiplets varies depending on the configuration: Mi300a has 6 XCD, 3 CCD, and 8 HBM stacks for CPU and GPU heterogeneous computing; Mi300X has 8 XCD and 8 HBM stacks for accelerating highly-parallel workloads that target GPUs; Mi300c is a CPU-only version with 12 CCD and 8 HBM stacks to compete with Intel Sapphire Rapids with HBM-equipped CPUs; and Mi300P is a smaller version, targeted for PCIe form factor, and has 4 XCD and 4 HBM stacks. The top configuration has 153 billion transistors, which is almost twice than that in NVIDIA H100.
111. i.e., different results of if-then-else statements.
112. e.g., register files and shared memory.
113. GPU and its off-chip memory are closely integrated. The memory is on the same PCB for GDDR (Part II Section 2.2.5 (Hanindhito et al., 2026)), or on the same package for HBM (Part II Section 2.2.6 (Hanindhito et al., 2026)). This allows for higher bandwidth (Part II Section 2.2.5 (Hanindhito et al., 2026)) at the expense of smaller capacity, and almost no upgrade-ability.
114. e.g., in thousands or millions.
115. e.g., libraries, frameworks, compilers.
116. Major FPGA manufacturers include Altera (bought by Intel in 2015) and Xilinx (bought by AMD in 2022).
117. e.g., machine learning.
118. Such ASICs are sometimes referred to as Application Specific Instruction Set Processors (ASIPs) (Keutzer et al., 2002).
119. Not to be confused with TPUv4i, which is the fourth-generation TPU introduced in 2020 for inference workloads (Jouppi et al., 2021).
120. Not to be confused with Intel's Infrastructure Processing Unit (IPU).
121. e.g., SSE, AVX.
122. By comparison, NVIDIA H100 GPU (2022) only has 116 MB of on-chip memory, consisting of registers, L1 cache, shared memory, and L2 cache. See Table 7.
123. In semiconductor manufacturing, a wafer is a thin slice of highly-pure crystallized silicon in a circular shape, through which the transistors and the interconnection between them are manufactured to produce chips (Hahn, 2001; Tilli, 2010). Usually, one wafer is used to implement 10 s to 100 s of dies before they are individually cut ("dicing") (Luo and Wang, 2008; Marks et al., 2022) and put into a package (Greig, 2007; Wong and Wong, 1999).
124. e.g., different memory size and bandwidth, interconnect topology.

References

Aamodt TM, Lun Fung WW and Rogers TG (2018) *Programming Model*. Chapter 2. Springer International Publishing, pp. 9–20.

Abadi M, Agarwal A, Barham P, et al. (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abdelfattah MS and Betz V (2012) Design tradeoffs for hard and soft FPGA-based networks-on-chip. In: 2012 international conference on field-programmable technology, Seoul, South Korea, 10–12 December 2012, pp. 95–103.

Abdennadher S, Tripician K and Singaravelu S (2020) At speed testing challenges and solutions for 56Gbps and 112Gbps PAM4 SerDes. In: 2020 IEEE Latin-American test symposium (LATS), Maceio, Brazil, 30 March–2 April 2020, pp. 1–5.

Agarwal V, Hrishikesh MS, Keckler SW, et al. (2000) Clock rate versus IPC: the end of the road for conventional micro-architectures. *SIGARCH Computer Architecture News* 28(2): 248–259.

Agarwal R, Cheng P, Shah P, et al. (2022) 3D packaging for heterogeneous integration. In: 2022 IEEE 72nd electronic components and technology conference (ECTC), San Diego, California, 31 May–3 June 2022, pp. 1103–1107.

Ahmad S, Subramanian S, Boppana V, et al. (2019) Xilinx first 7nm device: Versal AI core (VC1902). In: 2019 IEEE hot chips 31 symposium (HCS), Cupertino, CA, 18–20 August 2019, pp. 1–28.

Ahmed K and Schuegraf K (2011) Transistor wars. *IEEE Spectrum* 48(11): 50–66.

Ahn JH, Jouppi NP, Kozyrakis C, et al. (2009) Future scaling of processor-memory interfaces. In: Proceedings of the conference on high performance computing networking, storage and analysis, SC ’09. New York, NY: Association for Computing Machinery, pp. 1–12.

Aktemur B, Metzger M, Saiapova N, et al. (2020) Debugging SYCL programs on heterogeneous Intel® architectures. In: Proceedings of the international workshop on OpenCL, IWOCL ’20. New York, NY: Association for Computing Machinery, pp. 1–10.

Alameldeen A and Wood D (2006) IPC considered harmful for multiprocessor workloads. *IEEE Micro* 26(4): 8–17.

Albina CM and Hackl G (2007) Layout parasitic interconnections effects on high frequency circuits. In: 2007 6th IEEE Dallas circuits and systems workshop on system-on-chip, Dallas, TX, 15-16 November 2007, pp. 1–4.

Aleksic S (2017) The future of optical interconnects for data centers: a review of technology trends. In: 2017 14th international conference on telecommunications (ConTEL), Zagreb, Croatia, 28–30 June 2017, pp. 41–46.

Alexa M, Gross M, Pauly M, et al. (2004) Point-based computer graphics. In: ACM SIGGRAPH 2004 course notes, SIGGRAPH ’04. New York, NY: Association for Computing Machinery, p. 7.

Altaf MSB and Wood DA (2017) LogCA: a high-level performance model for hardware accelerators. In: Proceedings of the 44th annual international symposium on computer architecture, ISCA ’17. New York, NY: Association for Computing Machinery, pp. 375–388.

Alyaei BR and Glass A (2009) Line coded modulation. In: 2009 3rd international conference on signal processing and communication systems, Omaha, Nebraska, 28–30 September 2009, pp. 1–4.

Amann MC and Hofmann W (2009) InP-based long-wavelength VCSELs and VCSEL arrays. *IEEE Journal of Selected Topics in Quantum Electronics* 15(3): 861–868.

Ampere Computing LLC (2022) *Ampere® Altra® Max 64-bit Multi-Core Processor Featrures*. Whitepaper, Ampere Computing LLC.

Anand S and Razavi B (2001) A CMOS clock recovery circuit for 2.5-Gb/s NRZ data. *IEEE Journal of Solid-State Circuits* 36(3): 432–439.

Anderson R, Bond M, Clulow J, et al. (2006) Cryptographic processors-a survey. *Proceedings of the IEEE* 94(2): 357–369.

Andricacos PC (1999) Copper on-chip interconnections: a breakthrough in electrodeposition to make better chips. *The Electrochemical Society Interface* 8(1): 32–37.

Andricacos PC, Uzoh C, Dukovic JO, et al. (1998) Damascene copper electroplating for chip interconnections. *IBM Journal of Research and Development* 42(5): 567–574.

Angel E and Shreiner D (2011) Teaching a shader-based introduction to computer graphics. *IEEE Computer Graphics and Applications* 31(2): 9–13.

Anzt H, Tsai YM, Abdelfattah A, et al. (2020) Evaluating the performance of NVIDIA’s A100 ampere GPU for sparse and batched computations. In: 2020 IEEE/ACM performance modeling, benchmarking and simulation of high performance computer systems (PMBS), Atlanta, Georgia, 12 November 2020, pp. 26–38.

Arcelin B (2021) Comparison of graphcore IPUs and Nvidia GPUs for cosmology applications.

Arden WM (2002) The international technology roadmap for semiconductors—Perspectives and challenges for the next 15 years. *Current Opinion in Solid State and Materials Science* 6(5): 371–377.

Arora A, Mehta S, Betz V, et al. (2021) Tensor slices to the rescue: supercharging ML acceleration on FPGAs. In: The 2021 ACM/SIGDA international symposium on field-programmable gate arrays, FPGA ’21. New York, NY: Association for Computing Machinery, pp. 23–33.

Arora A, Anand T, Borda A, et al. (2022) CoMeFa: compute-in-memory blocks for FPGAs. In: 2022 IEEE 30th annual international symposium on field-programmable custom computing machines (FCCM), New York City, NY, 15–18 May 2022,pp. 1–9.

Arunkumar A, Bolotin E, Cho B, et al. (2017) MCM-GPU: multi-chip-module GPUs for continued performance scalability. In: Proceedings of the 44th annual international symposium on computer architecture, ISCA ’17. New York, NY: Association for Computing Machinery, pp. 320–332.

Avižienis A (1975) Fault-tolerance and fault-intolerance: complementary approaches to reliable computing. *ACM SIGPLAN Notices* 10(6): 458–464.

Ayar Labs Inc (2021) *Optical I/O Chiplets Eliminate Bottlenecks to Unleash Innovation*. Whitepaper, Ayar Labs Inc.

Aygün K, Hill MJ, Eilert K, et al. (2005) Power delivery for high-performance microprocessors. *Intel Technology Journal* 9(4): 273–283.

Bai Y, Bandyopadhyay N, Tsao S, et al. (2011) Room temperature quantum cascade lasers with 27% wall plug efficiency. *Applied Physics Letters* 98(18): 181102.

Balasubramanian K, Agili S and Morales A (2011) Investigating the new 64b/66b encoding scheme’s power spectral density. In: 2011 IEEE international conference on consumer electronics (ICCE), Las Vegas, Nevada, 9–12 January 2011, pp. 377–378.

Balewski J, Liu Z, Tsyplikhin A, et al. (2022) Time-series ML-regression on graphcore IPU-M2000 and Nvidia A100. In: 2022 IEEE/ACM international workshop on performance modeling, benchmarking and simulation of high performance computer systems (PMBS), Dallas, TX, 13–18 November 2022, pp. 141–146.

Ball T and Larus JR (1993) Branch prediction for free. In: Proceedings of the ACM SIGPLAN 1993 conference on programming language design and implementation, PLDI ’93. New York, NY: Association for Computing Machinery, pp. 300–313.

Bandyopadhyay J and Cases M (2000) Packaging challenges in the design of a 800 Mbps source-synchronous simultaneous

bi-directional parallel interface. In: IEEE 9th topical meeting on electrical performance of electronic packaging (Cat. No.00TH8524), Scottsdale, AZ, 23–25 October 2000, pp. 9–12.

Barnwell P and Wood J (1997) A novel thick film on ceramic MCM technology offering MCM-D performance. In: Proceedings 1997 international conference on multichip modules, Santa Cruz, CA, 4–5 February 1997, pp. 48–52.

Bashir J, Peter E and Sarangi SR (2019) A survey of on-chip optical interconnects. *ACM Computing Surveys* 51(6): 1–34.

Bauer H, Burkacky O, Kenevan P, et al. (2020) *Semiconductor Design and Manufacturing: Achieving Leading-Edge Capabilities*. Report. McKinsey & Company.

Bays W and Lange KD (2012) SPEC: driving better benchmarks. In: Proceedings of the 3rd ACM/SPEC international conference on performance engineering, ICPE '12. New York, NY: Association for Computing Machinery, pp. 249–250.

Beck N, White S, Paraschou M, et al. (2018) 'Zeppelin': an SoC for multichip architectures. In: 2018 IEEE international solid - state circuits conference - (ISSCC), San Francisco, CA, 11–15 February 2018, pp. 40–42.

Belletti F, Cotallo M, Cruz A, et al. (2009) Janus: an FPGA-based system for high-performance scientific computing. *Computing in Science & Engineering* 11(1): 48–58.

Berg T and Siegel H (1991) Instruction execution trade-offs for SIMD vs. MIMD vs. mixed mode parallelism. In: [1991] Proceedings. the fifth international parallel processing symposium, Anaheim, CA, 30 April–02 May 1991, pp. 301–308.

Besta M, Stanojevic D, Licht JDF, et al. (2019) Graph processing on FPGAs: taxonomy, survey, challenges.

Beyne E (2003) Cu interconnects and low-k dielectrics, challenges for chip interconnections and packaging. In: Proceedings of the IEEE 2003 international interconnect technology conference (Cat. No.03TH8695), Burlingame, CA, 04 June 2003, pp. 221–223.

Beyne E, Milojevic D, Van der Plas G, et al. (2021) 3D SoC integration, beyond 2.5D chiplets. In: 2021 IEEE international electron devices meeting (IEDM), San Francisco, 11–15 December 2021, pp. 3.6.1–3.6.4.

Bhandarkar D (1997) RISC versus CISC: a tale of two chips. *ACM SIGARCH Computer Architecture News* 25(1): 1–12.

Biswas A (2021) Sapphire rapids. In: 2021 IEEE hot chips 33 symposium (HCS), Palo Alto, CA, 22–24 August 2021, pp. 1–22.

Blagodurov S, Zhuravlev S and Fedorova A (2010) Contention-aware scheduling on multicore systems. *ACM Transactions on Computer Systems* 28(4): 1–45.

Blake G, Dreslinski RG and Mudge T (2009) A survey of multicore processors. *IEEE Signal Processing Magazine* 26(6): 26–37.

Blake G, Dreslinski RG, Mudge T, et al. (2010) Evolution of thread-level parallelism in desktop applications. *ACM SIGARCH Computer Architecture News* 38(3): 302–313.

Blem E, Menon J and Sankaralingam K (2013) Power struggles: revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures. In: 2013 IEEE 19th international symposium on high performance computer architecture (HPCA), Shenzhen, China, 23–27 February 2013, pp. 1–12.

Blythe D (2008) Rise of the graphics processor. *Proceedings of the IEEE* 96(5): 761–778.

Bogaerts W and Chrostowski L (2018) Silicon photonics circuit design: methods, tools and challenges. *Laser & Photonics Reviews* 12(4): 1700237.

Bogatin E (2011) Essential principles of signal integrity. *IEEE Microwave Magazine* 12(5): 34–41.

Bogatin E (2022) What's new in signal integrity and high-speed serial links: approaching the fundamental limits of copper interconnects. *IEEE Microwave Magazine* 23(5): 84–95.

Boggs D, Baktha A, Hawkins J, et al. (2004) The microarchitecture of the intel pentium 4 processor on 90nm technology. *Intel Technology Journal* 8(1): 7–23.

Bohl S (2022) Graphcore IPUs: accelerating Argonne's AI/ML applications. In: Argonne training program on extreme-scale computing (ATPESC) 2022.

Bohr M (2007) A 30 year retrospective on Dennard's MOSFET scaling paper. *IEEE Solid-State Circuits Newsletter* 12(1): 11–13.

Bonilla G, Lanzillo N, Hu CK, et al. (2020) Interconnect scaling challenges, and opportunities to enable system-level performance beyond 30 nm pitch. In: 2020 IEEE international electron devices meeting (IEDM), San Francisco, 12–18 December 2020, pp. 20.4.1–20.4.4.

Borkar S (2007) Thousand core chips: a technology perspective. In: Proceedings of the 44th annual design automation conference, DAC '07. New York, NY: Association for Computing Machinery, pp. 746–749.

Boutros A, Yazdanshenas S and Betz V (2018) You cannot improve what you do not measure: FPGA vs. ASIC efficiency gaps for convolutional neural network inference. *ACM Transactions on Reconfigurable Technology and Systems* 11(3): 1–23.

Boutros A, Nurvitadhi E, Ma R, et al. (2020) Beyond peak performance: comparing the real performance of AI-optimized FPGAs and GPUs. In: 2020 International conference on field-programmable technology (ICFPT), Maui, HI, 9–11 December 2020, pp. 10–19.

Boutros A, Arora A and Betz V (2024) Field-programmable gate array architecture for deep learning: survey & future directions. arXiv preprint arXiv:2404.10076.

Brekelbaum E, Rupley J, Wilkerson C, et al. (2002) Hierarchical scheduling windows. In: 35th Annual IEEE/ACM international symposium on microarchitecture, 2002. (MICRO-35). Proceedings, Istanbul, Turkey, 18–22 November 2002, pp. 27–36.

Brooks D, Bose P, Schuster S, et al. (2000) Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors. *IEEE Micro* 20(6): 26–44.

Brooks D, Dick RP, Joseph R, et al. (2007) Power, thermal, and reliability modeling in nanometer-scale microprocessors. *IEEE Micro* 27(3): 49–62.

Buchanan B (1999) *Fast Ethernet and Switches*. Chapter 39. Springer US, pp. 513–521.

Buck I (2007) GPU computing with NVIDIA CUDA. In: ACM SIGGRAPH 2007 courses, SIGGRAPH '07. New York, NY: Association for Computing Machinery, pp. 6–es.

Burd T, Beck N, White S, et al. (2019) "Zeppelin": an SoC for multichip architectures. *IEEE Journal of Solid-State Circuits* 54(1): 133–143.

Burd T, Li W, Pistole J, et al. (2022) Zen3: the AMD 2nd-generation 7nm x86-64 microprocessor core. In: 2022 IEEE international solid- state circuits conference (ISSCC), San Francisco, CA, 20–26 February 2022, pp. 1–3.

Burger D, Goodman JR and Kägi A (1996) Memory bandwidth limitations of future microprocessors. In: Proceedings of the 23rd annual international symposium on computer architecture, ISCA '96. New York, NY: Association for Computing Machinery, pp. 78–89.

Butcher N, Olivier SL, Berry J, et al. (2018) Optimizing for KNL usage modes when data doesn't fit in MCDRAM. In: Proceedings of the 47th international conference on parallel processing, ICPP '18. New York, NY: Association for Computing Machinery, pp. 1–10.

Butts J and Sohi G (2000) A static power model for architects. In: Proceedings 33rd annual IEEE/ACM international symposium on microarchitecture. MICRO-33 2000, Monterey CA, 10–13 December 2000. pp. 191–201.

Cadien KC, Reshotko MR, Block BA, et al. (2005) Challenges for on-chip optical interconnects. In: Kubby JA and Jabbour GE (eds) *Optoelectronic Integration on Silicon II*: International Society for Optics and Photonics, SPIE, Vol. 5730, 133–143.

Cai Y, Fang L, Ratemo R, et al. (2005) A test case for 3Gbps serial attached SCSI (SAS). In: IEEE international conference on test, 2005, Austin, TX, 8–10 November 2005, pp. 9–660.

Cairncross A, Henry B, Chalmers C, et al. (2023) *AI Benchmarking on Achronix Speedster® 7t FPGAs*. White Paper.

Calahan D and Ames W (1979) Vector processors: models and applications. *IEEE Transactions on Circuits and Systems* 26(9): 715–726.

Carter NP, Agrawal A, Borkar S, et al. (2013) Runnemede: an architecture for ubiquitous high-performance computing. In: 2013 IEEE 19th international symposium on high performance computer architecture (HPCA), Shenzhen, China, 23–27 February 2013, pp. 198–209.

Cascaval C, Chatterjee S, Franke H, et al. (2010) A taxonomy of accelerator architectures and their programming models. *IBM Journal of Research and Development* 54(5): 5:1–5:10.

Cass S (2020) Nvidia makes it easy to embed AI: the Jetson nano packs a lot of machine-learning power into DIY projects - [Hands on]. *IEEE Spectrum* 57(7): 14–16.

Cerebras Systems (2019) Wafer-scale deep learning. In: 2019 IEEE hot chips 31 symposium (HCS), Cupertino, CA, 18–20 August 2019, pp. 1–31.

Chang PY, Hao E, Patt YN, et al. (1996) Using predicated execution to improve the performance of a dynamically scheduled machine with speculative execution. *International Journal of Parallel Programming* 24(3): 209–234.

Chang CS, Oscilowski A and Bracken R (1998) Future challenges in electronics packaging. *IEEE Circuits and Devices Magazine* 14(2): 45–54.

Chang L, Tang S, King TJ, et al. (2000) Gate length scaling and threshold voltage control of double-gate MOSFETs. In: International Electron Devices Meeting 2000. Technical digest. IEDM (Cat. No.00CH37138), pp. 719–722.

Charitopoulos G, Papaefstathiou I and Pnevmatikatos DN (2021) Creating customized CGRAs for scientific applications. *Electronics* 10(4): 445.

Che D and Chen X (2023) Modulation format and digital signal processing for IM-DD optics at Post-200G era. *Journal of Lightwave Technology* 42(2): 588–605.

Chelton WN and Benaissa M (2008) Fast elliptic curve cryptography on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 16(2): 198–205.

Chen JC and Chien SY (2008) CRISP: coarse-grained reconfigurable image stream processor for digital still cameras and camcorders. *IEEE Transactions on Circuits and Systems for Video Technology* 18(9): 1223–1236.

Chen Z and Katopis G (2004) A comparison of performance potentials of single ended vs. differential signaling. In: Electrical performance of electronic packaging - 2004, pp. 185–188.

Chen Z and Segev M (2021) Highlighting photonics: looking into the next decade. *eLight* 1(1): 2.

Chen MK, Tai CC, Huang YJ, et al. (2002) Electrical characterization of BGA test socket for high-speed applications. In: Proceedings of the 4th international symposium on electronic materials and packaging 2002, Kaohsiung, Taiwan, 4–6 December 2002, pp. 123–126.

Chen J, Gordon MI, Thies W, et al. (2005) A reconfigurable architecture for load-balanced rendering. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware, HWWS '05. New York, NY: Association for Computing Machinery, pp. 71–80.

Chen G, Chen H, Haurylau M, et al. (2006) On-chip copper-based vs. optical interconnects: delay uncertainty, latency, power, and bandwidth density comparative predictions. In: 2006 International interconnect technology conference, Burlingame, CA, 5–7 June 2006, pp. 39–41.

Chen GY, Jensen B, Stolojan V, et al. (2011) Growth of carbon nanotubes at temperatures compatible with integrated circuit technologies. *Carbon* 49(1): 280–285.

Chen S, Irving S, Peng L, et al. (2017) Using switchable pins to increase off-chip bandwidth in chip-multiprocessors. *IEEE Transactions on Parallel and Distributed Systems* 28(1): 274–289.

Chen S, Hu W and Li Z (2019a) High performance data encryption with AES implementation on FPGA. In: 2019 IEEE 5th intl conference on big data security on cloud (BigDataSecurity), IEEE intl conference on high performance and smart computing, (HPSC) and IEEE intl conference on intelligent data and security (IDS), pp. 149–153.

Chen Y, He J, Zhang X, et al. (2019b) Cloud-DNN: an open framework for mapping DNN models to cloud FPGAs. In: Proceedings of the 2019 ACM/SIGDA international symposium on field-programmable gate arrays, FPGA '19. New York, NY: Association for Computing Machinery, pp. 73–82.

Chen SS, Tajali A, Holden B, et al. (2023) Noise performance comparison: ENRZ, NRZ, PAM3, and PAM4. In: 2023 IEEE symposium on electromagnetic compatibility & signal/power integrity (EMC+SIPI), Grand Rapids, MI, 31 July–4 August 2023, pp. 275–279.

Cheng W, Tan Z, Gao X, et al. (2008) High speed serial interface & some key technology research. In: 2008 International symposium on electronic commerce and security, Guangzhou, China, 3–5 August 2008, pp. 562–566.

Cheng H, Gao J, Wu HC, et al. (2016) Optics vs. copper — from the perspective of Thunderbolt 3 interconnect technology. In: 2016 China semiconductor technology international conference (CSTIC), Shanghai, China, 13–14 March 2016, pp. 1–3.

Chin SA, Sakamoto N, Rui A, et al. (2017) CGRA-ME: a unified framework for CGRA modelling and exploration. In: 2017 IEEE 28th international conference on application-specific systems, architectures and processors (ASAP), Seattle, WA, 10–12 July 2017, pp. 184–189.

Chirkov G and Wentzlaff D (2023) Seizing the bandwidth scaling of on-package interconnect in a post-Moore's law world. In: Proceedings of the 37th ACM international conference on supercomputing, ICS '23. New York, NY: Association for Computing Machinery, pp. 410–422.

Cho H, Koo KH, Kapur P, et al. (2007) The delay, energy, and bandwidth comparisons between copper, carbon nanotube, and optical interconnects for local and global wiring application. In: 2007 IEEE international interconnect technology conference, Burlingame, CA, 3–6 June 2007, pp. 135–137.

Cho H, Koo KH, Kapur P, et al. (2008) Performance comparisons between Cu/Low-κ, carbon-nanotube, and optics for future on-chip interconnects. *IEEE Electron Device Letters* 29(1): 122–124.

Cho SJ, Ahn J, Choi H, et al. (2012) Performance analysis of multi-bank DRAM with increased clock frequency. In: 2012 IEEE international symposium on circuits and systems (ISCAS), Seoul, South Korea, 20–23 May 2012, pp. 2477–2480.

Choi K, Soma R and Pedram M (2004) Dynamic voltage and frequency scaling based on workload decomposition. In: Proceedings of the 2004 international symposium on low power electronics and design, ISLPED '04. New York, NY: Association for Computing Machinery, pp. 174–179.

Chong FT, Heck MJR, Ranganathan P, et al. (2014) Data center energy efficiency:improving energy efficiency in data centers beyond technology scaling. *IEEE Design & Test* 31(1): 93–104.

Choquette J, Gandhi W, Giroux O, et al. (2021) NVIDIA A100 tensor core GPU: performance and innovation. *IEEE Micro* 41(2): 29–35.

Chow EM, Chua C, Hantschel T, et al. (2006) Pressure contact micro-springs in small pitch flip-chip packages. *IEEE Transactions on Components and Packaging Technologies* 29(4): 796–803.

Chrysos G (2012) Intel® Xeon Phi coprocessor (codename Knights Corner). In: 2012 IEEE hot chips 24 symposium (HCS), Cupertino, CA, 27–29 August 2012, pp. 1–31.

Cideciyan RD, Gustlin M, Li MP, et al. (2013) Next generation backplane and copper cable challenges. *IEEE Communications Magazine* 51(12): 130–136.

Ciofi I, Contino A, Roussel PJ, et al. (2016) Impact of wire geometry on interconnect RC and circuit delay. *IEEE Transactions on Electron Devices* 63(6): 2488–2496.

Cochran R, Hankendi C, Coskun AK, et al. (2011) Pack & cap: adaptive DVFS and thread packing under power caps. In: Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture, MICRO-44. New York, NY: Association for Computing Machinery, pp. 175–185.

Cong J, Huang M, Pan P, et al. (2016) Software infrastructure for enabling FPGA-based accelerations in data centers: invited paper. In: Proceedings of the 2016 international symposium on low power electronics and design, ISLPED '16. New York, NY: Association for Computing Machinery, pp. 154–155.

Corbin JS, Ramirez CN and Massey DE (2002) Land grid array sockets for server applications. *IBM Journal of Research and Development* 46(6): 763–778.

Couch LW (1994) *Modern Communication Systems*. Macmillan.

Crawford J (1990) The i486 CPU: executing instructions in one clock cycle. *IEEE Micro* 10(1): 27–36.

Cristal A, Santana O, Cazorla F, et al. (2005) Kilo-instruction processors: overcoming the memory wall. *IEEE Micro* 25(3): 48–57.

Cui Y, Ingalz C, Gao T, et al. (2017) Total cost of ownership model for data center technology evaluation. In: 2017 16th IEEE intersociety conference on thermal and thermomechanical phenomena in electronic systems (ITherm), Orlando, FL, 30 May–2 June 2017, pp. 936–942.

Cunningham D (2001) The status of the 10-Gigabit ethernet standard. In: Proceedings 27th European conference on optical communication (Cat. No.01TH8551), Vol. 3, pp. 364–367.

Dai K, Wang X, Niu S, et al. (2017) Simulation and structure optimization of triboelectric nanogenerators considering the effects of parasitic capacitance. *Nano Research* 10(1): 157–171.

Dally WJ, Turakhia Y and Han S (2020) Domain-specific hardware accelerators. *Communications of the ACM* 63(7): 48–57.

Danowitz A, Kelley K, Mao J, et al. (2012a) CPU DB: recording microprocessor history. *Communications of the ACM* 55(4): 55–63.

Danowitz A, Kelley K, Mao J, et al. (2012b) CPU DB: recording microprocessor history: with this open database, you can mine microprocessor trends over the past 40 years. *Queue* 10(4): 10–27.

Davidson J and Jinturkar S (1995) Improving instruction-level parallelism by loop unrolling and dynamic memory disambiguation. In: Proceedings of the 28th annual international symposium on microarchitecture, Ann Arbor, MI, 29 November–1 December 1995, pp. 125–132.

Dawoud DS and Dawoud P (2020) *6 Serial Peripheral Interface (SPI)*. Chapter 1. River Publishers, pp. 1–44.

DeBenedictis EP (2017) It's time to redefine Moore's law again. *Computer* 50(2): 72–75.

Deepaksubramanyan BS and Nunez A (2007) Analysis of subthreshold leakage reduction in CMOS digital circuits. In: 2007 50th midwest symposium on circuits and systems, Montreal Canada, 5–8 August 2007, pp. 1400–1404.

Deering M, Winner S, Schediwy B, et al. (1988) The triangle processor and normal vector shader: a VLSI system for high performance graphics. *ACM SIGGRAPH Computer Graphics* 22(4): 21–30.

Dennard R, Gaensslen F, Yu HN, et al. (1974) Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9(5): 256–268.

Dikhaminjia N, He J, Tsiklauri M, et al. (2016) PAM4 signaling considerations for high-speed serial links. In: 2016 IEEE international symposium on electromagnetic compatibility (EMC), Ottawa, ON, 25–29 July 2016, pp. 906–910.

Domke J, Matsumura K, Wahib M, et al. (2019) Double-precision FPUs in high-performance computing: an embarrassment of riches? In: 2019 IEEE international parallel and distributed processing symposium (IPDPS), Rio de Janeiro, Brazil, 20–24 May 2019, pp. 78–88.

Duan C, Tirumala A and Khatri S (2001) Analysis and avoidance of cross-talk in on-chip buses. In: HOT 9 interconnects. symposium on high performance interconnects, Stanford, CA, 22–24 August 2001, pp. 133–138.

Dubey R (2009) *Hardware Description Language: Verilog*. Chapter 2. Springer, pp. 17–51.

Dubrow A (2015) What got done in one year at NSF's stampede supercomputer. *Computing in Science & Engineering* 17(02): 83–88.

Duesberg G, Graham A, Kreupl F, et al. (2004) Ways towards the scaleable integration of carbon nanotubes into silicon based technology. *Diamond and Related Materials* 13(2): 354–361, Carbon materials for active electronics. proceedings of symposium L, E-MRS spring meeting 2003.

Dujmovic JJ and Dujmovic I (1998) Evolution and evaluation of SPEC benchmarks. *ACM SIGMETRICS Performance Evaluation Review* 26(3): 2–9.

Duran A and Klemm M (2012) The Intel® many integrated core architecture. In: 2012 International conference on high performance computing & simulation (HPCS), Madrid, Spain, 2–6 July 2012, pp. 365–366.

Dysart T, Moore B, Schaelicke L, et al. (2004) Cache implications of aggressively pipelined high performance microprocessors. In: IEEE international symposium on - ISPASS performance analysis of systems and software, 2004, Austin, TX, 10–12 March 2004, pp. 123–132.

D'Arnese E, Conficconi D, Santambrogio MD, et al. (2023) *Reconfigurable Architectures: The Shift from General Systems to Domain Specific Solutions*. Chapter 5. Springer Nature Singapore, pp. 435–456.

Eddington C (2002) InfiniBridge: an InfiniBand channel adapter with integrated switch. *IEEE Micro* 22(2): 48–56.

Edelstein DC, Sai-Halasz GA and Mii YJ (1995) VLSI on-chip interconnection performance simulations and measurements. *IBM Journal of Research and Development* 39(4): 383–401.

Eggers S, Emer J, Levy H, et al. (1997) Simultaneous multithreading: a platform for next-generation processors. *IEEE Micro* 17(5): 12–19.

Eidson S, Gaines B and Wolf P (2003) 30.2: HDMI: high-definition multimedia interface. *SID Symposium Digest of Technical Papers* 34(1): 1024–1027.

Elgharbawy W and Bayoumi M (2005) Leakage sources and possible solutions in nanometer CMOS technologies. *IEEE Circuits and Systems Magazine* 5(4): 6–17.

Elliott C (2004) Programming graphics processors functionally. In: Proceedings of the 2004 ACM SIGPLAN workshop on Haskell, Haskell '04. New York, NY: Association for Computing Machinery, pp. 45–56.

Emani M, Vishwanath V, Adams C, et al. (2021) Accelerating scientific applications with SambaNova reconfigurable dataflow architecture. *Computing in Science & Engineering* 23(2): 114–119.

Emma PG (1997) Understanding some simple processor-performance limits. *IBM Journal of Research and Development* 41(3): 215–232.

Esmaeilzadeh H, Blem E, St Amant R, et al. (2011a) Dark silicon and the end of multicore scaling. *SIGARCH Computer Architecture News* 39(3): 365–376.

Esmaeilzadeh H, Blem E, St Amant R, et al. (2011b) Dark silicon and the end of multicore scaling. In: Proceedings of the 38th annual international symposium on computer architecture, ISCA '11. New York, NY: Association for Computing Machinery, pp. 365–376.

Espasa R and Valero M (1997) Exploiting instruction- and data-level parallelism. *IEEE Micro* 17(5): 20–27.

Evans J (2022) Nvidia grace. In: 2022 IEEE hot chips 34 symposium (HCS). Los Alamitos, CA: IEEE Computer Society, pp. 1–20.

Evers M, Barnes L and Clark M (2022) The AMD next-generation "Zen 3" core. *IEEE Micro* 42(3): 7–12.

Eyerman S, Eeckhout L, Karkhanis T, et al. (2009) A mechanistic performance model for superscalar out-of-order processors. *ACM Transactions on Computer Systems* 27(2): 1–37.

Fair I, Grover W, Krzymien W, et al. (1991) Guided scrambling: a new line coding technique for high bit rate fiber optic transmission systems. *IEEE Transactions on Communications* 39(2): 289–297.

Fang J, Sips H, Zhang L, et al. (2014) Test-driving Intel Xeon Phi. In: Proceedings of the 5th ACM/SPEC international conference on performance engineering, ICPE '14. New York, NY: Association for Computing Machinery, pp. 137–148.

Farrell S, Calafiura P, Leggett C, et al. (2017) Multi-threaded ATLAS simulation on Intel Knights Landing processors. *Journal of Physics: Conference Series* 898(4): 042012.

Fawcett B (1995) Designing PCI bus interfaces with programmable logic. In: Proceedings of eighth international application specific integrated circuits conference, 321–324.

Firoozshahian A, Coburn J, Levenstein R, et al. (2023) MTIA: first generation silicon targeting meta's recommendation systems. In: Proceedings of the 50th annual international symposium on computer architecture, ISCA '23. New York, NY: Association for Computing Machinery, pp. 1–13.

Flynn M (1966) Very high-speed computing systems. *Proceedings of the IEEE* 54(12): 1901–1909.

Flynn MJ (1972) Some computer organizations and their effectiveness. *IEEE Transactions on Computers* 21(9): 948–960.

Flynn MJ and Rudd KW (1996) Parallel architectures. *ACM Computing Surveys* 28(1): 67–70.

Forghani M and Razavi B (2022) Circuit bandwidth requirements for NRZ and PAM4 signals. In: 2022 IEEE international symposium on circuits and systems (ISCAS), Austin, TX, 27 May–1 June 2022, pp. 990–994.

Foss R (1997) Taking DRAM from 4 MBytes/s to 4 GBytes/s. In: Proceedings of the 23rd European solid-state circuits conference, Southampton, UK, 16–18 September 1997, p. 2.

Frazier H (1998) The 802.3z gigabit ethernet standard. *IEEE Network* 12(3): 6–7.

Frenzel L (2007) *Principles of Electronic Communication Systems*. 3 edition. McGraw-Hill, Inc.

Friedman E (2001) Clock distribution networks in synchronous digital integrated circuits. *Proceedings of the IEEE* 89(5): 665–692.

Fritz F (2019a) Amd 12nm zen+ pinnacle ridge die shot.

Fritz F (2019b) Amd 7nm ccd and 12nm iod zen2 rome epyc 7702 die shot.

Fritz F (2020) Amd 7nm ccd and 12nm iod zen3 vermeer die shot.

Fu X, Zhang Z, Fan H, et al. (2024) Distributed training of large language lodels on AWS trainium. In: Proceedings of the 2024 ACM symposium on cloud computing, Redmond, WA, 20–22 November 2024, pp. 961–976.

Fujiki D, Wang X, Subramaniyan A, et al. (2021) *Domain-Specific Accelerators*. Chapter 6. Springer International Publishing, pp. 61–84.

Fujimori I (2014) Evolution of multi-gigabit wireline transceivers in CMOS. In: 2014 IEEE compound semiconductor integrated circuit symposium (CSICS), pp. 1–4.

Fung WWL and Aamodt TM (2011) Thread block compaction for efficient SIMT control flow. In: 2011 IEEE 17th international symposium on high performance computer architecture, San Antonio, TX, 12–16 February 2011, pp. 25–36.

Gallet B and Gowanlock M (2022) Leveraging GPU tensor cores for double precision Euclidean distance calculations. In: 2022 IEEE 29th international conference on high performance computing, data, and analytics (HiPC). Los Alamitos, CA: IEEE Computer Society, pp. 135–144.

Ganapathy D and Warner EJ (2008) Defining thermal design power based on real-world usage models. In: 2008 11th Intersociety conference on thermal and thermomechanical phenomena in electronic systems, Orlando, FL, 28–31 May 2008, pp. 1242–1246.

Gandhare S and Karthikeyan B (2019) Survey on FPGA architecture and recent applications. In: 2019 International conference on vision towards emerging trends in communication and networking (Vitecon), pp. 1–4.

Gandhi A, Akkary H and Srinivasan S (2004) Reducing branch misprediction penalty via selective branch recovery. In: 10th International symposium on high performance computer architecture (HPCA'04), pp. 254–264.

Gao M and Kozyrakis C (2016) HRL: efficient and flexible reconfigurable logic for near-data processing. In: 2016 IEEE international symposium on high performance computer architecture (HPCA), pp. 126–137.

Gao Z, Zhou B, Li Y, et al. (2020) Design and implementation of an On-Chip low-power and high-flexibility system for data acquisition and processing of an inertial measurement unit. *Sensors* 20(2): 462.

Garcia JC, Montiel-Nelson JA and Nooshabadi S (2007) Adaptive low/high voltage swing CMOS driver for on-chip interconnects. In: 2007 IEEE International symposium on circuits and systems (ISCAS), pp. 881–884.

Gargini P (2002) The global route to future semiconductor technology. *IEEE Circuits and Devices Magazine* 18(2): 13–17.

Gargini PA (2017) How to successfully overcome inflection points, or long live Moore's law. *Computing in Science & Engineering* 19(2): 51–62.

Geer D (2005) Chip makers turn to multicore processors. *Computer* 38(5): 11–13.

Gelatos A, Jain A, Marsh R, et al. (1994) Chemical vapor deposition of copper for advanced on-chip interconnects. *MRS Bulletin* 19(8): 49–54.

Gelatos AV, Nguyen BY, Perry KA, et al. (1996) Copper metallization for on-chip interconnects. In: Chen IC, Sasaki N, Patel DN, et al. (eds) *Microelectronic Device and Multilevel Interconnection Technology II*. International Society for Optics and Photonics, SPIE, Vol. 2875, pp. 346–357.

Gelsinger P (2001) Microprocessors for the new millennium: challenges, opportunities, and new frontiers. In: 2001 IEEE international solid-state circuits conference. Digest of technical papers. ISSCC (Cat. No.01CH37177), pp. 22–25.

Gelsinger P (2022) Semiconductors run the world: hot chips 2022. In: 2022 IEEE hot chips 34 symposium (HCS), pp. 1–19.

Geng T, Wu C, Tan C, et al. (2020) CQNN: a CGRA-based QNN framework. In: 2020 IEEE high performance extreme computing conference (HPEC), pp. 1–7.

Georganas E, Avancha S, Banerjee K, et al. (2018) Anatomy of high-performance deep learning convolutions on SIMD architectures. In: SC18: International conference for high performance computing, networking, storage and analysis, pp. 830–841.

Gepner P and Kowalik M (2006) Multi-core processors: new way to achieve high system performance. In: International symposium on parallel computing in electrical engineering (PARELEC'06), pp. 9–13.

Geppert L (2002) The amazing vanishing transistor act. *IEEE Spectrum* 39(10): 28–33.

Ghosal B, Sigliano R and Kunimatsu Y (2001) *Ceramic and Plastic Pin Grid Array Technology*. Chapter 14. Springer US, pp. 551–576.

Giles MB and Reguly I (2014) Trends in high-performance computing for engineering calculations. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences* 372(2022): 20130319.

Gindin R, Cidon I and Keidar I (2007) NoC-Based FPGA: architecture and routing. In: First international symposium on networks-on-chip (NOCS'07), pp. 253–264.

Giuma T and Hart K (1996) Microcomputer bus architectures. In: Southcon/96 conference record, pp. 431–437.

Glaskowsky P (1997) *3D Chips Break Megatriangle Barrier: Better Designs and Processes Help Crank Out 1M Polygons/s in Mainstream PCs*. Technical report, Microprocessor Report.

Gohel T (2012) The practical realities of high-speed digital test in a production environment. In: 2012 IEEE AUTOTESTCON proceedings, pp. 272–277.

Gomes W, Khushu S, Ingerly DB, et al. (2020) 8.1 lakefield and mobility compute: a 3D stacked 10nm and 22FFL hybrid processor system in 12× 12mm2, 1mm package-on-package. In: 2020 IEEE international solid- state circuits conference - (ISSCC), pp. 144–146.

Gonzalez R and Horowitz M (1996) Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits* 31(9): 1277–1284.

Goodnight N, Wang R and Humphreys G (2005) Computation on programmable graphics hardware. *IEEE Computer Graphics and Applications* 25(5): 12–15.

Greig WJ (2007) *Packaging the IC—Single Chip Packaging*. Chapter 3. Springer US, pp. 31–45.

Grelck C and Sarris N (2017) Towards compiling SAC for the Xeon Phi Knights Corner and Knights Landing architectures: strategies and experiments. In: Proceedings of the 29th symposium on the implementation and application of functional programming languages, IFL '17. New York, NY: Association for Computing Machinery, pp. 1–12.

Gribok S and Pasca B (2024) Efficient 8-bit matrix multiplication on Intel Agilex-5 FPGAs. In: 2024 IEEE 32nd annual international symposium on field-programmable custom computing machines (FCCM), pp. 43–53.

Grimsrud K and Smith H (2003) *Serial ATA Storage Architecture and Applications: Designing High-Performance, Cost-Effective I/O Solutions*. Intel Press.

Groeneveld P, James M, Kibardin V, et al. (2021) ISPD 2021 wafer-scale physics modeling contest: a new frontier for partitioning, placement and routing. In: Proceedings of the 2021 international symposium on physical design, ISPD '21. New York, NY: Association for Computing Machinery, pp. 143–147.

Gronowski P, Bowhill W, Preston R, et al. (1998) High-performance microprocessor design. *IEEE Journal of Solid-State Circuits* 33(5): 676–686.

Guermouche A and Orgerie AC (2022) Thermal design power and vectorized instructions behavior. *Concurrency and Computation: Practice and Experience* 34(2): e6261.

Guri M, Monitz M, Mirski Y, et al. (2015) BitWhisper: covert signaling channel between air-gapped computers using thermal manipulations. In: 2015 IEEE 28th computer security foundations symposium, pp. 276–289.

Gurrum S, Suman S, Joshi Y, et al. (2004) Thermal issues in next-generation integrated circuits. *IEEE Transactions on Device and Materials Reliability* 4(4): 709–714.

Habermaier A and Knapp A (2012) On the correctness of the SIMT execution model of GPUs. In: Seidl H (ed) *Programming Languages and Systems*. Springer Berlin Heidelberg, pp. 316–335.

Hahn PO (2001) The 300 mm silicon wafer — A cost and technology challenge. *Microelectronic Engineering* 56(1): 3–13, sub-Quarter-Micron Silicon Issues in the 200/300 mm Conversion Era.

Halawani Y and Mohammad B (2024) *Data-Centric Computing Paradigm Shift, and Domain-specific Architecture and Hardware*. Chapter 1. Springer Nature Switzerland, pp. 1–6.

Halvorsen OH and Clarke D (2011) *Universal Serial Bus*. Chapter 8. Apress, pp. 141–172.

Hameed R, Qadeer W, Wachs M, et al. (2010a) Understanding sources of inefficiency in general-purpose chips. *SIGARCH Computer Architecture News* 38(3): 37–47.

Hameed R, Qadeer W, Wachs M, et al. (2010b) Understanding sources of inefficiency in general-purpose chips. In: Proceedings of the 37th annual international symposium on computer architecture, ISCA '10. New York, NY: Association for Computing Machinery, pp. 37–47.

Han SC, Yoo SK, Park SW, et al. (1996) An ASIC implementation of the MPEG-2 audio decoder. *IEEE Transactions on Consumer Electronics* 42(3): 540–545.

Hanindhito B, Fathi A, Gourounas D, et al. (2026) Technology trends in computing hardware and their impacts on high-performance scientific computing Part II: memory systems, interconnects, and system integration. *The International Journal of High Performance Computing Applications*: 10943420251347461.

Hao Q, Qin M, Qi N, et al. (2021) A chip-level optical interconnect for CPU. *IEEE Photonics Technology Letters* 33(16): 852–855.

Haque MS, Ali SZ, Guha PK, et al. (2008) Growth of carbon nanotubes on fully processed silicon-on-insulator CMOS substrates. *Journal of Nanoscience and Nanotechnology* 8(11): 5667–5672.

Harrand M, Henry M, Chaisemartin P, et al. (1995) A single chip videophone video encoder/decoder. In: Proceedings ISSCC '95 - international solid-state circuits conference, pp. 292–293.

Hassaballah M, Omran S and Mahdy YB (2008) A review of SIMD multimedia extensions and their usage in scientific and engineering applications. *The Computer Journal* 51(6): 630–649.

Hawick KA and Playne DP (2014) Developmental directions in parallel accelerators. In: Proceedings of the twelfth Australasian symposium on parallel and distributed computing - Volume 152, AusPDC '14. AUS: Australian Computer Society, Inc., pp. 21–27.

He X, Chen Z, Sun J, et al. (2017) Exploring synchronization in cache coherent manycore systems: a case study with Xeon Phi. In: 2017 IEEE 23rd international conference on parallel and distributed systems (ICPADS), pp. 232–239.

Hecht U, Wittenhagen E, Cirit H, et al. (2022) PAM-4/6/8 performance and power analysis for next generation 224Gbit/s links. In: 2022 IEEE international symposium on circuits and systems (ISCAS), pp. 752–756.

Heinecke A, Klemm M and Bungartz HJ (2012) From GPGPU to many-core: Nvidia fermi and Intel many integrated core architecture. *Computing in Science & Engineering* 14(2): 78–83.

Hennessy JL (2021) The 50 year history of the microprocessor as five technology eras. *IEEE Micro* 41(6): 20–21.

Hennessy JL and Patterson DA (2019) A new golden age for computer architecture. *Communications of the ACM* 62(2): 48–60.

Henning J (2000) SPEC CPU2000: measuring CPU performance in the new millennium. *Computer* 33(7): 28–35.

Herbert S and Marculescu D (2007) Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In: Proceedings of the 2007 international symposium on low power electronics and design, ISLPED '07. New York, NY: Association for Computing Machinery, pp. 38–43.

Hill MD and Reddi VJ (2021) Accelerator-level parallelism. *Communications of the ACM* 64(12): 36–38.

Hitachi, Ltd (1997) Hitachi releases the 125 MHz 8M-Bit synchronous graphic RAM.

Ho R, Mai K and Horowitz M (2001) The future of wires. *Proceedings of the IEEE* 89(4): 490–504.

Holzinger P, Reiser D, Hahn T, et al. (2021) Fast HBM access with FPGAs: analysis, architectures, and applications. In: 2021 IEEE international parallel and distributed processing symposium workshops (IPDPSW), pp. 152–159.

Hood R, Jin H, Mehrotra P, et al. (2010) Performance impact of resource contention in multicore systems. In: 2010 IEEE international symposium on parallel & distributed processing (IPDPS), pp. 1–12.

Hoozemans J, Peltenburg J, Nonnemacher F, et al. (2021) FPGA acceleration for big data analytics: challenges and opportunities. *IEEE Circuits and Systems Magazine* 21(2): 30–47.

Horowitz M (2014) 1.1 Computing's energy problem (and what we can do about it). In: 2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC), pp. 10–14.

Horowitz M, Labonte F, Shacham O, et al. (2015) 35 Years of microprocessor trend data.

Hosseini R, Simini F, Vishwanath V, et al. (2023) Exploring the use of dataflow architectures for graph neural network workloads. In: Bienz A, Weiland M, Baboulin M, et al. (eds) *High Performance Computing*. Springer Nature Switzerland, pp. 648–661.

Hruska J (2018) As chip design costs skyrocket, 3nm process node is in Jeopardy.

Hu A and Yuan F (2009) Intersignal timing skew compensation of parallel links with voltage-mode incremental signaling. *IEEE Transactions on Circuits and Systems I: Regular Papers* 56(4): 773–783.

Hu C, Gignac LM, Liniger E, et al. (2009) Electromigration challenges for nanoscale Cu wiring. *AIP Conference Proceedings* 1143(1): 3–11.

Hu X, Stow D and Xie Y (2018) Die Stacking is happening. *IEEE Micro* 38(1): 22–28.

Hu Y, Du Y, Ustun E, et al. (2021) GraphLily: accelerating graph linear algebra on HBM-equipped FPGAs. In: 2021 IEEE/ACM international conference on computer aided design (ICCAD), pp. 1–9.

Huang CY, Yin YF, Hsu CJ, et al. (2011) SoC HW/SW verification and validation. In: 16th Asia and South Pacific design automation conference (ASP-DAC 2011), pp. 297–300.

Huang Y, Guo N, Seok M, et al. (2017) Hybrid analog-digital solution of nonlinear partial differential equations. In: Proceedings of the 50th annual IEEE/ACM international symposium on microarchitecture, MICRO-50 '17. New York, NY: Association for Computing Machinery, pp. 665–678.

Huang V, Shim D, Simka H, et al. (2020) From interconnect materials and processes to chip level performance: modeling and design for conventional and exploratory concepts. In: 2020 IEEE international electron devices meeting (IEDM), pp. 32.6.1–32.6.4.

Hwu WW and Patel S (2008) Guest editors' introduction: accelerator architectures. *IEEE Micro* 28(04): 4–12.

Hwu WM and Patel S (2018) Accelerator architectures —A ten-year retrospective. *IEEE Micro* 38(6): 56–62.

Ikarashi Y, Bernstein GL, Reinking A, et al. (2022) Exocompilation for productive programming of hardware accelerators. In: Proceedings of the 43rd ACM SIGPLAN international conference on programming language design and implementation, PLDI 2022. New York, NY: Association for Computing Machinery, pp. 703–718.

Imai H and Hirakawa S (1977) A new multilevel coding method using error-correcting codes. *IEEE Transactions on Information Theory* 23(3): 371–377.

Ingerly DB, Amin S, Aryasomayajula L, et al. (2019) Foveros: 3D integration and the use of face-to-face chip stacking for logic devices. In: 2019 IEEE international electron devices meeting (IEDM), pp. 19.6.1–19.6.4.

Iwai H (2009) Roadmap for 22nm and beyond (Invited paper). *Microelectronic Engineering* 86(7): 1520–1528, iNFOS 2009.

Iyer R (2012) Accelerator-rich architectures: implications, opportunities and challenges. In: 17th Asia and South Pacific design automation conference, pp. 106–107.

Iyer R, De V, Illikkal R, et al. (2021) Advances in microprocessor cache architectures over the last 25 years. *IEEE Micro* 41(6): 78–88.

Jagt J (1998) Reliability of electrically conductive adhesive joints for surface mount applications: a summary of the state of the art. *IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part A* 21(2): 215–225.

Jain S and Murugesan S (2021) *Smart Connected World: A Broader Perspective*. Chapter 1. Springer International Publishing, pp. 3–23.

Jeruchim M (1984) Techniques for estimating the bit error rate in the simulation of digital communication systems. *IEEE Journal on Selected Areas in Communications* 2(1): 153–170.

Jia Z, Tillman B, Maggioni M, et al. (2019) Dissecting the graphcore IPU architecture via microbenchmarking.

Jiang H (2022) Intel's Ponte Vecchio GPU: architecture, systems & software. In: 2022 IEEE hot chips 34 symposium (HCS). Los Alamitos, CA: IEEE Computer Society, pp. 1–29.

Jiménez V, Gioiosa R, Cazorla FJ, et al. (2012) Making data prefetch smarter: adaptive prefetching on POWER7. In: Proceedings of the 21st international conference on parallel architectures and compilation techniques, PACT '12. New York, NY: Association for Computing Machinery, pp. 137–146.

JM Veendrick H (2017) *Effects of Scaling on MOS IC Design and Consequences for the Roadmap*. Chapter 1. Springer International Publishing, pp. 573–594.

Jordà M, Valero-Lara P and Peña AJ (2019) Performance evaluation of cuDNN convolution algorithms on NVIDIA Volta GPUs. *IEEE Access* 7: 70461–70473.

Joshi A and Soni G (2016) A comparative analysis of copper and carbon nanotubes-based global interconnects in 32 nm technology. In: Pant M, Deep K, Bansal JC, et al. (eds) *Proceedings of Fifth International Conference on Soft Computing for Problem Solving*. Springer Singapore, pp. 425–437.

Jouppi NP and Wall DW (1989) Available instruction-level parallelism for superscalar and superpipelined machines. In: Proceedings of the third international conference on architectural support for programming languages and operating systems, ASPLOS III. New York, NY: Association for Computing Machinery, pp. 272–282.

Jouppi NP, Young C, Patil N, et al. (2017) In-Datacenter performance analysis of a tensor processing unit. *ACM SIGARCH Computer Architecture News* 45(2): 1–12.

Jouppi N, Young C, Patil N, et al. (2018) Motivation for and evaluation of the first tensor processing unit. *IEEE Micro* 38(3): 10–19.

Jouppi NP, Yoon DH, Kurian G, et al. (2020) A domain-specific supercomputer for training deep neural networks. *Communications of the ACM* 63(7): 67–78.

Jouppi NP, Hyun Yoon D, Ashcraft M, et al. (2021) Ten lessons from three generations shaped Google's TPUv4i: industrial product. In: 2021 ACM/IEEE 48th annual international symposium on computer architecture (ISCA). pp. 1–14.

Jouppi N, Kurian G, Li S, et al. (2023) TPU v4: an optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In: Proceedings of the 50th annual international symposium on computer architecture, ISCA '23. New York, NY: Association for Computing Machinery, pp. 1–14.

Juan T, Navarro JJ and Temam O (1997) Data caches for superscalar processors. In: Proceedings of the 11th international conference on supercomputing, ICS '97. New York, NY: Association for Computing Machinery, pp. 60–67.

Juan PS, Alonso-Jordá P and Quintana-Ortí ES (2021) High performance and energy efficient integer matrix multiplication for deep learning. In: 2021 29th Euromicro international conference on parallel, distributed and network-based processing (PDP). Institute of Electrical and Electronics Engineers, pp. 122–125.

Kao J, Narendra S and Chandrakasan A (2002) Subthreshold leakage modeling and reduction techniques. In: Proceedings of the 2002 IEEE/ACM international conference on computer-aided design, ICCAD '02. New York, NY: Association for Computing Machinery, pp. 141–148.

Kaplan R (2024) Intel Gaudi 3 AI accelerator: architected for Gen AI training and inference. In: 2024 IEEE hot chips 36 symposium (HCS). IEEE, pp. 1–16.

Kapur P and Saraswat K (2002) Comparisons between electrical and optical interconnects for on-chip signaling. In: Proceedings of the IEEE 2002 international interconnect technology conference (Cat. No.02EX519), pp. 89–91.

Kapur P, Chandra G, McVittie J, et al. (2002) Technology and reliability constrained future copper interconnects. II. Performance implications. *IEEE Transactions on Electron Devices* 49(4): 598–604.

Kara K, Hagleitner C, Diamantopoulos D, et al. (2020) High bandwidth memory on FPGAs: a data analytics perspective. In: 2020 30th international conference on field-programmable logic and applications (FPL), pp. 1–8.

Karabchevsky A, Katiyi A, Ang AS, et al. (2020) On-chip nanophotonics and future challenges. *Nanophotonics* 9(12): 3733–3753.

Karnik T and Hazucha P (2004) Characterization of soft errors caused by single event upsets in CMOS processes. *IEEE Transactions on Dependable and Secure Computing* 1(2): 128–143.

Karstensen H, Auracher F, Ebel N, et al. (2000) Module packaging for high-speed serial and parallel transmission. In: 2000 Proceedings. 50th Electronic components and technology conference (Cat. No.00CH37070), pp. 479–486.

Käsgen PS, Weinhardt M and Hochberger C (2018) A coarse-grained reconfigurable array for high-performance computing applications. In: 2018 International conference on reconfigurable computing and FPGAs (ReConFig), pp. 1–4.

Kaufman B (1999) Memory chip packaging: an enabling technology for high-performance recce systems. In: Fishell WG (ed) *Airborne Reconnaissance XXIII*. International Society for Optics and Photonics, SPIE, Vol. 3751, pp. 170–174.

Kaushik BK, Goel S and Rauthan G (2007) Future VLSI interconnects: optical fiber or carbon nanotube – A review. *Microelectronics International* 24(2): 53–63.

Kaushik BK, Majumder MK and Kumar VR (2014) Carbon nanotube based 3-D interconnects - a reality or a distant dream. *IEEE Circuits and Systems Magazine* 14(4): 16–35.

Kaushik P, Rao AM, Singh DP, et al. (2021) Cloud computing and comparison based on service and performance between Amazon AWS, Microsoft Azure, and Google Cloud. In: 2021 International conference on technological advancements and innovations (ICTAI), pp. 268–273.

Kaxiras S and Martonosi M (2008) *Optimizing Capacitance and Switching Activity to Reduce Dynamic Power*. Chapter 4. Springer International Publishing, pp. 45–129.

Keckler SW, Dally WJ, Maskit D, et al. (1998) Exploiting fine-grain thread level parallelism on the MIT Multi-ALU processor. In: Proceedings of the 25th annual international symposium on computer architecture, ISCA '98. USA: IEEE Computer Society, pp. 306–317.

Kelleher A (2022) Celebrating 75 years of the transistor A look at the evolution of Moore's law innovation. In: 2022 International electron devices meeting (IEDM), pp. 1.1.1–1.1.5.

Keutzer K, Malik S and Newton A (2002) From ASIC to ASIP: the next design discontinuity. In: Proceedings. IEEE international conference on computer design: VLSI in computers and processors, pp. 84–90.

Khaldi D, Luo Y, Yu B, et al. (2021) Extending LLVM IR for DPC++ matrix support: a case study with Intel® advanced matrix extensions (Intel® AMX). In: 2021 IEEE/ACM 7th workshop on the LLVM compiler infrastructure in HPC (LLVM-HPC), pp. 20–26.

Khan N, Rao VS, Lim S, et al. (2010) Development of 3-D silicon module with TSV for system in packaging. *IEEE Transactions on Components and Packaging Technologies* 33(1): 3–9.

Khazraee M, Zhang L, Vega L, et al. (2017) Moonwalk: NRE optimization in ASIC clouds. In: Proceedings of the twenty-second international conference on architectural support for programming languages and operating systems, ASPLOS '17. New York, NY: Association for Computing Machinery, pp. 511–526.

Killian Z (2023) AMD's Zen 4 I/O die shot reveals a fascinating Ryzen CCD detail.

Kim N, Austin T, Baauw D, et al. (2003) Leakage current: Moore's law meets static power. *Computer* 36(12): 68–75.

Kim S, Lee J, Yang J, et al. (2005) Novel instructions and their hardware architecture for video signal processing. In: 2005 IEEE international symposium on circuits and systems (ISCAS), Vol. 4, pp. 3323–3326.

Kim C, Song J and Lee HW (2014) *An I/O Line Configuration and Organization of DRAM*. Chapter 2. Springer International Publishing, pp. 13–24.

Kim S, Gholami A, Yao Z, et al. (2021) I-BERT: integer-only BERT quantization. In: Meila M and Zhang T (eds) Proceedings of the 38th international conference on machine learning, proceedings of machine learning research. PMLR, Vol. 139, pp. 5506–5518.

Kish LB (2002) End of Moore's law: thermal (noise) death of integration in micro and nano electronics. *Physics Letters A* 305(3): 144–149.

Kleveland B, Qi X, Madden L, et al. (2002) High-frequency characterization of on-chip digital interconnects. *IEEE Journal of Solid-State Circuits* 37(6): 716–725.

Knowles S (2021) Graphcore. In: 2021 IEEE hot chips 33 symposium (HCS), pp. 1–25.

Ko H (2022) High-speed serial link trend and technical challenge. In: 2022 IEEE Asian solid-state circuits conference (A-SSCC), pp. 1–3.

Kobbelt L and Botsch M (2004) A survey of point-based techniques in computer graphics. *Computers & Graphics* 28(6): 801–814.

Kobrinsky MJ, Block BA, Jun-Fei Z, et al. (2004) On-chip optical interconnects. *Intel Technology Journal* 8(2): 129–141.

Kocanda P and Kos A (2015) Static and dynamic energy losses vs. temperature in different CMOS technologies. In: 2015 22nd international conference mixed design of integrated circuits & systems (MIXDES), pp. 446–449.

Koeplinger D, Feldman M, Prabhakar R, et al. (2018) Spatial: a language and compiler for application accelerators. *ACM SIGPLAN Notices* 53(4): 296–311.

Kohli P, Sobczak M, Bowin J, et al. (2001) Advanced thermal interface materials for enhanced flip chip BGA. In: 2001 Proceedings. 51st Electronic components and technology conference (Cat. No.01CH37220), pp. 564–570.

Kolodzey J (1981) CRAY-1 computer technology. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology* 4(2): 181–186.

Koo KH, Cho H, Kapur P, et al. (2007) Performance comparisons between carbon nanotubes, optical, and Cu for future high-performance on-chip interconnect applications. *IEEE Transactions on Electron Devices* 54(12): 3206–3215.

Koul K, Melchert J, Sreedhar K, et al. (2023) AHA: an agile approach to the design of coarse-grained reconfigurable accelerators and compilers. *ACM Transactions on Embedded Computing Systems* 22(2): 1–34.

Koyanagi M, Kurino H, Lee KW, et al. (1998) Future system-on-silicon LSI chips. *IEEE Micro* 18(4): 17–22.

Krishnakumar A, Ogras U, Marculescu R, et al. (2023) Domain-specific architectures: research problems and promising approaches. *ACM Transactions on Embedded Computing Systems* 22(2): 1–26.

Küçük G, Güney İ and Ponomarev D (2013) *Instruction Scheduling in Microprocessors*. Chapter 2. Springer Berlin Heidelberg, pp. 39–60.

Kudo H, Takano T, Akazawa M, et al. (2021) High-speed, high-density, and highly-manufacturable Cu-filled through-glass-via channel (Cu bridge) for multi-chiplet systems. In: 2021 IEEE 71st electronic components and technology conference (ECTC), pp. 1031–1037.

Kumar K, Ramkumar K, Kaur A, et al. (2020) A survey on hardware implementation of cryptographic algorithms using field programmable gate array. In: 2020 IEEE

9th international conference on communication systems and network technologies (CSNT), pp. 189–194.

Kuon I and Rose J (2006) Measuring the gap between FPGAs and ASICs. In: Proceedings of the 2006 ACM/SIGDA 14th international symposium on field programmable gate arrays, FPGA '06. New York, NY: Association for Computing Machinery, pp. 21–30.

Lachance R, Lavoie H and Montanari A (1997) Corrosion/migration study of flip chip underfill and ceramic overcoating. In: 1997 Proceedings 47th electronic components and technology conference, pp. 885–889.

Lai JW (2021) Opportunity and challenge of Chiplet-based HPC and AIoT. In: 2021 international symposium on VLSI design, automation and test (VLSI-DAT), pp. 1–2.

Langhammer M and Pasca B (2015) Floating-point DSP block architecture for FPGAs. In: Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays, FPGA '15. New York, NY: Association for Computing Machinery, pp. 117–125.

Langhammer M, Nurvitadhi E, Pasca B, et al. (2021) Stratix 10 NX architecture and applications. In: The 2021 ACM/SIGDA international symposium on field-programmable gate arrays, FPGA '21. New York, NY: Association for Computing Machinery, pp. 57–67.

Lathi BP and Ding Z (2022) *Modern Digital and Analog Communication*. The Oxford Series in Electrical and Computer Engineering. 5 edition. Oxford University Press.

Lauterbach G (2021) The path to successful wafer-scale integration: the cerebras story. *IEEE Micro* 41(6): 52–57.

Law D, Dove D, D'Ambrosia J, et al. (2013) Evolution of ethernet standards in the IEEE 802.3 working group. *IEEE Communications Magazine* 51(8): 88–96.

Le Beux S, Li H, Nicolescu G, et al. (2014) Optical crossbars on chip, a comparative study based on worst-case losses. *Concurrency and Computation: Practice and Experience* 26(15): 2492–2503.

Le Sueur E and Heiser G (2010) Dynamic voltage and frequency scaling: the laws of diminishing returns. In: Proceedings of the 2010 international conference on power aware computing and systems, HotPower'10. USA: USENIX Association, pp. 1–8.

Lee E (1990) Programmable DSPs: a brief overview. *IEEE Micro* 10(5): 14–16.

Lee SY and Wu CJ (2014) Characterizing the latency hiding ability of GPUs. In: 2014 IEEE international symposium on performance analysis of systems and software (ISPASS), pp. 145–146.

Lee JW, Kim HJ, Jeong CS, et al. (2013) Skew compensation technique for source-synchronous parallel DRAM interface. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21(11): 2155–2159.

Lee WK, Seo H, Zhang Z, et al. (2022) TensorCrypto: high throughput acceleration of lattice-based cryptography using tensor core on GPU. *IEEE Access* 10: 20616–20632.

Leiserson CE, Thompson NC, Emer JS, et al. (2020) There's plenty of room at the top: what will drive computer performance after Moore's law? *Science* 368(6495): eaam9744.

Lenihan TG, Matthew L and Vardaman EJ (2013) Developments in 2.5D: the role of silicon interposers. In: 2013 IEEE 15th electronics packaging technology conference (EPTC 2013), pp. 53–55.

Leong PHW (2008) Recent trends in FPGA architectures and applications. In: 4th IEEE international symposium on electronic design, test and applications (Delta 2008), pp. 137–141.

Li A and Su S (2021) Accelerating binarized neural networks via bit-tensor-cores in turing GPUs. *IEEE Transactions on Parallel and Distributed Systems* 32(7): 1878–1891.

Li Y, Li D, Cui W, et al. (2011) Research based on OSI model. In: 2011 IEEE 3rd international conference on communication software and networks, pp. 554–557.

Li S, Reddy D and Jacob B (2018) A performance & power comparison of modern high-speed DRAM architectures. In: Proceedings of the international symposium on memory systems, MEMSYS '18. New York, NY: Association for Computing Machinery, pp. 341–353.

Li R, Antunes EF, Kalfon-Cohen E, et al. (2019) Low-temperature growth of carbon nanotubes catalyzed by sodium-based ingredients. *Angewandte Chemie* 58(27): 9204–9209.

Li T, Hou J, Yan J, et al. (2020) Chiplet heterogeneous integration technology—Status and challenges. *Electronics* 9(4): 670.

Li G, Xue J, Liu L, et al. (2021) Unleashing the low-precision computation potential of tensor cores on GPUs. In: 2021 IEEE/ACM international symposium on code generation and optimization (CGO), pp. 90–102.

Li N, Chen G, Ng DKT, et al. (2022) Integrated lasers on silicon at communication wavelength: a progress review. *Advanced Optical Materials* 10(23): 2201008.

Lidow A and Sheridan G (2003) Defining the future for microprocessor power delivery. In: Eighteenth annual IEEE applied power electronics conference and exposition, 2003. APEC '03, Vol. 1. pp. 3–9.

Lie S (2022) Cerebras architecture deep dive: first look inside the HW/SW co-design for deep learning: cerebras systems. In: 2022 IEEE hot chips 34 symposium (HCS), pp. 1–34.

Lie S (2024) Inside the cerebras wafer-scale cluster. *IEEE Micro* 44(3): 49–57.

Lin Z, Mantor M and Zhou H (2018) GPU performance vs. thread-level parallelism: scalability analysis and a novel way to improve TLP. *ACM Transactions on Architecture and Code Optimization* 15(1): 1–21.

Lin Y, Liang R, Li Y, et al. (2022) Mapping large scale finite element computing on to wafer-scale engines. In: 2022 27th Asia and South Pacific design automation conference (ASP-DAC), pp. 147–153.

Lindholm E, Kilgard MJ and Moreton H (2001) A user-programmable vertex engine. In: Proceedings of the 28th annual conference on computer graphics and interactive techniques, SIGGRAPH '01. New York, NY: Association for Computing Machinery, pp. 149–158.

Lindholm E, Nickolls J, Oberman S, et al. (2008) NVIDIA tesla: a unified graphics and computing architecture. *IEEE Micro* 28(2): 39–55.

Liu X and Liu Y (2017) Multi-functional serial communication interface design based on FPGA. In: 2017 3rd IEEE international conference on computer and communications (ICCC), pp. 758–761.

Liu D and Svensson C (1993) Trading speed for low power by choice of supply and threshold voltages. *IEEE Journal of Solid-State Circuits* 28(1): 10–17.

Liu D and Svensson C (1994) Power consumption estimation in CMOS VLSI chips. *IEEE Journal of Solid-State Circuits* 29(6): 663–670.

Liu L, Zhu J, Li Z, et al. (2019) A survey of coarse-grained reconfigurable architecture and design: taxonomy, challenges, and applications. *ACM Computing Surveys* 52(6): 1–39.

Loan A (2007) *Line Coding*. Chapter 34. John Wiley & Sons, Ltd., pp. 522–537.

LoCicero JL and Patel BP (2018) *The Communications Handbook*. CRC Press.

Locuza (2020) Zen 2 (+1) layman die shot analysis (tutorial style) - Part 6.

Locuza (2022) Zen evolution: a small overview.

Loghin D (2024) Are arm cloud servers ready for database workloads? An experimental study. *IEEE Transactions on Cloud Computing* 12(3): 818–829.

Loh GH, Naffziger S and Lepak K (2021) Understanding chiplets today to anticipate future integration opportunities and limits. In: 2021 design, automation and test in Europe conference & exhibition (DATE), pp. 142–145.

Loh GH, Schulte MJ, Ignatowski M, et al. (2023) A research retrospective on AMD's exascale computing journey. In: Proceedings of the 50th annual international symposium on computer architecture, ISCA '23. New York, NY: Association for Computing Machinery, pp. 1–14.

Luebke D and Humphreys G (2007) How GPUs work. *Computer* 40(2): 96–100.

Luo SY and Wang ZW (2008) Studies of chipping mechanisms for dicing silicon wafers. *The International Journal of Advanced Manufacturing Technology* 35(11): 1206–1218.

Luo C, Su Z and Lü Z (2023) MS-CLS: an effective partitioning and placement metaheuristic for wafer-scale physics modeling. *IEEE Transactions on Emerging Topics in Computational Intelligence* PP(99): 1–15.

Mack C (2015) The multiple lives of Moore's law. *IEEE Spectrum* 52(4): 31.

Macri J (2015) AMD's next generation GPU and high bandwidth memory architecture: FURY. In: 2015 IEEE hot chips 27 symposium (HCS), pp. 1–26.

Maddrell-Mander S, Mohan LRM, Marshall A, et al. (2021) Studying the potential of graphcore® IPUs for applications in particle physics. *Computing and Software for Big Science* 5(1): 8.

Madhow U (2008) *Fundamentals of Digital Communication*. Cambridge University Press.

Magaki I, Khazraee M, Gutierrez LV, et al. (2016) ASIC clouds: specializing the datacenter. In: 2016 ACM/IEEE 43rd annual international symposium on computer architecture (ISCA), pp. 178–190.

Magro W, Petersen P and Shah S (2002) Hyper-threading technology: impact on compute-intensive workloads. *Intel Technology Journal* 6(1): 1.

Mahajan R, Chiu C and Chrysler G (2006a) Cooling a microprocessor chip. *Proceedings of the IEEE* 94(8): 1476–1486.

Mahajan R, Mallik D, Sankman R, et al. (2006b) Advances and challenges in flip-chip packaging. In: IEEE custom integrated circuits conference 2006, pp. 703–709.

Mallik A, Ryckaert J, Kim RH, et al. (2019) Economics of semiconductor scaling - a cost analysis for advanced technology node. In: 2019 symposium on VLSI technology, pp. T202–T203.

Mandal A, Fowler R and Porterfield A (2010) Modeling memory concurrency for multi-socket multi-core systems. In: 2010 IEEE international symposium on performance analysis of systems & software (ISPASS), pp. 66–75.

Marculescu D and Talpes E (2005) Variability and energy awareness: a microarchitecture-level perspective. In: Proceedings of the 42nd annual design automation conference, DAC '05. New York, NY: Association for Computing Machinery, pp. 11–16.

Markidis S, Chien SWD, Laure E, et al. (2018) NVIDIA tensor core programmability, performance & precision. In: 2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW), pp. 522–531.

Marks MR, Cheong KY and Hassan Z (2022) A review of laser ablation and dicing of Si wafers. *Precision Engineering* 73: 377–408.

Marowka A (2011) Back to thin-core massively parallel processors. *Computer* 44(12): 49–54.

Marquardt A, Betz V and Rose J (2000) Speed and area tradeoffs in cluster-based FPGA architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8(1): 84–93.

Martens B, Walterbusch M and Teuteberg F (2012) Costing of cloud computing services: a total cost of ownership approach. In: 2012 45th Hawaii international conference on system sciences, pp. 1563–1572.

Martin KJM (2022) Twenty years of automated methods for mapping applications on CGRA. In: 2022 IEEE international parallel and distributed processing symposium workshops (IPDPSW), pp. 679–686.

Matin MA (2018) *AM, Angle Modulation and Digital Modulation Systems*. Chapter 4. Springer International Publishing, pp. 43–69.

Matteis TD, Licht JF and Hoefler T (2020) FBLAS: streaming linear algebra on FPGA. In: SC20: International conference for high performance computing, networking, storage and analysis, pp. 1–13.

Mattioli M (2021) Rome to Milan, AMD continues its tour of Italy. *IEEE Micro* 41(4): 78–83.

McClelland F (1983) Services and protocols of the physical layer. *Proceedings of the IEEE* 71(12): 1372–1377.

Mechaik M (2001) An evaluation of single-ended and differential impedance in PCBs. In: Proceedings of the IEEE 2001. 2nd international symposium on quality electronic design, pp. 301–306.

Mehis A and Radhakrishnan R (2002) Optimizing applications for performance on the pentium 4 architecture. In: 2002 IEEE international workshop on workload characterization, pp. 59–67.

Mei B, De Sutter B, Vander Aa T, et al. (2008) Implementation of a coarse-grained reconfigurable media processor for AVC decoder. *Journal of Signal Processing Systems* 51(3): 225–243.

Meijer M, Pessolano F and de Gyvez JP (2004) Technology exploration for adaptive power and frequency scaling in 90nm CMOS. In: Proceedings of the 2004 international symposium on low power electronics and design, ISLPED '04. New York, NY: Association for Computing Machinery, pp. 14–19.

Mekawey H, Elsayed M, Ismail Y, et al. (2022) Optical interconnects finally seeing the light in silicon photonics: past the hype. *Nanomaterials* 12(3): 485.

Mencer O, Allison D, Blatt E, et al. (2020) The history, status, and future of FPGAs: hitting a nerve with field-programmable gate arrays. *Queue* 18(3): 71–82.

Mercier H, Bhargava VK and Tarokh V (2010) A survey of error-correcting codes for channels with symbol synchronization errors. *IEEE Communications Surveys & Tutorials* 12(1): 87–96.

Messerschmitt D (1990) Synchronization in digital system design. *IEEE Journal on Selected Areas in Communications* 8(8): 1404–1419.

Mhaboobkhan F, Fathimaparveen M, Gokila K, et al. (2019) Implementation of high speed data transfer serialized 128/130 bit encoding algorithm using 90nm technology. In: 2019 5th International conference on advanced computing & communication systems (ICACCS), pp. 732–736.

Micikevicius P, Narang S, Alben J, et al. (2018) Mixed precision training. In: *International Conference on Learning Representations*. Open Review, pp. 1–12.

Miller DAB (2009) Device requirements for optical interconnects to silicon chips. *Proceedings of the IEEE* 97(7): 1166–1185.

Miller D and Ozaktas H (1997) Limit to the bit-rate capacity of electrical interconnects from the aspect ratio of the system architecture. *Journal of Parallel and Distributed Computing* 41(1): 42–52.

Mittal S (2019) A survey of techniques for dynamic branch prediction. *Concurrency and Computation: Practice and Experience* 31(1): e4666.

Mittal S (2020a) A survey on evaluating and optimizing performance of Intel Xeon Phi. *Concurrency and Computation: Practice and Experience* 32(19): e5742.

Mittal S (2020b) A survey on evaluating and optimizing performance of Intel Xeon Phi. *Concurrency and Computation: Practice and Experience* 32(19): e5742.

Miyake T, Yamashita T, Asari N, et al. (2001) Design methodology of high performance microprocessor using ultra-low threshold voltage CMOS. In: Proceedings of the IEEE 2001 custom integrated circuits conference (Cat. No.01CH37169), pp. 275–278.

Mochocki B, Lahiri K and Cadambi S (2006) Power analysis of mobile 3D graphics. *Proceedings of the Design Automation & Test in Europe Conference* 1: 1–6.

Modi H, Spracklen L, Chou Y, et al. (2005) Accurate modeling of aggressive speculation in modern microprocessor architectures. In: 13th IEEE international symposium on modeling, analysis, and simulation of computer and telecommunication systems, pp. 75–84.

Mohapatra S, Gupta HS, Singh J, et al. (2017) A 64b/66b line encoding for high speed serializers. In: 2017 30th International conference on VLSI design and 2017 16th international conference on embedded systems (VLSID), pp. 303–308.

Moore GE (1965) Cramming more components onto integrated circuits. *Electronics* 38(8): 114–117.

Moore GE (2006) Progress in digital integrated electronics [Technical literaiture, Copyright 1975 IEEE. Reprinted, with permission. Technical Digest. International Electron Devices Meeting, IEEE, 1975, pp. 11-13]. IEEE Solid-State Circuits Society Newsletter 11(3): 36–37, originally published in 1975.

Moore SK (2020) The node is nonsense. *IEEE Spectrum* 57(8): 24–30.

Moore SK and Schneider D (2022) The state of the transistor: in 75 years, it's become tiny, mighty, ubiquitous, and just plain weird. *IEEE Spectrum* 59(12): 30–31.

Mujtaba H (2015) Intel's 10nm Knights Hill powered Aurora supercomputer to feature up to 180 PetaFlops computational power – 2018 launch scheduled.

Mujtaba H (2022) AMD SP5 socket pictured in all its glory, LGA 6096 for future EPYC CPUs with 96 cores & above.

Müller S, Reuschel T, Rimolo-Donadio R, et al. (2015) Energy-aware signal integrity analysis for high-speed PCB links. *IEEE Transactions on Electromagnetic Compatibility* 57(5): 1226–1234.

Munger B, Wilcox K, Sniderman J, et al. (2023) "Zen 4": the AMD 5nm 5.7GHz x86-64 microprocessor core. In: 2023 IEEE international solid- state circuits conference (ISSCC), pp. 38–39.

Mutoh S, Douseki T, Matsuya Y, et al. (1995) 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS. *IEEE Journal of Solid-State Circuits* 30(8): 847–854.

Na N, Wang J, Long S, et al. (2017) Exploring DDR4 address bus design for high speed memory interface. In: 2017 IEEE 67th Electronic components and technology conference (ECTC), pp. 1843–1848.

Naeemi A, Sarvari R and Meindl J (2006) On-chip interconnect networks at the end of the roadmap: limits and nanotechnology opportunities. In: 2006 International interconnect technology conference, pp. 201–203.

Naffziger S, Lepak K, Paraschou M, et al. (2020) 2.2 AMD chiplet architecture for high-performance server and desktop products. In: 2020 IEEE international solid- state circuits conference - (ISSCC), pp. 44–45.

Naffziger S, Beck N, Burd T, et al. (2021) Pioneering chiplet technology and design for the AMD EPYC^{TM} and Ryzen^{TM} processor families: industrial product. In: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). pp. 57–70.

Nane R, Sima VM, Pilato C, et al. (2016) A survey and evaluation of FPGA high-level synthesis tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35(10): 1591–1604.

Narayanan S, Swamy BN and Seznec A (2015) An empirical high level performance model for future many-cores. In: Proceedings of the 12th ACM international conference on computing Frontiers, CF '15. New York, NY: Association for Computing Machinery, pp. 1–8.

Nassif N, Munch AO, Molnar CL, et al. (2022) Sapphire rapids: the next-generation Intel Xeon scalable processor. In: 2022 IEEE international solid- state circuits conference (ISSCC), Vol. 65, pp. 44–46.

Naveh Y and Likharev K (2000) Shrinking limits of silicon MOSFETs: numerical study of 10 nm scale devices. *Superlattices and Microstructures* 27(2): 111–123.

Nieuwoudt A and Massoud Y (2008) On the optimal design, performance, and reliability of future carbon nanotube-based interconnect solutions. *IEEE Transactions on Electron Devices* 55(8): 2097–2110.

Niu KP and Anderson JH (2018) Compact area and performance modelling for CGRA architecture evaluation. In: 2018 International conference on field-programmable technology (FPT), pp. 126–133.

Nowatzki T, Gangadhar V, Ardalani N, et al. (2017) Stream-dataflow acceleration. In: 2017 ACM/IEEE 44th annual international symposium on computer architecture (ISCA), pp. 416–429.

Oberman S, Favor G and Weber F (1999) AMD 3DNow! technology: architecture and implementations. *IEEE Micro* 19(2): 37–48.

Ogasawara A (2002) Trends in research and development of lithography technology for next-generation LSIs. *Science & Technology Trends Quarterly Review.*

Olukotun K and Hammond L (2005) The future of microprocessors: chip multiprocessors' promise of huge performance gains is now a reality. *Queue* 3(7): 26–29.

Owens JD, Luebke D, Govindaraju N, et al. (2007) A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum* 26(1): 80–113.

Pahl C, Brogi A, Soldani J, et al. (2019) Cloud container technologies: a state-of-the-art review. *IEEE Transactions on Cloud Computing* 7(3): 677–692.

Pai V, Ranganathan P and Adve S (1997) The impact of instruction-level parallelism on multiprocessor performance and simulation methodology. In: Proceedings third international symposium on high-performance computer architecture, pp. 72–83.

Palacharla S, Jouppi NP and Smith JE (1997) Complexity-effective superscalar processors. *SIGARCH Computer Architecture News* 25(2): 206–218.

Pan Z, Li C, Hao W, et al. (2022) ESD protection designs: topical overview and perspective. *IEEE Transactions on Device and Materials Reliability* 22(3): 356–370.

Pandey V, Subramanian S, Rangaraj S, et al. (2005) Mechanical design and analysis of land grid array (LGA) sockets. In: International electronic packaging technical conference and exhibition advances in electronic packaging, Parts A, B, and C, pp. 1005–1011.

Pandey AK, Jangale A and Narayan S (2020) Signal integrity and compliance test of DSI and CSI2 serial interface over MIPI D-PHY. In: 2020 IEEE 24th workshop on signal and power integrity (SPI), pp. 1–4.

Papadimitriou G, Chatzidimitriou A and Gizopoulos D (2019) Adaptive voltage/frequency scaling and core allocation for balanced energy and performance on multicore CPUs. In: 2019 IEEE international symposium on high performance computer architecture (HPCA), pp. 133–146.

Park H, Park Y and Mahlke S (2009) Polymorphic pipeline array: a flexible multicore accelerator with virtualized execution for mobile multimedia applications. In: 2009 42nd annual IEEE/ACM international symposium on microarchitecture (MICRO), pp. 370–380.

Parkhurst J, Darringer J and Grundmann B (2006) From single core to multi-core: preparing for a new exponential. In: Proceedings of the 2006 IEEE/ACM international conference on computer-aided design, ICCAD '06. New York, NY: Association for Computing Machinery, pp. 67–72.

Pasricha S, Kurdahi FJ and Dutt N (2010) Evaluating carbon nanotube global interconnects for chip multiprocessor applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18(9): 1376–1380.

Paszke A, Gross S, Massa F, et al. (2019) PyTorch: an imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035.

Patton G (2009) Semiconductor technology-trends, challenges and opportunities. In: 2009 13th international workshop on computational electronics, pp. 1–4.

Paulin P (2004) DATE panel chips of the future: soft, crunchy or hard? In: Proceedings design, automation and test in Europe conference and exhibition, Vol. 2, pp. 844–849.

Peccerillo B, Mannino M, Mondelli A, et al. (2022) A survey on hardware accelerators: taxonomy, trends, challenges, and perspectives. *Journal of Systems Architecture* 129: 102561.

Peddie J (2023a) *The History of the GPU - Eras and Environment*. Springer International Publishing.

Peddie J (2023b) *The History of the GPU - New Developments*. Springer International Publishing.

Peddie J (2023c) *The History of the GPU - Steps to Invention*. Springer International Publishing.

Peleg A and Weister U (1991) Future trends in microprocessors: out-of-order execution, speculative branching and their CISC performance potential. In: 17th Convention of electrical and electronics engineers in Israel, pp. 263–266.

Peleg A, Wilkie S and Weiser U (1997) Intel MMX for multimedia PCs. *Communications of the ACM* 40(1): 24–38.

Pellerite P and Suhl D (1988) Sockets: considerations as an alternative to direct surface mounting of components. In: Fourth IEEE/CHMT European international electronic manufacturing technology symposium, pp. 89–91.

Peng Z, Fattal D, Fiorentino M, et al. (2010) Fabrication variations in SOI microrings for DWDM networks. In: 7th IEEE international conference on group IV photonics, pp. 120–122.

Peng IB, Gioiosa R, Kestor G, et al. (2017) Exploring the performance benefit of hybrid memory system on HPC environments. In: 2017 IEEE international parallel and distributed processing symposium workshops (IPDPSW), pp. 683–692.

Pohl C and Sattler KU (2018) Joins in a heterogeneous memory hierarchy: exploiting high-bandwidth memory. In: Proceedings of the 14th international workshop on data management on new hardware, DAMON '18. New York, NY: Association for Computing Machinery, pp. 1–10.

Prabhakar R and Jairath S (2021) SambaNova SN10 RDU: accelerating software 2.0 with dataflow. In: 2021 IEEE hot chips 33 symposium (HCS), pp. 1–37.

Prabhakar R, Zhang Y and Olukotun K (2020) *Coarse-Grained Reconfigurable Architectures*. Chapter 14. Springer International Publishing, pp. 227–246.

Prabhakar R, Jairath S and Shin JL (2022) SambaNova SN10 RDU: a 7nm dataflow architecture to accelerate software 2.0. 2022 IEEE international solid- state circuits conference (ISSCC), Vol. 65, pp. 350–352.

Pratheek B, Jawalkar N and Basu A (2022) Designing virtual memory system of MCM GPUs. In: 2022 55th IEEE/ACM international symposium on microarchitecture (MICRO), pp. 404–422.

Prince B (1999) A tribute to graphics DRAMs. In: Records of the 1999 IEEE international workshop on memory technology, design and testing, pp. 123–130.

Qasaimeh M, Denolf K, Lo J, et al. (2019) Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels. In: 2019 IEEE international conference on embedded software and systems (ICESS), pp. 1–8.

Qian J, Pullela S and Pillage L (1994) Modeling the "effective capacitance" for the RC interconnect of CMOS gates. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13(12): 1526–1535.

Quinones E, Parcerisa JM and Gonzailez A (2007) Improving branch prediction and predicated execution in out-of-order processors. In: 2007 IEEE 13th international symposium on high performance computer architecture, pp. 75–84.

Radhakrishnan K, Swaminathan M and Bhattacharyya BK (2021) Power delivery for high-performance microprocessors—Challenges, solutions, and future trends. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 11(4): 655–671.

Rahim A, Spuesens T, Baets R, et al. (2018) Open-access silicon photonics: current status and emerging initiatives. *Proceedings of the IEEE* 106(12): 2313–2330.

Rahman BMA, Leung DMH, Obayya SSA, et al. (2008) Numerical analysis of bent waveguides: bending loss, transmission loss, mode coupling, and polarization coupling. *Applied optics* 47(16): 2961–2970.

Rakheja S and Kumar V (2012) Comparison of electrical, optical and plasmonic on-chip interconnects based on delay and energy considerations. In: Thirteenth international symposium on quality electronic design (ISQED), pp. 732–739.

Raman S, Pentkovski V and Keshava J (2000) Implementing streaming SIMD extensions on the Pentium III processor. *IEEE Micro* 20(4): 47–57.

Randell B, Lee P and Treleaven PC (1978) Reliability issues in computing system design. *ACM Computing Surveys* 10(2): 123–165.

Rashdan M, El-Sayed F and Salman M (2020) Performance comparison between SerDes and time-based serial links. In: 2020 7th International conference on electrical and electronics engineering (ICEEE), pp. 37–41.

Rau BR and Fisher JA (2003) *Instruction-Level Parallelism*. Chapter I. John Wiley and Sons Ltd, pp. 883–887.

Ray J and Hoe JC (2003) High-level modeling and FPGA prototyping of microprocessors. In: Proceedings of the 2003 ACM/SIGDA eleventh international symposium on field programmable gate arrays, FPGA '03. New York, NY: Association for Computing Machinery, pp. 100–107.

Rezaei F, Galappaththige D, Tellambura C, et al. (2023) Coding techniques for backscatter communications—A contemporary survey. *IEEE Communications Surveys & Tutorials* 25(2): 1020–1058.

Rieger ML (2019) Retrospective on VLSI value scaling and lithography. *Journal of Micro/Nanolithography, MEMS, and MOEMS* 18(4): 040902.

Riesgo T, Torroja Y and de la Torre E (1999) Design methodologies based on hardware description languages. *IEEE Transactions on Industrial Electronics* 46(1): 3–12.

Robe T, Banwell T, Hodge J, et al. (1993) 4B/5B block code to SONET OC-3 (155-Mbit/s) interface for ATM local area networks. In: Conference on optical fiber communication/international conference on integrated optics and optical fiber communication. Optica Publishing Group, p. WJ3.

Rodriguez JN, Canosa MC and Pereira EH (2011) Improving electrical power grid visualization using geometry shaders. In: 2011 Eighth international conference computer graphics, imaging and visualization, pp. 177–182.

Roman S (1998) *The Parallel Interface*. Chapter 18. Springer New York, pp. 291–298.

Ronak B and Fahmy SA (2016) Mapping for maximum performance on FPGA DSP blocks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35(4): 573–585.

Ronen R, Mendelson A, Lai K, et al. (2001) Coming challenges in microarchitecture and architecture. *Proceedings of the IEEE* 89(3): 325–340.

Roorda E, Rasoulinezhad S, Leong PHW, et al. (2022) FPGA architecture exploration for DNN acceleration. *ACM Transactions on Reconfigurable Technology and Systems* 15(3): 1–37.

Rose J (2004) Hard vs. soft: the central question of pre-fabricated silicon. In: Proceedings. 34th international symposium on multiple-valued logic, pp. 2–5.

Rotem E, Naveh A, Ananthakrishnan A, et al. (2012) Power-management architecture of the intel microarchitecture code-named Sandy Bridge. *IEEE Micro* 32(2): 20–27.

Rotem E, Yoaz A, Rappoport L, et al. (2022) Intel alder lake CPU architectures. *IEEE Micro* 42(3): 13–19.

Roy K, Mukhopadhyay S and Mahmoodi-Meimand H (2003) Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proceedings of the IEEE* 91(2): 305–327.

Ruehli A and Brennan P (1975) Capacitance models for integrated circuit metallization wires. *IEEE Journal of Solid-State Circuits* 10(6): 530–536.

Rupp K (2022) Microprocessor trend data.

Rysavy P (2014) Challenges and considerations in defining spectrum efficiency. *Proceedings of the IEEE* 102(3): 386–392.

Ryu C, Kwon KW, Loke A, et al. (1999) Microstructure and reliability of copper interconnects. *IEEE Transactions on Electron Devices* 46(6): 1113–1120.

Sakurai T (1993) Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSIs. *IEEE Transactions on Electron Devices* 40(1): 118–124.

Salehian S and Yan Y (2017) Evaluation of knight landing high bandwidth memory for HPC workloads. In: Proceedings of the seventh workshop on irregular applications: architectures and algorithms, IA3'17. New York, NY: Association for Computing Machinery, pp. 1–4.

Sanca V and Ailamaki A (2023) Post-Moore's law fusion: high-bandwidth memory, accelerators, and native half-precision processing for CPU-local analytics. In: Joint workshops at 49th international conference on very large data bases (VLDBW'23), pp. 13.

Sancho JC, Lang M and Kerbyson DJ (2010) Analyzing the trade-off between multiple memory controllers and memory channels on multi-core processor performance. In: 2010 IEEE international symposium on parallel & distributed processing, workshops and Phd forum (IPDPSW), pp. 1–7.

Saraswat K, Cho H, Kapur P, et al. (2008) Performance comparison between copper, carbon nanotube, and optical interconnects. In: 2008 IEEE international symposium on circuits and systems (ISCAS), pp. 2781–2784.

Sarmah MJ and Azeemuddin S (2014) A circuit to synchronize high speed serial communication channel. In: 2014 International conference on field-programmable technology (FPT), pp. 239–242.

Sarmah MJ and Azeemuddin S (2015) A circuit to eliminate serial skew in high-speed serial communication channels. *IEEE Transactions on Circuits and Systems II: Express Briefs* 62(12): 1179–1183.

Sarmah MJ and Azeemuddin S (2017) Circuits for initializing simplex communication channels. In: 2017 IEEE international conference on computational intelligence and computing research (ICCIC), pp. 1–4.

Sato T, Takeda K, Shinya A, et al. (2015) Photonic crystal lasers for chip-to-chip and On-Chip optical interconnects. *IEEE Journal of Selected Topics in Quantum Electronics* 21(6): 728–737.

Savage N (2002) Linking with light [high-speed optical interconnects]. *IEEE Spectrum* 39(8): 32–36.

Schlansker M, Conte T, Dehnert J, et al. (1997) Compilers for instruction-level parallelism. *Computer* 30(12): 63–69.

Schmidl D, Cramer T, Wienke S, et al. (2013) Assessing the performance of OpenMP programs on the Intel Xeon Phi. In: Wolf F, Mohr B and an Mey D (eds) *Euro-Par 2013 Parallel Processing*. Springer Berlin Heidelberg, pp. 547–558.

Schrimpf RD, Warren KM, Weller RA, et al. (2008) Reliability and radiation effects in IC technologies. In: 2008 IEEE international reliability physics symposium, pp. 97–106.

Seckin U and Yang CKK (2008) A comprehensive delay model for CMOS CML circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers* 55(9): 2608–2618.

Seiler L, Carmean D, Sprangle E, et al. (2009) Larrabee: a many-core x86 architecture for visual computing. *IEEE Micro* 29(1): 10–21.

Seshadri N, Sundberg CEW and Weerackody V (1993) Advanced techniques for modulation, error correction, channel equalization, and diversity. *AT&T Technical Journal* 72(4): 48–63.

Shah K and Mello M (2004) Ball grid array solder joint failure envelope development for dynamic loading. In: 2004 Proceedings. 54th Electronic components and technology conference (IEEE Cat. No.04CH37546), Vol. 1. pp. 1067–1074.

Shahdad M, Lipsett R, Marschner E, et al. (1985) VHSIC hardware description language. *Computer* 18(2): 94–103.

Shahidi GG (2007) Evolution of CMOS technology at 32 nm and beyond. In: 2007 IEEE custom integrated circuits conference, pp. 413–416.

Shahzad H, Sanaullah A and Herbordt M (2021) Survey and future trends for FPGA cloud architectures. In: 2021 IEEE high performance extreme computing conference (HPEC), pp. 1–10.

Shanley T, Anderson D, Swindle J, et al. (1995) *ISA System Architecture*. Mindshare PC System Architecture. Addison-Wesley.

Shao YS and Brooks D (2013) Energy characterization and instruction-level energy model of Intel's Xeon Phi processor. In: International symposium on low power electronics and design (ISLPED), pp. 389–394.

Sharma A, Bamiedakis N, Karinou F, et al. (2021) Multi-chiplet system architecture with shared uniform access memory based on board-level optical interconnects. In: 2021 Optical fiber communications conference and exhibition (OFC), pp. 1–3.

Shipman GM, Swaminarayan S, Grider G, et al. (2022) Early performance results on 4th Gen Intel(R) Xeon (R) scalable processors with DDR and Intel(R) Xeon(R) processors, codenamed sapphire rapids with HBM.

Shooman ML (2002) *Introduction*. Chapter 1. John Wiley & Sons, Ltd, pp. 1–29.

Shorey AB and Lu R (2016) Progress and application of through glass via (TGV) technology. In: 2016 Pan Pacific microelectronics symposium (Pan Pacific), pp. 1–6.

Sideco F (2023) Design once. Sell multiple times – AMD leveraging chiplets to execute on workload optimized processing strategy.

Silva VRG, Furtunato AFA, Georgiou K, et al. (2019) Energy-optimal configurations for single-node HPC applications. In: 2019 International conference on high performance computing & simulation (HPCS), pp. 448–454.

Sim SP, Krishnan S, Petranovic D, et al. (2003) A unified RLC model for high-speed on-chip interconnects. *IEEE Transactions on Electron Devices* 50(6): 1501–1510.

Singh S and Singh N (2016) Containers & Docker: emerging roles & future of Cloud technology. In: 2016 2nd International conference on applied and theoretical computing and communication technology (iCATccT), pp. 804–807.

Singhal S, Gaur N, Mehra A, et al. (2015) Analysis and comparison of leakage power reduction techniques in CMOS circuits. In: 2015 2nd International conference on signal processing and integrated networks (SPIN), pp. 936–944.

Smith J and Sohi G (1995) The microarchitecture of superscalar processors. *Proceedings of the IEEE* 83(12): 1609–1624.

Smithson G (1998) Introduction to digital modulation schemes. In: IEE colloquium on the design of digital cellular handsets (Ref. No. 1998/240), pp. 2/1–2/9.

Sodani A (2015) Knights landing (KNL): 2nd generation Intel® Xeon Phi processor. In: 2015 IEEE hot chips 27 symposium (HCS), pp. 1–24.

Sodani A, Gramunt R, Corbal J, et al. (2016) Knights landing: second-generation Intel Xeon Phi product. *IEEE Micro* 36(2): 34–46.

Song BS and Soo D (1997) NRZ timing recovery technique for band-limited channels. *IEEE Journal of Solid-State Circuits* 32(4): 514–520.

Soref R and Bennett B (1987) Electrooptical effects in silicon. *IEEE Journal of Quantum Electronics* 23(1): 123–129.

Soref R and Larenzo J (1986) All-silicon active and passive guided-wave components for $\lambda = 1.3$ and $1.6\ \mu$ m. *IEEE Journal of Quantum Electronics* 22(6): 873–879.

Spector A and Gifford D (1984) The space shuttle primary computer system. *Communications of the ACM* 27(9): 872–900.

Sreerama C, Hall SH, Huray PG, et al. (2018) A crosstalk-friendly signaling method. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 8(9): 1621–1631.

Srinivasan V, Brooks D, Gschwind M, et al. (2002) Optimizing pipelines for power and performance. In: 35th Annual IEEE/ACM international symposium on microarchitecture, 2002. (MICRO-35). Proceedings. pp. 333–344.

Srivastava N and Banerjee K (2004) Interconnect challenges for nanoscale electronic circuits. *JOM* 56(10): 30–31.

Staff IRE (2019) One big wire change in '97 still helping chips achieve tiny scale.

Stanley-Marbell P, Cabezas VC and Luijten RP (2011) Pinned to the walls — impact of packaging and application properties on the memory and power walls. In: IEEE/ACM international symposium on low power electronics and design, pp. 51–56.

Stanzione D, Barth B, Gaffney N, et al. (2017) Stampede 2: the evolution of an XSEDE supercomputer. In: Proceedings of the practice and experience in advanced research computing 2017 on sustainability, success and impact, PEARC17. New York, NY: Association for Computing Machinery, pp. 1–8.

Stillmaker A and Baas B (2017) Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm. *Integration* 58: 74–81.

Stoll C, Gumhold S and Seidel HP (2005) Visualization with stylized line primitives. In: VIS 05: IEEE visualization, 2005, pp. 695–702.

Su LT, Naffziger S and Papermaster M (2017) Multi-chip technologies to unleash computing performance gains over the next decade. In: 2017 IEEE international electron devices meeting (IEDM), pp. 1.1.1–1.1.8.

Suggs D, Subramony M and Bouvier D (2020) The AMD "Zen 2" processor. *IEEE Micro* 40(2): 45–52.

Sugimoto S, Hayashi K and Mano F (1989) Design of 2B1Q transceiver for ISDN subscriber loops. In: IEEE international conference on communications, world prosperity through communications, Vol.1, pp. 228–232.

Sumeet N, Rawat K and Nambiar M (2022) Performance evaluation of GraphCore IPU-M2000 accelerator for text detection application. In: Companion of the 2022 ACM/SPEC international conference on performance engineering, ICPE '22. New York, NY: Association for Computing Machinery, pp. 145–152.

Summerville DH (2009) *Serial Communication*. Chapter 2. Springer International Publishing, pp. 77–111.

Sun Y, Mukherjee S, Baruah T, et al. (2018) Evaluating performance tradeoffs on the radeon open compute platform. In: 2018 IEEE international symposium on performance analysis of systems and software (ISPASS), pp. 209–218.

Sun Y, Agostini NB, Dong S, et al. (2020) Summarizing CPU and GPU design trends with product data.

Sun PSV, Titterton A, Gopiani A, et al. (2022a) Intelligence processing units accelerate neuromorphic learning.

Sun Y, Zheng L, Wang Q, et al. (2022b) Accelerating sparse deep neural network inference using GPU tensor cores. In: 2022 IEEE high performance extreme computing conference (HPEC), pp. 1–7.

Sundareshan B (1992) Digital modulation—Baseband techniques. *IETE Journal of Education* 33(1): 35–44.

Suzuki O (2020) Recent advances in underfill for new package architectures. In: 2020 Pan Pacific microelectronics symposium (Pan Pacific), pp. 1–7.

Sylvester D and Kaul H (2001) Power-driven challenges in nanometer design. *IEEE Design & Test of Computers* 18(6): 12–21.

Sylvester D and Keutzer K (1998) Getting to the bottom of deep submicron. In: 1998 IEEE/ACM international conference on computer-aided design. Digest of technical papers (IEEE Cat. No.98CB36287), pp. 203–211.

Sylvester D and Keutzer K (2000) A global wiring paradigm for deep submicron design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19(2): 242–252.

Sze V, Chen YH, Emer J, et al. (2017) Hardware for machine learning: challenges and opportunities. In: 2017 IEEE custom integrated circuits conference (CICC), pp. 1–8.

Taka E, Arora A, Wu KC, et al. (2023) MaxEVA: maximizing the efficiency of matrix multiplication on versal AI engine. In: 2023 International conference on field programmable technology (ICFPT), pp. 96–105.

Taka E, Gourounas D, Gerstlauer A, et al. (2024) Efficient approaches for GEMM acceleration on leading AI-Optimized FPGAs. In: 2024 IEEE 32nd annual international symposium on field-programmable custom computing machines (FCCM), pp. 54–65.

Taka E, Huang NC, Chang CC, et al. (2025) Systolic sparse tensor slices: FPGA building blocks for sparse and dense AI acceleration. In: Proceedings of the 2025 ACM/SIGDA international symposium on field programmable gate arrays, FPGA '25. New York, NY: Association for Computing Machinery, pp. 159–171.

Takahashi S, Horiuchi K, Tatsukoshi K, et al. (2013) Development of through glass via (TGV) formation technology using electrical discharging for 2.5/3D integrated packaging. In: 2013 IEEE 63rd electronic components and technology conference, pp. 348–352.

Talpes E, Williams D and Sarma DD (2022) Dojo: the microarchitecture of tesla's exa-scale computer. In: 2022 IEEE Hot Chips 34 Symposium (HCS), pp. 1–28.

Talpes E, Sarma DD, Williams D, et al. (2023) The microarchitecture of DOJO, Tesla's exa-scale computer. *IEEE Micro* 43(3): 31–39.

Tam SM, Muljono H, Huang M, et al. (2018) SkyLake-SP: a 14nm 28-Core xeon® processor. In: 2018 IEEE international solid - state circuits conference - (ISSCC), pp. 34–36.

Tamitani I, Ohta M, Nomura M, et al. (1992) An encoder/decoder chip set for the MPEG video standard. In: Acoustics, speech, and signal processing, IEEE international conference on, Volume 5. Los Alamitos, CA: IEEE Computer Society, pp. 661–664.

Tan L, Kothapalli S, Chen L, et al. (2014) A survey of power and energy efficient techniques for high performance numerical linear algebra operations. *Parallel Computing* 40(10): 559–573.

Tan C, Xie C, Li A, et al. (2021) AURORA: automated refinement of coarse-grained reconfigurable accelerators. In: 2021 Design, automation & test in Europe conference & exhibition (DATE), pp. 1388–1393.

Taur Y (1999a) CMOS scaling beyond 0.1/spl mu/m: how far can it go? In: 1999 International symposium on VLSI technology, systems, and applications. Proceedings of technical papers. (Cat. No.99TH8453). pp. 6–9.

Taur Y (1999b) The incredible shrinking transistor. *IEEE Spectrum* 36(7): 25–29.

Taylor MB (2012) Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In: Proceedings of the 49th annual design automation conference, DAC '12. New York, NY: Association for Computing Machinery, pp. 1131–1136.

Taylor MB (2013) A landscape of the new dark silicon design regime. *IEEE Micro* 33(5): 8–19.

Teixeira M and Zaharov V (2007) *Digital Transmission*. Chapter 7. John Wiley & Sons, Ltd, pp. 84–101.

Teshima M, Kobayashi S, Yamamoto T, et al. (2008) Bit-error-tolerant (512*N)B/(513*N+1)B code for 40Gb/s and 100Gb/s ethernet transport. In: IEEE INFOCOM workshops 2008, pp. 1–6.

Theis TN (2000) The future of interconnection technology. *IBM Journal of Research and Development* 44(3): 379–390.

Theis TN and Wong HSP (2017) The end of Moore's law: a new beginning for information technology. *Computing in Science & Engineering* 19(2): 41–50.

Tilli M (2010) Chapter five - silicon wafers: preparation and properties. In: Lindroos V, Tilli M, Lehto A, et al. (eds) *Handbook of Silicon Based MEMS Materials and Technologies, Micro and Nano Technologies*. William Andrew Publishing, pp. 71–88.

Tiwari V, Singh D, Rajgopal S, et al. (1998) Reducing power in high-performance microprocessors. In: Proceedings of the 35th annual design automation conference, DAC '98. New York, NY: Association for Computing Machinery, pp. 732–737.

Todri-Sanial A, Ramos R, Okuno H, et al. (2017) A survey of carbon nanotube interconnects for energy efficient integrated circuits. *IEEE Circuits and Systems Magazine* 17(2): 47–62.

Tőkei Z, Ciofi I, Roussel P, et al. (2016) On-chip interconnect trends, challenges and solutions: how to keep RC and reliability under control. In: 2016 IEEE symposium on VLSI technology, pp. 1–2.

Tong JG, Anderson IDL and Khalid MAS (2006) Soft-core processors for embedded systems. In: 2006 International conference on microelectronics, pp. 170–173.

Tonietto D (2022) The future of short reach interconnect. In: ESSCIRC 2022- IEEE 48th European solid state circuits conference (ESSCIRC), pp. 1–8.

Trader T (2017) Graphcore readies launch of 16nm Colossus-IPU chip.

Tukanov N, Srinivasaraghavan R, Moreira JE, et al. (2022) Modeling matrix engines for portability and performance. In: 2022 IEEE international parallel and distributed processing symposium (IPDPS), pp. 1173–1183.

Turkane SM and Kureshi AK (2017) Emerging interconnects: a state-of-the-art review and emerging solutions. *International Journal of Electronics* 104(7): 1107–1119.

Tzimpragos G, Kachris C, Djordjevic IB, et al. (2016) A survey on FEC codes for 100 G and beyond optical networks. *IEEE Communications Surveys & Tutorials* 18(1): 209–221.

Van Kerrebrouck J, De Keulenaer T, Pierco R, et al. (2019) NRZ, duobinary, or PAM4?: Choosing among high-speed electrical interconnects. *IEEE Microwave Magazine* 20(7): 24–35.

Vanderwiel SP and Lilja DJ (2000) Data prefetch mechanisms. *ACM Computing Surveys* 32(2): 174–199.

Vanna-Iampikul P, Zhu L, Erdogan S, et al. (2023) Glass interposer integration of logic and memory chiplets: PPA and power/signal integrity benefits. In: 2023 60th ACM/IEEE design automation conference (DAC), pp. 1–6.

Vasilakis E, Sourdis I, Papaefstathiou V, et al. (2017) Modeling energy-performance tradeoffs in ARM big.LITTLE architectures. In: 2017 27th International symposium on power and timing modeling, optimization and simulation (PATMOS), pp. 1–8.

Veendrick H (1984) Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits. *IEEE Journal of Solid-State Circuits* 19(4): 468–473.

Venkataraman A, Amadi EV, Chen Y, et al. (2019) Carbon nanotube assembly and integration for applications. *Nanoscale Research Letters* 14(1): 220.

Verbauwhede I, Schaumont P, Piguet C, et al. (2004) Architectures and design techniques for energy efficient embedded DSP and multimedia processing. *Proceedings Design, Automation and Test in Europe Conference and Exhibition* 2: 988–993.

Véstias M and Neto H (2014) Trends of CPU, GPU and FPGA for high-performance computing. In: 2014 24th International conference on field programmable logic and applications (FPL), pp. 1–6.

Vidya S, Kamat SV, Khan A, et al. (2018) 3D FinFET for next generation nano devices. In: 2018 International conference on current trends towards converging technologies (ICCTCT), pp. 1–9.

Wade M, Anderson E, Ardalan S, et al. (2020) TeraPHY: a chiplet technology for low-power, high-bandwidth in-package optical I/O. *IEEE Micro* 40(2): 63–71.

Wall DW (1991) Limits of instruction-level parallelism. *SIGPLAN Not* 26(4): 176–188.

Wang F and Agrawal VD (2008) Single event upset: an embedded tutorial. In: 21st International conference on VLSI design (VLSID 2008), pp. 429–434.

Wang S and Kanwar P (2019) BFloat16: the secret to high performance on Cloud TPUs. Google Cloud Blog 4.

Wang L and Skadron K (2013) Implications of the power wall: dim cores and reconfigurable logic. *IEEE Micro* 33(5): 40–48.

Wang Q, Hua S and Wang D (2010) A 1.1 GHz 8B/10B encoder and decoder design. In: 2010 Asia Pacific conference on postgraduate research in microelectronics and electronics (PrimeAsia), pp. 138–141.

Wang E, Zhang Q, Shen B, et al. (2014) *High-Performance Computing on the Intel® Xeon Phi™*. Springer International Publishing.

Wang N, Choi J, Brand D, et al. (2018) Training deep neural networks with 8-Bit floating point numbers. In: Proceedings of the 32nd international conference on neural information processing systems, NIPS'18. Red Hook, NY: Curran Associates Inc., pp. 7686–7695.

Wang Z, Huang H, Zhang J, et al. (2020) Shuhai: benchmarking high bandwidth memory on FPGAS. In: 2020 IEEE 28th annual international symposium on field-programmable custom computing machines (FCCM), pp. 111–119.

Wei J (2008) Challenges in cooling design of CPU packages for high-performance servers. *Heat Transfer Engineering* 29(2): 178–187.

Wei L, Chen Z, Johnson M, et al. (1998) Design and optimization of low voltage high performance dual threshold CMOS circuits. In: Proceedings of the 35th annual design automation conference, DAC '98. New York, NY: Association for Computing Machinery, pp. 489–494.

Wei S, Lin X, Tu F, et al. (2023) Reconfigurability, why it matters in AI tasks processing: a survey of reconfigurable AI chips. *IEEE Transactions on Circuits and Systems I: Regular Papers* 70(3): 1228–1241.

Weng PY, Chen CH, Chen CH, et al. (2018) Factors affecting near-end crosstalk (NEXT) in high speed serial links. In: 2018 15th international conference on electromagnetic interference & compatibility (INCEMIC), pp. 1–4.

Weng PY, Chen CH, Chen J, et al. (2021) Eye comparison between unencoded and 128b/130b-encoded NRZ signals. In: 2021 IEEE 30th conference on electrical performance of electronic packaging and systems (EPEPS), pp. 1–3.

Werner S, Navaridas J and Luján M (2017) A survey on optical network-on-chip architectures. *ACM Computing Surveys* 50(6): 1–37.

Weste N and Harris D (2010) *CMOS VLSI Design: A Circuits and Systems Perspective*. 4th edition. Addison-Wesley Publishing Company.

Wijerathne D, Li Z, Pathania A, et al. (2022) HiMap: fast and scalable high-quality mapping on CGRA via hierarchical abstraction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41(10): 3290–3303.

Winzer PJ (2012) High-spectral-efficiency optical modulation formats. *Journal of Lightwave Technology* 30(24): 3824–3835.

Wong C and Wong M (1999) Recent advances in plastic packaging of flip-chip and multichip modules (MCM) of microelectronics. *IEEE Transactions on Components and Packaging Technologies* 22(1): 21–25.

Wu KC and Tsai YW (2004) Structured ASIC, evolution or revolution? In: Proceedings of the 2004 international symposium on physical design, ISPD '04. New York, NY: Association for Computing Machinery, pp. 103–106.

Wu Q, Xu J, Li X, et al. (2009) The research and implementation of interfacing based on PCI express. In: 2009 9th International conference on electronic measurement & instruments, pp. 3–116–3–121.

Wu TL, Buesink F and Canavero F (2013) Overview of signal integrity and EMC design technologies on PCB: fundamentals and latest progress. *IEEE Transactions on Electromagnetic Compatibility* 55(4): 624–638.

Wu Z, Zhang C, Li F, et al. (2016) High speed serial interface transceiver controller based on JESD204B. In: 2016 14th IEEE international new circuits and systems conference (NEWCAS), pp. 1–4.

Xiong F (2006) *Digital Modulation Techniques*. 2nd edition. Artech House, Inc.

Xiu L (2017) Clock technology: the next frontier. *IEEE Circuits and Systems Magazine* 17(2): 27–46.

Xiu L (2019) Time moore: exploiting Moore's law from the perspective of time. *IEEE Solid-State Circuits Magazine* 11(1): 39–55.

Xu C, Chen X, Dick RP, et al. (2010) Cache contention and application performance prediction for multi-core systems. In: 2010 IEEE international symposium on performance analysis of systems & software (ISPASS), pp. 76–86.

Xu B, Chen R, Zhou J, et al. (2022) Recent progress and challenges regarding carbon nanotube on-chip interconnects. *Micromachines* 13(7): 1148.

Xylon doo (2021) *logi3D Scalable 3D Graphic Accelerator. Datasheet*. Xylon D.O.O.

Yazdani M, Ferry D and Akers L (1997) Microprocessor pin predicting. *IEEE Circuits and Devices Magazine* 13(2): 28–31.

Yeoh HP, Lii MJ, Sankman B, et al. (2000) Flip chip pin grid array (FC-PGA) packaging technology. In: Proceedings of 3rd electronics packaging technology conference (EPTC 2000) (Cat. No.00EX456), pp. 33–40.

Yoon SW, Ku JH, Suthiwongsunthorn N, et al. (2009) Fabrication and packaging of microbump interconnections for 3D TSV. In: 2009 IEEE international conference on 3D system integration, pp. 1–5.

Yue P and Shekhar S (2022) F4: paving the way to 200Gb/s transceivers. In: 2022 IEEE international solid- state circuits conference (ISSCC), 65, pp. 537–539.

Zahiri B (2003) Structured ASICs: opportunities and challenges. In: Proceedings 21st international conference on computer design, San Jose, CA, 13–15 October 2003, pp. 404–409.

Zhang HY, Zhang XW, Lau BL, et al. (2013) Thermal characterization and simulation study of 2.5D packages with multichip module on through silicon interposer. In: 2013 IEEE 15th electronics packaging technology conference (EPTC 2013), Piscataway, NJ, 11–13 December 2013, pp. 363–368.

Zhao W, Li X, Gu S, et al. (2009) Field-based capacitance modeling for sub-65-nm on-chip interconnect. *IEEE Transactions on Electron Devices* 56(9): 1862–1872.

Zheng H, Lin J, Zhang Z, et al. (2008) Mini-rank: adaptive DRAM architecture for improving memory power efficiency. In: 2008 41st IEEE/ACM international symposium on microarchitecture, Lake Como, Italy, 8–12 November 2008, pp. 210–221.

Zhou D, Preparata F and Kang S (1988) Interconnection delay in very high-speed VLSI. In: Proceedings 1988 IEEE international conference on computer design: VLSI, Rye Brook, NY, 03–05 October 1988, pp. 52–55.

Zhou S, Kannan R, Prasanna VK, et al. (2019) HitGraph: high-throughput graph processing framework on FPGA. *IEEE Transactions on Parallel and Distributed Systems* 30(10): 2249–2264.

Zhou Z, Ou X, Fang Y, et al. (2023) Prospects and applications of on-chip lasers. *eLight* 3(1): 1.

Zhuang B, Liu J, Pan Z, et al. (2023a) A survey on efficient training of transformers.

Zhuang J, Lau J, Ye H, et al. (2023b) CHARM: composing heterogeneous accelerators for matrix multiply on versal ACAP architecture. In: Proceedings of the 2023 ACM/SIGDA international symposium on field programmable gate arrays, FPGA '23. New York, NY: Association for Computing Machinery, pp. 153–164.

Zhuravlev S, Blagodurov S and Fedorova A (2010) Addressing shared resource contention in multicore processors via scheduling. In: Proceedings of the fifteenth international conference on architectural support for programming languages and operating systems, ASPLOS XV. New York, NY: Association for Computing Machinery, pp. 129–142.

Zyuban V and Kogge P (2000) Optimization of high-performance superscalar architectures for energy efficiency. In: ISLPED'00: Proceedings of the 2000 international symposium on low power electronics and design (Cat. No.00TH8514), Rapallo, Italy, 26–27 July 2000, pp. 84–89.

Zyuban V and Kogge P (2001) Inherently lower-power high-performance superscalar architectures. *IEEE Transactions on Computers* 50(3): 268–285.

Author biographies

Bagus Hanindhito was a PhD student at the Laboratory of Computer Architecture, Department of Electrical and Computer Engineering, The University of Texas at Austin. His research interests include workload characterization, performance evaluation, and architecture-aware optimizations of HPC and AI/ML applications on GPU-accelerated computing clusters and emerging accelerators. After graduating with his PhD in 2024, he works as a Principal Engineer in the Chief Technology Office of the Infrastructure Solutions Group at Dell Technologies. He received his MSE in Computer Engineering from The University of Texas at Austin in 2020, and earned a BS and an MS in Electrical Engineering from Institut Teknologi Bandung, Indonesia, in 2015 and 2019, respectively.

Arash Fathi is a computational scientist at ExxonMobil with broad interests in computational engineering, mathematics, and HPC. At ExxonMobil Corporate Strategic Research, he led a hardware–algorithm co-design project, developing specialized hardware and tailored algorithms to maximize the computing efficiency of wave simulations. He has also explored the potential of quantum computing for solving PDEs prevalent in the oil and gas industry. Arash has organized several symposiums on novel computational

algorithms for future computing platforms and played a key role in projects involving PDE-constrained optimization, uncertainty quantification, scientific machine learning, emission detection, and digital twins.

Dimitrios Gourounas is pursuing his PhD in the System-Level Architecture and Modeling (SLAM) group at the University of Texas at Austin. His research interests lie in reconfigurable architectures targeting high-performance computing and machine learning workloads. His work includes the design of a reconfigurable accelerator for memory-bound, discontinuous-Galerkin–based PDE solvers and automated frameworks for generating high-efficiency matrix-multiplication units on modern AI-optimized FPGA architectures. He holds a BSc in Electrical and Computer Engineering from the National Technical University of Athens.

Dimitar Trenev is a Computational Scientist at ExxonMobil Technology and Engineering Company, focusing on high-performance computing, quantum computing, and numerical analysis. His work centers on developing and applying advanced computational methods to challenging problems in the energy industry. Dimitar has played key roles in projects involving large-scale seismic inversion, reservoir simulation, uncertainty quantification and other compute-intensive workflows. He also led a collaboration between ExxonMobil and IBM exploring the use of quantum computing for energy applications.

Andreas Gerstlauer received the PhD degree in Information and Computer Science (ICS) from the University of California at Irvine (UCI), Irvine, CA, USA, in 2004. He is a Cullen Trust for Higher Education Professor in the Chandra Family Department of Electrical and Computer Engineering at The University of Texas at Austin (UT Austin), Austin, TX, USA. Prior to joining UT Austin in 2008, he was an Assistant Researcher with the Center for Embedded Computer Systems (CECS) at UCI. His research interests include systems-level design automation, system modeling, design languages and methodologies, and embedded hardware and software synthesis. Prof. Gerstlauer's work was recognized with several best paper awards and nominations from major conferences, such as DAC, DATE, and HOST, as one of the most influential contributions in ten years at DATE in 2008, and as recipient of a 2016–2017 Humboldt Research Fellowship. He serves or has served as an Editor for ACM TECS and TODAES journals, as well as the General or Program Chair for major international conferences such as ESWEEK.

Lizy Kurian John holds the Truchard Foundation Chair in Engineering in the Department of Electrical and Computer Engineering at The University of Texas at Austin. Her research is in the areas of computer architecture, multicore processors, memory systems, performance evaluation and benchmarking, workload characterization, and reconfigurable computing. She has published four books, 300+ refereed journal and conference publications and holds 20 U. S. patents. Prof. John was the Editor-in-Chief of IEEE Micro from 2019-2023. She is an IEEE Fellow, ACM Fellow, AAAS Fellow, and Fellow of the National Academy of Inventors.

Appendix 1

AI	Artificial Intelligence
AMD	Advanced Micro Devices
AMX	Advanced Matrix Extension
API	Application Programming Interface
AVX	Advanced Vector Extension
ASIC	Application-Specific Integrated Circuit
AWS	Amazon Web Services
BER	Bit Error Rate
BGA	Ball Grid Array
CC	CUDA Cores
CCD	Core Chiplet Die
CCX	Core CompleX
CGRA	Coarse-Grain Reconfigurable Arrays
CMOS	Complementary MOSFET
CNT	Carbon Nanotubes
CPU	Central Processing Unit
CTA	Cooperative Thread Arrays
CUDA	Compute Unified Device Architecture
DDR	Double-Data Rate
DLP	Data-Level Parallelism
DRAM	Dynamic Random Access Memory
DSA	Domain-Specific Architecture
DSP	Digital Signal Processing
EDA	Electronic Design Automation
EDR	Extended Data Rate
FDR	Fourteen Data Rate
FPGA	Field-Programmable Gate Array
GDDR	Graphics Double-Data Rate
GEMM	General Matrix-Matrix multiplication
GPC	Graphics Processing Cluster
GPD	Graphics Processing Die (unofficial)
GPGPU	General-Purpose Graphics Processing Unit
GPU	Graphics Processing Unit
HBM	High-Bandwidth Memory
HCC	High Core Count
HDL	Hardware Description Language
HDMI	High-Definition Multimedia Interface
HLS	High Level Synthesis
HMC	Hybric Memory Cube
HPC	High Performance Computing
IBM	International Business Machines corporation
ILP	Instruction-Level Parallelism
IOD	Input/Output Die
IP	Intellectual Property
IPC	Instructions Per Cycle

IPU Intelligence Processing Unit
ISA Instruction Set Architecture
ITRS International Technology Roadmap for Semiconductors
LGA Land Grid Array
MCDRAM Multi-Channel Dynamic Random Access Memory
MCM Multi Chip Module
MIMD Multiple-Instruction Multiple-Data
MISD Multiple-Instruction Single-Data
ML Machine Learning
MMX Multi-Media EXtension
MOSFET Metal Oxide Semiconductor Field-Effect Transistors
NSF National Science Foundation
OIF Optical Internetworking Forum
PAM Pulse-Amplitude Modulation
PCB Printed Circuit Board
PCIe Peripheral Component Interconnect Express
PGA Pin Grid Array
PHY PHYsical layer
RC Resistor-Capacitance
RDU Reconfigurable Dataflow Unit
ROB Re-Order Buffer
RZ Return to Zero
SATA Serial AT Attachment
SDR Single-Data Rate
SerDes Serializer-Deserializer
SIMD Single-Instruction Multiple-Data
SISD Single-Instruction Single-Data
SM Streaming Multiprocessor
SMSP Streaming Multiprocessor Sub-Partition
SP Streaming Processor
SPEC Standard Performance Evaluation Corporation
SoC System on Chip
SRAM Static Random-Access Memory
SXM Server PCI Express Module
TACC Texas Advanced Computing Center
TC Tensor Core
TCO Total Cost of Ownership
TDP Thermal Design Power
TLP Thread-Level Parallelism
TPC Texture Processing Cluster
TPU Tensor Processing Unit
TSMC Taiwan Semiconductor Manufacturing Company
USB Universal Serial Bus
VHDL VHSIC Hardware Description Language
VLIW Very Long Instruction Words
VRM Voltage Regulator Module
WDM Wavelength Division Multiplexing
WSE Wafer Scale Engine
XCD Accelerator Complex Die