Energy Efficient Computing Systems: Architectures, Abstractions and Modeling to Techniques and Standards

RAJEEV MURALIDHAR, The University of Melbourne and Amazon Web Services, Australia
RENATA BOROVICA-GAJIC and RAJKUMAR BUYYA, The University of Melbourne, Australia

Computing systems have undergone a tremendous change in the last few decades with several inflexion points. While Moore's law guided the semiconductor industry to cram more and more transistors and logic into the same volume, the limits of instruction-level parallelism (ILP) and the end of Dennard's scaling drove the industry towards multi-core chips. More recently, we have entered the era of domain-specific architectures and chips for new workloads like artificial intelligence (AI) and machine learning (ML). These trends continue, arguably with other limits, along with challenges imposed by tighter integration, extreme form factors and increasingly diverse workloads, making systems more complex to architect, design, implement and optimize from an energy efficiency perspective. Energy efficiency has now become a first order design parameter and constraint across the entire spectrum of computing devices.
Many research surveys have gone into different aspects of energy efficiency techniques implemented in hardware and microarchitecture across devices, servers, HPC/cloud, data center systems along with improved software, algorithms, frameworks, and modeling energy/thermals. Somewhat in parallel, the semiconductor industry has developed techniques and standards around specification, modeling/simulation, benchmarking and verification of complex chips; these areas have not been addressed in detail by previous research surveys. This survey aims to bring these domains holistically together, present the latest in each of these areas, highlight potential gaps and challenges, and discuss opportunities for the next generation of energy efficient systems. The survey is composed of a systematic categorization of key aspects of building energy efficient systems - (1) *specification* - the ability to precisely specify the power intent, attributes or properties at different layers (2) *modeling* and *simulation* of the entire system or subsystem (hardware or software or both) so as to be able to experiment with possible options and perform what-if analysis, (3) *techniques* used for implementing energy efficiency at different levels of the stack, (4) *verification* techniques used to provide guarantees that the functionality of complex designs are preserved, and (5) *energy efficiency benchmarks, standards and consortiums* that aim to standardize different aspects of energy efficiency, including cross-layer optimizations.

CCS Concepts: • **Hardware → Power and energy**.

Additional Key Words and Phrases: Energy Efficiency, Low Power, Specification, Modeling, Low Power Optimizations, Platform-Level Power Management, Dynamic Power Management

1 INTRODUCTION

The computing industry has gone through tremendous change in the last few decades. While Moore's law [120] drove the semiconductor industry to cram more and more transistors and logic into the same volume, the end of Dennard's scaling [41] limited how much we could shrink voltage and current without losing predictability, and the Instruction Level Parallelism (ILP) wall (David Wall et al. [159]) defined the start of the multi-core

Authors' addresses: Rajeev Muralidhar, rajeevm@ieee.org, The University of Melbourne and Amazon Web Services, Australia; Renata Borovica-Gajic, renata.borovica@unimelb.edu.au; Rajkumar Buyya, rbuyya@unimelb.edu.au, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, Victoria, 3010, Australia.

0360-0300/2022/1-ART1 $15.00
https://doi.org/10.1145/3511094

and tera-scale era [87]. As the number of cores and threads-per-core increased, energy efficiency and thermal management presented unique challenges. We soon ran out of parallelizability as well, both due to limits imposed by Amdahl's law [5] and a fundamental lack of general purpose parallelizable applications and workloads. Fig 1, referenced from Karl Rupp [139] shows 42 years of microprocessor trends taking into account transistor density, performance, frequency, typical power and number of cores. The figure is based on known transistor counts published by Intel, AMD and IBM's Power processors and it also overlays the key architectural inflexion points detailed by Henessey and Patterson in [82]. The graph, as well as studies such as Fagas et al. [56], illustrate that as transistor count and power consumption continues to increase, frequency and the number of logical cores has tapered out. Furthermore, as Moore's Law slows down, while energy efficiency has improved, power density continues to raise across the spectrum of computing devices (Mack et al. [110]). With multi-core architectures reaching its limits, the last few years have seen the emergence of domain specific architectures to attain the best performance-cost-energy tradeoffs for well defined tasks. Systems also evolved from multi-chip packages to system-on-a-chip (SOC) architectures with accelerators like Graphics Processing Units (GPU), imaging, Artificial Intelligence (AI)/deep learning and networking, integrated with high-bandwidth interconnects. Workloads such as deep learning require massive amounts of data transfer to/from memory, leading to the *memory wall*, which is the bottleneck imposed by the bandwidth of the channel between the CPU and memory subsystems. Recent memory technologies like Non-Volatile Random Access Memory (NVRAM), Intel's Optane, Spin Transfer Torque RAM (STT-SRAM), and interfaces such as Hybrid Memory Cube (HMC) [76] and High Bandwidth Memory (HBM) [101] that enable high-performance RAM interfaces have pushed the boundaries of the memory wall. Standards such as PCIe [134] and the more recent Compute Express Link (CXL) [25] are industry standards for integrating accelerators, memory and compute elements. Deep learning has also triggered looking at the traditional von-Neumann architectural model and its limits thereof and several non-von Neumann models have now gained popularity, such as those based on dataflow, spiking neural networks, neuromorphic computing and bio-inspired computing (Ganguly et al. [65]).

The nature of computing systems has transformed across the spectrum of devices, from being pure compute-based to being a mixture of CPUs, GPUs, accelerators and Field Programmable Gate Arrays (FPGA). Heterogeneous computing capabilities are now also available on "edge devices" such as the Raspberry PI, Google's Coral Tensor Processing Unit [93] and Intel's Movidius [27]. As devices have shrunk, the industry is struggling to eliminate the effects of thermodynamic fluctuations, which are unavoidable at lower technology nodes (sub-10nm scale). Even as architectures become more energy efficient, recent research has shown that workloads such as deep learning consumes significant energy (Schwartz et al. [142]). Ironically, deep learning was inspired by the human brain, which is remarkably energy efficient. Shrinking and extreme form factors, diverse workloads and computing models have thus greatly accelerated the limitations imposed by fundamental physics and architectural, power and thermal walls. Designing energy efficient systems present unique challenges due to the domain-specific processing capabilities required, heterogeneous nature (workloads that can run on CPUs, GPUs or specialized chips), system architecture (high bandwidth interconnects for the enormous amounts of data transfer required) and extreme form factors (with devices capable of doing Tiny ML, which is the ability to do machine learning in less than 1 mW of power [154]). Systems have become complex to architect, design, implement and verify, with energy efficiency transforming into a multi-disciplinary art requiring expertise across hardware/circuits, process technology, microarchitecture, domain-specific hardware/software, firmware/micro-kernels, operating systems, schedulers, thermal management, virtualization and workloads, only to name a few. While specific end systems (IoT, wearables, servers, HPC) need some techniques more aggressively than others due to the constraints, the underlying energy efficiency techniques tend to overlap across systems and hence we need to take a holistic view as we look to improve and architect next generation systems.

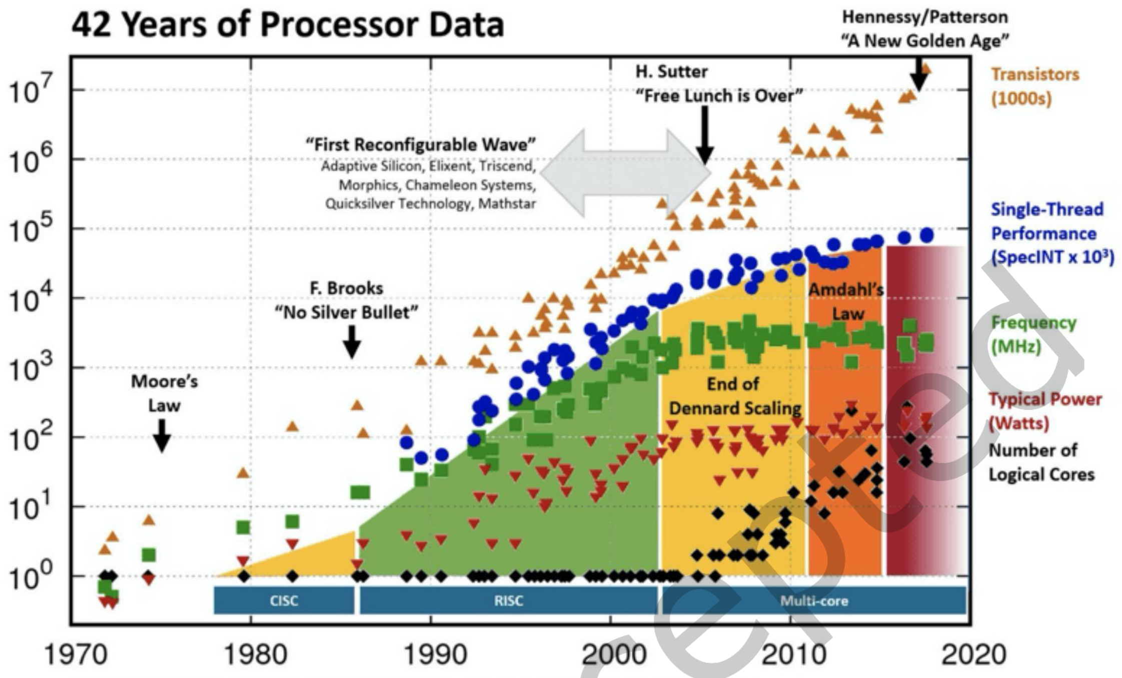

Fig. 1. Microprocessor trend data during 1972-2020. Hennessey and Patterson, Turing Lecture 2018 [82], overlaid with "42 Years of Processors Data" [139]

1.1 Related Surveys

Several research surveys have looked at energy efficiency techniques used in hardware, circuits/RTL, microarchitecture and process technology, across the spectrum of computing systems. Another area of active research has been around modeling and simulation of power, performance and thermals for individual hardware components (processors, memory, GPUs, and accelerators), system-on-a-chip (SOC) and the entire system. In parallel, techniques and standards have evolved in the semiconductor and Electronic Design and Automation (EDA) industry around specification and verification of large, complex chips. The industry has also collaborated to build highly optimized software/system level techniques and has defined energy related benchmarks, regulations and standards. This survey brings the domains together and presents the latest in each area, highlights potential gaps/challenges, and discusses opportunities for next generation energy efficient systems. Some current related research surveys are listed in Table 1 - this list is, by no means exhaustive, but merely points to some key surveys or books in respective areas.

1.2 Need for a holistic approach to energy efficiency

Designing energy efficient systems is now a virtuous cycle and cannot be done in hardware or software alone, or in isolation of other domains or components due to diverse architectures, hardware/software interactions and varied form factors. Power-related constraints have to be imposed through the entire design cycle in order to maximize performance and reliability. In the context of large and complex chip designs, reliability and minimizing power dissipation have become major challenges for design teams, which have dependencies on software as well.

Table 1. Summary of Energy Efficiency Related Surveys

Topic	Key survey or book
Energy efficiency/sustainability, metrics in cloud	Mastelic et al. [114], Gill et al. [67]
Energy efficiency techniques in hardware, circuits	Venkatachalam et al. [158]
Hardware techniques for energy efficiency in CPUs, GPUs	Mittal et al. [117]
Energy Efficiency of compute nodes	Kaxiras and Martonosi [95]
Energy efficiency at data center level	Barroso and Hoelzle [16]

Creating optimal low-power designs involves making trade-offs such as timing-versus-power and area-versus-power at different stages of the design flow. Additionally, trade-offs that are applied at a certain phase of the chip have implications on future software techniques that push the boundaries of what the chip has been designed to do. In many cases, if certain design choices are known ahead of time, specific workloads will benefit from them with respect to energy efficiency.

Feedback from running real workloads on current generation systems is used in architecting next generation systems. Architects need to perform "what-if" analysis using different algorithmic knobs at different stages as illustrated in Figure 2. For example, it is important to simulate different techniques of frequency state selection, their transition latencies and the impact of these states on different workloads. Adding or removing power efficiency features can make or break the chip launch timeline, which could have market implications and could impact the company's future itself. The ability to model power consumption of different hardware components across generations of hardware in a standardized manner has become a key focus of industry efforts such as the IEEE P2416 standard for power modeling [14]. As another example, the ability to run a real workload on a simulated future design and making use of new power/performance features is an important to expose bugs in the underlying hardware. If these bugs are found later in post-silicon, it could cause unacceptable delays due to a hardware re-spin. Such scenarios need information exchange across layers of the hardware-software stack - such as new frequency states being exposed, how the OS and higher layers can make use of it and the ability to model performance gain therein. The goal of the recent IEEE P2415 [13] is to build cross layer abstractions such as this to facilitate easier information exchange across different layers of the stack as well as different phases of architecture, modeling and verification.

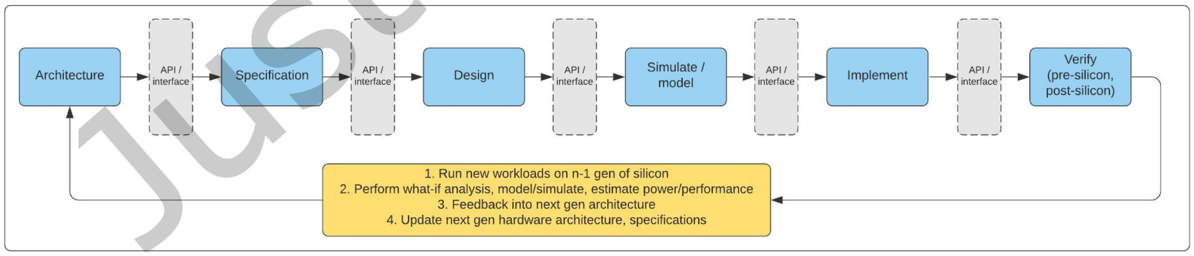
Architecture
API / interface
Specification
API / interface
Design
API / interface
Simulate / model
API / interface
Implement
API / interface
Verify (pre-silicon, post-silicon)
1. Run new workloads on n-1 gen of silicon
2. Perform what-if analysis, model/simulate, estimate power/performance
3. Feedback into next gen architecture
4. Update next gen hardware architecture, specifications

Fig. 2. Phases of energy efficient system design

Energy efficiency in HPC systems has also become important of late. The Energy Efficient HPC (EEHPC) [70] Consortium is a group focused on driving implementation of cross layer energy conservation measures and energy efficient designs HPC systems. The working groups cover several aspects of energy efficient HPC - infrastructure (cooling, highly efficient power sources), algorithms and runtime (energy and power aware job scheduling), and specifications (Power API). Similarly, the Global Extensible Open Power Manager (GEOPM) (Eastep et al. [51]) is an open source runtime HPC framework for enabling new energy management strategies at the node, cluster and data center level.

Holistic energy efficiency across layers and across phases of evolution is crucial and cannot ignore any of the platform components; neither can it be done in hardware or software alone and must encompass all aspects of energy efficient system design - from architecture to modeling/simulating to implementing and optimizing each component as well as the system as a whole.

1.3 Contributions of This Survey

Previous surveys have looked at energy efficiency in hardware/microarchitecture, at different layers (software and algorithms) and at different systems (devices, servers, cluster and cloud). In such surveys, it is assumed that hardware architectures and features of energy efficiency in hardware evolve on their own, and software then takes the best possible approach by designing energy aware algorithms. Additionally, several industry trends, benchmarks, standards and consortiums related to energy efficiency have not been surveyed in detail. As systems become complex, energy efficiency considerations must be imposed across the entire cycle - from hardware/system architecture, design, specification, modeling/simulation, to higher layers of software algorithms that use these features to optimize the system. With that goal in mind, this survey is composed of a systematic categorization of the following energy efficiency methods across the wide spectrum of computing systems:

(1) *Architectural Techniques for Energy Efficiency*: Energy efficiency techniques can be implemented at different levels of the hierarchy - circuit/RTL, microarchitecture/architecture, CPU, GPU or other hardware blocks/accelerators.
(2) *System level Techniques for Energy Efficiency*: At the system level, underlying architectural and microarchitectural techniques are used at different levels of the software hierarchy (firmware, operating system and applications) and energy efficiency is implemented at the entire system.
(3) *Specification* of the energy efficiency technique: This involves specifying the technique in a standardized manner, and includes cross-layer abstractions and interfaces (hardware, hardware-firmware, firmware-OS, and OS-applications).
(4) *Modeling and Simulation*: Given a set of techniques for energy efficiency, this involves modeling/simulating the functionality/technique of the component or set of components, and run real workloads (or traces of a real workload).
(5) *Verification*: Given each of the above, this involves verifying the energy efficiency of the entire system with different thermal constraints, real workloads and different form factors.
(6) *Energy Efficiency Benchmarks, Standards and Consortiums*: Recent trends at standardizing different aspects of energy efficiency at IEEE and other industry consortiums is an important area of research/industrial collaboration.

1.4 Organization of this Paper

This paper is organised as follows:

(1) Section 2 elaborates on recent architectural inflexion points, evolution of energy efficiency features and upcoming trends.
(2) Section 3 discusses energy metrics, power/thermal dissipation and fundamental energy efficiency techniques. This section is offered as online appendix A.
(3) Section 4, offered as online appendix C, discusses *architectural and microarchitectural techniques* used in CPUs, GPUs, memory and domain-specific accelerators.

(4) Section 5, offered as online appendix D, discusses *specification* of power management techniques. Being able to capture the power intent in a formal description is key to design, modeling/simulation as well as verification of the system as a whole. This is also crucial for the Electronic Design Automation (EDA) industry, IP-reuse and building complex systems. We survey specifications and abstractions at different levels of the hierarchy.
(5) Section 6 covers *modeling and simulation* of power, performance and thermal dissipation across processors, GPUs, accelerators, SOC and complete systems. We describe state of the art modeling and simulation tools and technologies in use today. This section is offered as online appendix E.
(6) Section 7, offered as online appendix F, covers *verification* of power management design and techniques in large SOCs and systems across gate, RTL/architectural and system levels.
(7) In Section 8, we cover *system and software techniques* used for energy efficiency. In this, we cover energy efficiency techniques implemented in firmware, device drivers, and operating systems such as Linux and Windows. Where relevant, we also provide methods used across different classes of designs, like mobile processors, data centers, servers, etc.
(8) Section 9 discusses recent advances in system level energy efficiency implemented across real products from Intel, AMD, ARM, including the emergence of custom-built high performance ARM designs such as AWS Graviton3, and ARM in HPC.
(9) Section 10, offered as online appendix G, surveys *energy efficiency related benchmarks, standards and consortiums* that aim to address energy efficiency through regulations, standardization of abstractions, energy/performance models and cross-layer optimizations.
(10) In Section 11, we will discuss the road ahead for next generation of energy efficient systems and in Section 12, we offer our summary and conclusions.

2 ARCHITECTURAL TRENDS AND SYSTEM LEVEL ENERGY EFFICIENCY

John Hennessy and David Patterson, in their recent ACM Turing award lecture and publication [82] trace the history of computer architecture and touch upon some of the recent trends, including domain-specific architectures (DSA), domain-specific languages (DSL) and open instruction set architectures such as RISC-V (Patterson et al. [133]). In this section, we elaborate on some of the key observations highlighted in Hennessey and Patterson [82], look at how the underlying architecture of computing systems has transformed in the last couple of decades due to several fundamental laws and limits, and focus on system level energy efficiency. Markov [113] discusses some of these trends as well, specifically with regard to *limits on fundamental limits to computation.* We will look at the trends, inflexion points and their respective impact on system level energy efficiency detailed in Table 2. This list is, by no means exhaustive, however it aims to illustrate the influence of key inflexion points on energy efficiency.

2.1 Moore’s Law, Dennard Scaling and Instruction-Level Parallelism

MooreâĂŹs Law [120] has enabled the doubling of transistors on chips approximately every 18 months through innovations in device, process technology, circuits and microarchitecture, and this has in turn spurred several innovations in system software, applications, thermal management, heat dissipation, advanced packaging and extreme form factors. It is interesting to note that Gordon Moore had himself predicted a slowdown in 2003 as CMOS technology approached fundamental limits (Moore [121]). In addition to this, there have been other important laws that have shaped computer systems. One such is Dennard Scaling [41]. Robert Dennard observed in 1974 that power density stays constant as transistors get smaller. The key idea was that as the dimensions of a device go down, so does power consumption. For example, if a transistorâĂŹs linear dimension shrank by a factor of 2, that gives 4 times the number of transistors. If both the current and voltage are also reduced by

Table 2. Trends in system architecture and energy efficiency

Architectural Trends	Energy Efficiency Features
Moore's Law[120], ILP wall (David Wall [159]), Dennard Scaling[41]	Increased performance via superscalar, VLIW arch, Clock/power gating, processor, cache, memory sleep states, Dynamic Voltage Frequency Scaling (DVFS), power delivery improvements
Multi-cores[87], Amdahl's limit (Amdahl [5], Hill et al. [84])	OS guided / controlled sleep states, fine grained clock/power gating, per-core, per-module DVFS, on-die voltage regulators
Memory wall, Phase Change Memory (PCM) [100], Magneto-resistive RAM (MRAM)[68], Spin Transfer Torque RAM (STT-RAM)[98], Resistive RAM (ReRAM) [138]	Memory DVFS (Deng et al. [40], David et al. [36]), system level techniques for memory power management[7]
Domain-specific architectures such as programmable network processors (Li et al. [105]), deep learning chips (Jouppi et al. [93]), Intel Mobileye [33]	Chip/IP-level clock/power gating, DVFS
Dark silicon challenges (Esmaeilzadeh et al. [54])	Fine grained power domains and islands
High bandwidth interconnects	Standards like CXL[25] and PCIe [134]
Non von-Neumann architectures (David Culler [35], SpiNNaker[136], Thakur et al. [152])	Energy-aware dataflow architectures
Combining von Neumann and non-von Neumann chips (Nowatzki et al. [125])	Emerging area, mix of different techniques
Power delivery miniaturization	On-die/chip voltage regulators, software control, reconfigurable power delivery (Lee [103])
Programmable architectures - FPGAs	Energy-aware FPGAs, still in nascent stage
Energy Proportional Computing[17]	Energy-aware data centers, system components
Near/sub-threshold voltage designs[60][119], 3D stacking[107], and chiplets[31]	Ultra low voltage designs, Thermal algorithms
Thermodynamic computing [26], Landauer Limit [99] and Quantum Computing [72]	Emerging areas, system architectures unclear / evolving

a factor of 2, the power it used would fall by 4, giving the same power at the same frequency. While this law held, smaller transistors ran faster, used less power, and cost less. During the last decade of the 20th century and the first half of the 21st, computer architects made the best use of MooreâĂŹs Law and Dennard scaling to increase resources and performance with sophisticated processor designs and memory hierarchies that exploited instruction level parallelism (ILP). Dennard scaling, however, soon ended because current and voltage could not keep dropping while remaining dependable. Recently, near-threshold and sub-threshold voltage technologies [126] are attempting to push these boundaries.

Instruction Level Parallelism (ILP) can be implemented through several different techniques, and the amount of ILP in programs can be application specific. Scientific computing, graphics applications may exhibit high ILP whereas workloads such as cryptography may not. Micro-architectural techniques that are used to exploit ILP include:

(1) *Instruction pipelining*: Here the execution of multiple instructions can be partially overlapped thereby reducing the overall Clocks-per-instruction (CPI).
(2) *Superscalar execution*, *Very Long Instruction Word (VLIW)*, *Explicitly Parallel Instruction Computing (EPIC)*: In these, multiple execution units are used to execute multiple instructions in parallel. In superscalar designs, multiple instructions can be executed in a clock cycle by dispatching multiple instructions to different execution units on the processor (Palacharla et al. [130]). There were several variations of this architecture as well, such as the Ultrascalar (Henry et al. [83]) and Multiscalar processors (Sohi et al. [146]). In Very Long Instruction Word (VLIW) designs, one VLIW instruction encodes multiple operations with at least one operation for each execution unit. Efficiency of VLIW architectures relies heavily on compilers to correctly schedule operations (Fisher [59]). EPIC architectures evolved from VLIW (Schlansker and Rau [140]), but retained many concepts of superscalar architectures, and formed the foundation of many generations of Intel processors, including Itanium. While VLIW and EPIC architectures did not gain popularity in mainstream processors, some domain-specific chips have used VLIW architectures. For example, AMD's TeraScale GPU [1] was based on VLIW and more recently, Intel's Movidius [27] is a VLIW-based low power inference chip.
(3) *Out-of-order execution*: In this, instructions execute in any order as long as they do not violate data dependencies. This can be implemented on any of the above architectures (pipeline-based on superscalar).
(4) *Register renaming* is used to avoid unnecessary serialization of program operations when hardware registers are used to store program operands. This technique, originally devised as Tomasulo's algorithm [155], is widely used in almost all processor architectures today.
(5) *Speculative execution*: This allows the execution of instructions before being certain whether the instruction would be executed, and is implemented by using techniques such as control flow speculation, memory dependence prediction, etc.
(6) *Branch prediction*: This is used to avoid stalling, and is heavily used with speculative execution.

While some of these ILP improvement techniques ran out of steam due to various reasons [159], many of these techniques are still used today in modern processors. Even as ILP limits were worked around through afore mentioned techniques, the industry started to switch from single energy-hogging processors to multiple efficient processors or many cores per chip, ushering in the many/multi-core era. Recent times have also seen hybrid designs that combined low power/low performance and high power/high performance cores, like ARM's BIG.LITTLE architecture [156] and the recent Intel Lakefield chip [31]. Hameed et al. [80] explore the sources of performance and energy overheads of common workloads on a general purpose CMP system, and look into methods to eliminate these overheads by customizations to CPU cores. The general approach is that as ASICs are significantly more energy efficient than general purpose CMP systems, achieving comparable energy reduction requires algorithm-specific optimizations, such as specialized functional units. Even as Moore's Law slows down, transistor density scaling has continued to be exponential, as illustrated in Figure 1. Etiemble [55] describes evolution of CPUs over the last 45 years.

2.2 Multi-core era, Amdahl's law

There were limits to the multi-core era too, as dictated by Amdahl's law [5], which states says that the theoretical speedup from parallelism is limited by the sequential part of the task; so, for example, if $\frac{1}{8}th$ of the task is serial, the maximum speedup is 8 times the original performance, even if the rest is easily parallelizable and we add any number of processors. Hill et al. [84] elaborate on the impact of this law on multi-core chips.

Let speedup be the original execution time divided by an enhanced execution time. Amdahl's law states that if we enhance a fraction f of a computation by a speedup S, the overall speedup is:

$$\text{Speedup}_{\text{enhanced}}(f, S) = \frac{1}{(1-f) + \frac{f}{S}}$$

More specifically, if we are using n processor cores:

$$\text{Speedup}_{\text{parallel}}(f, n) = \frac{1}{(1-f) + \frac{f}{n}}$$

2.3 The Problem of Dark Silicon

For decades, Dennard scaling permitted more transistors, faster transistors, and energy efficient transistors with each new process node, justifying the costs required to develop each new process node. Dennard scalingâĂŹs failure led the industry to race down the multicore path, which for some time permitted performance scaling for parallel and multitasking workloads, permitting the economics of process scaling to hold. The next problem that all chips have had to deal with over the last decade is that of *dark silicon*. Several studies, like Esmaeilzadeh et al. [54] show that regardless of chip organization, architecture or topology, the runtime software (at OS/firmware/hardware levels) must essentially shut off several parts of the silicon due to fundamental power and thermal limits. This part of the hardware is termed as *dark silicon*.

Due to dark silicon, even if the increased number of transistors is used to implement additional processor cores, not all available cores can be powered at the same time, to avoid overloading the thermal budget of the chip. Recent designs, especially in power-constrained devices, use dedicated co-processors to run particular tasks in a power-optimized fashion that can be turned off when not in use. These designs rely heavily on aggressive power gating, dynamic voltage and frequency scaling and are organized into fine-grained power and voltage domains. Software and operating system guided energy efficiency is all the more paramount since higher layer of intelligent software should devise strategies for aggressively powering on/off different components of the system based on the usage scenario.

Other techniques are being explored as well to mitigate the effect of dark silicon. Asynchronous circuits is one such technique. While synchronous circuits use a single global control signal, which is active at times even when there is no processing needed in a particular pipeline, asynchronous circuits are only active when workloads are in local pipelines. Techniques like desynchronization are used to convert a synchronous circuit into an asynchronous one. Boundary synchronization is another technique that is used to perform synchronization of signals as they cross clock and voltage domains. Krstic et al. [97] provide a detailed survey of *Globally Asynchronous, Locally Synchronous (GALS) circuits*. Another method for reducing power consumption in asynchronous circuits is *energy modulated computing* (Yakovlev [161]). Here, asynchronous logic uses the power available to it and adjusts the performance to meet that energy level.

2.4 Memory wall, improved memory technologies

Dynamic Random Access Memory (DRAM) has been the mainstay of memory systems over the last few decades across almost all computing systems. As applications/workloads evolve, the data set sizes have rapidly grown, along with an increase in the need for rapid analysis of such data. Moving data from memory to the processing unit and back turned out to be a limiting factor for both performance and power consumption, especially for workloads such as deep learning that involve repetitive operations on large data sets. This limiting factor is termed the *von Neumann bottleneck*, or *memory wall*, which is essentially the bottleneck imposed by the bandwidth of the channel between the CPU/GPU or accelerator and the memory subsystem. While GPUs were a good fit for the computational elements of deep learning algorithms, the limitations from the memory wall proved to be the next obstacle to overcome. For these real-time big data workloads, DRAM was not big enough and traditional storage was not fast enough.

DRAM scaling faces significant challenges; Mandelman et al. [111] describe how the scaling techniques used in earlier generations are encountering limitations that require major innovations. Mutlu [122] describes in detail the demands and challenges faced by the memory system, and examines some recent research and industrial trends to overcome these challenges, primarily around new DRAM architectures, better integration of DRAM with the rest of the system, designing new memory systems that employ emerging memory technologies and providing predictable performance and QoS as workloads become more data-movement intensive.

Improvements in memory technology over the last two decades has focused on newer memory technologies, improving energy efficiency of the memory systems, and more recently, embedding logic closer to, or along with, memory. Due to the limitation of the number of pages, we describe energy efficiency techniques used in memory systems here. The online appendix B discusses newer memory technologies such as PCM, MRAM, STT-RAM, ReRAM, and techniques such as processing-near-memory and processing-in-memory.

2.4.1 Energy Efficiency Techniques for Memory Systems. Several techniques to optimize DRAM-based systems have been explored and implemented in research and commercial systems, a few of which are highlighted here. Lee et al. [102] describe the power and performance relationship of modern DRAM devices. In [73] and [74], Ha et al. provide an exhaustive analysis of state-of-the-art DRAMs, LPDDR4, HBM and describe the design and implementation of several energy reduction techniques by optimizing accesses (Half page DRAM technique reduces energy by 38%) and refresh cycles (Charge Recycling Refresh technique conserves 32% energy and Smart Refresh improves this even further). Similarly, Liu et al. [109] propose RAIDR (Retention-Aware Intelligent DRAM Refresh), a mechanism that can identify and skip unnecessary refreshes using knowledge of cell retention times. This is done by grouping DRAM rows into retention time bins and applying a different refresh rate to each bin, thereby reducing the refresh cycles of less frequently used bins/cells. This technique achieves an impressive 74% refresh reduction leading to a DRAM power reduction of 16%. Chang et al. [21] explore reducing the DRAM supply voltage more aggressively to reduce energy consumption by studying about 125 real LPDDR3 DRAM chips. They find that while reducing supply voltage introduces bit errors, they can be avoided by increasing the latency of key DRAM operations such as activation, restoration and precharge. They also propose a technique called Voltron, which uses a performance model to determine how much the supply voltage can be dropped without errors. These improvements outperform previous DRAM DVFS algorithms for memory intensive workloads.

Going beyond the DRAM subsystem itself, several approaches to using DVFS have been researched and implemented, both for the memory subsystem itself, as well as coordinated DVFS across CPU, memory and other subsystems. The advent of Memory DVFS, which is the ability to dynamically scale the voltage and frequency of the memory subsystem, independent of CPU DVFS, allowed for optimizing the system as a whole from an energy efficiency perspective. There have been several approaches to this. One of the earliest approaches was to adjust CPU DVFS based on memory accesses, so that the memory subsystem could enter idle low power states if the CPU was busy executing computations and there were no pending memory operations. For example, Liang et al. [106] demonstrated how performance monitoring counters could be used to alter CPU DVFS and help lower system energy consumption in an embedded device. Howard et al. [36] was one of the earliest works in memory DVFS that demonstrated a simple control algorithm that adjusts memory voltage and frequency based on memory bandwidth utilization, and was implemented on a real system. Deng et al. [40] describe MemScale, a technique that leverages dynamic profiling, performance and power modeling, DVFS of the memory controller, and DFS of the memory channels and DRAM devices, all done independent of CPU DVFS. MemScale is guided by an operating system policy that determines the DVFS/DFS mode of the memory subsystem based on the current need for memory bandwidth, the potential energy savings, and the performance degradation that applications are willing to withstand. Bianchini et al. [39] describe CoScale, for coordinating memory and CPU DVFS in server systems. CoScale relies on execution profiling of each core through performance counters, and models of core and memory performance and power consumption. It uses fixed-size epochs (matching an OS time quantum). In each epoch, there is a system profiling phase followed by the selection of core and memory subsystem frequencies that minimize total system energy while maintaining performance within the target bound. The advent of GPUs and memory bandwidth hungry workloads extended this concept to coordinated CPU, GPU and memory DVFS using performance and power monitoring counters. Chau et al. [22] describe a scheduling algorithm that optimizes the CPU and GPU DVFS states based on currently running workloads and their predicted runtime. Most recent systems from Intel, AMD, NVIDIA, etc. support independent DVFS for CPU, GPU, memory, along with techniques such as dynamic memory throttling (inserting idle cycles between reads and writes). Mittal and Vetter [118] present a survey of these CPU-GPU coordination techniques.

2.4.2 Newer Memory Technologies. This section, produced as online appendix B.1, discusses newer memory technologies such as Phase Change Memory (PCM), Magneto-resistive RAM (MRAM), Spin Transfer Torque RAM (STT-RAM), and resistive RAM (ReRAM) are described in online Appendix B.1.

2.4.3 Processing In Memory (PIM). With recent advances in existing memory systems, and the advent of newer memory techniques, integration of memory and logic, an old idea, has re-emerged. Mutlu et al. [123] describe this in exhaustive detail and we use the same terminology here. Broadly this is called *processing-in-memory* and it involves placing computation mechanisms in or near where the data is stored (memory chips, or the logic layer, or the memory controllers, etc.), so that data movement is reduced or eliminated. This section is described in online appendix B.2

2.5 Domain Specific Architectures and the limits of chip specialization

Domain Specific Accelerators (DSA) are architectures that are tailored to a specific problem domain and offer significant performance and efficiency gains for that domain. Some examples are GPUs, neural network processors for deep learning and programmable network processors for high speed packet forwarding in software-defined networks (SDNs).

DSAs for high speed packet processing accelerators have been implemented over the decades starting with ASICs/DSPs to FGPAs and dedicated programmable network processors. In these core internet routers and switches, data plane algorithms must be implemented in hardware in order to do packet processing at line rate of 100s of Gigabits/sec, and they must also be programmable. Several generations of such programmable networking devices form the internet backbone today. Li et al. [105] discuss P4GPU for high speed packet processing. The P4 language is an emerging domain-specific language for describing the data plane processing at a network device. P4 has been mapped to a wide range of forwarding devices including NPUs, programmable Network Interface Chips (NICs) and FPGAs. In Sivaraman et al. [145], the authors show how to program data-plane algorithms in a high-level language and compile those programs into low-level microcode that can run on emerging programmable line-rate switching chips using the notion of *packet transactions*, an atomic packet-processing sequence of code.

Domain specific accelerators for camera/imaging, deep learning, amongst others, have been implemented in several industrial devices in the last 15 years. For example, Qualcomm's Snapdragon SOC contains a Hexagon cores [91] for AI processing in camera, voice, VR and gaming applications. PowerVR's Neural Net Accelerator

(NNA) [151] is used in several phones/devices. Similarly, Apple's Neural Engine is an AI accelerator core within the Apple recent Bionic SoC [150]. Google has built the Tensor Processing Unit (TPU) (Jouppi et al. [93]) that is an ASIC optimized for machine learning and is specifically designed for its TensorFlow framework, which is extensively used for CNNs. Similarly, Intel's Myriad 2 [27] is a many-core VLIW AI accelerator complemented with video fixed function units and is reported to be capable of operating in the sub-1W range and delivering 300 GOPS or just over 1 TOPS per watt [28]. Intel's Mobileye's EyeQ Ultra [33] is a processor specialized for vision processing for self-driving cars.

The trend of domain specific architectures continues especially in the areas of AI/ML, edge computing for vision/recognition, low power audio processing, and several others. However, much of the benefits of chip specialization stem from optimizing a computational problem within a given chipâĂŹs transistor budget. As detailed in Fuchs and Wentzlaff [61], for 5nm CMOS chips, the number of transistors can reach 100 billion; however not all of them can be utilized due to the challenge of dark silicon. Chips will be severely limited by thermal budgets. This will also cause stagnation of the number of useful transistors available on a chip, thereby limiting the accelerator design optimization space, leading to diminishing specialization returns, ultimately hitting an *accelerator wall* in the near future.

2.6 SOC Integration, evolution of software power management

Computing systems have transformed from predominantly CPU-based systems to more complex system-on-a-chip (SOC) based ones with highly integrated single/multi-core CPUs, newer memory technologies/components, domain-specific accelerators for graphics, imaging, deep learning, high speed interconnects/ peripherals and multi-comms for connectivity. The more recent Compute Express Link (CXL) [25] is an industry standard to integrating accelerators, memory and compute elements. Similarly, PCI [134] have emerged as standards for high bandwidth, low power interconnects between CPU cores, memory and accelerators. As systems have become more capable in terms of their performance and capabilities, their energy consumption and heat production has also grown rapidly. The explosion of highly powerful and complex SOCs across all kinds of computing systems have surpassed the rate of evolution of software thereby presenting unique challenges to meet the power and thermal limits. From a systems perspective, such platforms present wide ranging issues on SOC integration, power closure/verification, hardware/software power management and fine-grained thermal management strategies. *This is perhaps a unique phase in the semiconductor industry which has always prided on a specific cadence of hardware growth and the assumption that software will always be ready to meet the requirements of the hardware.* In order to meet the needs of complex SOCs, operating system and software-guided power management infrastructures, frameworks, and algorithms have evolved different hardware/software techniques. Embedded real time operating systems and open source operating systems such as Linux have developed several software techniques and frameworks to perform aggressive system level power, performance and thermal management such as tickless scheduling (Siddha et al. [148]), DVFS frameworks [43], idle power management (Pallipadi [131]), active/runtime power management [48], and various energy efficient system standby states. Windows has also standardized Connected Standby, Modern standby [34] and several more energy efficiency strategies and algorithms to manage idle and active workloads. Both Linux and Windows kernel device drivers also implement aggressive energy management techniques at the system level through workload aware PCI link power management (to put internal PCI links in low power state dynamically), network/communications power management, only to name a few. Thus, software guided and software controlled energy efficiency have gained significant importance for complex SOCs and systems.

2.7 Advent of non-von Neumann architectures

Traditional architectures have largely followed the von Neumann computing model. One of the major deviations from von Neumann architectures, *dataflow machines* were proposed a couple of decades ago (Culler [35]) and Veen [157]. However, they were severely limited by the availability of data movement infrastructures, effective software parallelism and functional units in hardware (Gurd et al. [71]). Thus, dataflow machines did not see commercial deployment for general purpose computing. However, dataflow architectures have been used significantly in implementing specialized hardware for digital signal processing (DSP), graphics processing, imaging/video engines, etc.

More recently, dataflow or near-dataflow architectures have been applied to AI/ML workloads. Deep learning workloads are largely free of control flow and are instead steered by availability of data for executing a predetermined set of operations. Embodying this algorithmic characteristic, dataflow based systems are being developed which are completely controlled by data flow and not by control. The algorithmic parallelism that such workloads exhibit makes them perfect candidates for dataflow modeling which has the potential of reducing energy consumption by orders of magnitude as compared to their execution on control flow based systems. Most architectures for deep learning acceleration work towards optimizing the data size or the number of operations to be performed which may hold relevance for better performance but do not necessarily translate into energy efficiency. As discussed in Yang et al. [162], there are two reasons to this, data movement and not the computation requires more energy and that the flow of data along with the levels in memory hierarchy have a major impact on energy efficiency.

Chen et al. [24] propose Eyeriss, an optimized algorithmic dataflow for CNNs by exploiting local data reuse and optimization of intermediate data movement. Tetris (Gao et al. [66]) uses the dataflow model of Chen et al. [24] along with scheduling and partitioning in software to implement CNN acceleration in HMC. In Farabet et al. [57], the authors present Neuflow, a compiler that transforms high level dataflow graphs into machine code representations. Li et al. [104] present SmartShuttle, a framework that adaptively switches among different data reuse schemes and the corresponding tiling factor settings to dynamically match different convolutional layers. Its adaptive layer partitioning and scheduling scheme can be added on existing state-of-the-art accelerators to enhance performance of each layer in the network. The industry has also seen some innovative products in this space. Wave Computing (Chris Nichol [124]), Chaudhuri et al. [23]) present an implementation of a dataflow architecture as an alternative to train and process DNNs for AI especially when models require a high degree of scaling across multiple processing nodes. Instead of building fast parallel processors to act as an offload math acceleration engine for CPUs, Wave Computing's dataflow machine directly processes the flow of data of the DNN itself.

Spiking Neural networks (SNN) are another form of brain-inspired networks that takes a step closer in mimicking the working of the brain. The pulse width and timing relationship between signals adds to the value of the data being computed and SNNs precisely work with these kind of network parameters. Thus, implementations of such neuromorphic loads fall in the larger circle of non-von Neumann computing that are largely asynchronous event-driven systems.

Some of the implementations of the SNN computing acceleration include IBM TrueNorth (Merolla et al. [3]), SpiNNaker from the University of Manchester (Plana et al. [62]), Intel's Loihi (Davies et al. [37]) and many more. IBM TrueNorth (Merolla et al. [3]) is a many-core processor network on a chip design, with 4096 cores, each one having 256 programmable simulated neurons for a total of just over a million neurons. In turn, each neuron has 256 programmable "synapses" that convey the signals between them. Since memory, computation, and communication are handled in each of the 4096 neurosynaptic cores, TrueNorth circumvents the von Neumann architecture bottleneck and is very energy-efficient, consuming 70 milliwatts with a power density that is 1/10,000th of conventional microprocessors. SpiNNaker (Plana et al. [62]) is a digital neuromorphic neural array designed for scalability and energy efficiency by incorporating brain-inspired communication methods. It can be used for simulating large neural networks and performing event-based processing for other applications. Each node is made of 18 ARM968 processor cores, each with 32 kilobytes of local instruction memory, 64 kilobytes of local data memory, packet router, and supporting circuitry. A single node consists of 16,000 digital neurons consuming 1W of power per node. Ganguly et al. [65] discuss these and other non-von Neumann architectures in more detail with respect to their energy efficiency.

2.8 Architectures mixing von Neumann and non-von Neumann chips

With non-von Neumann computing models gaining traction, mixing von Neumann and non-von Neumann architectures/computational models is also being explored. Nowatzki et al. [125] discussed that if both out-of-order and explicit-dataflow were available in one processor, the system can benefit from dynamically switching during certain phases of an application's lifetime. They present analysis that reveals that an ideal explicit-dataflow engine could be profitable for more than half of instructions, providing significant performance and energy improvements. More recently, Intel's Configurable Spatial Accelerator (CSA) [30] is an effort to mix von Neumann and non-von Neumann processors. The core idea is that there is basic control of data flow (the traditional von Neumann model) but there is also a configurable way to program dataflow parts of the computations. The system takes the dataflow graph of a program (created by compilers) before it is translated down to a specific processorâĂŹs instruction set, data storage, and lays down that data flow directly on a massively parallel series of compute elements and interconnects between them. The architecture presents very dense compute and memory, and also very high energy efficiency because only the elements needed for a particular dataflow are activated as a program runs, with all other parts of the chip going idle. The configurable part is that the system will have many different CSA configurations tuned to the dataflows of specific applications (single precision, double precision floating point, mixture of floating point and integer). This is intended to be the first exascale machine deployed in the USA by 2021. It is largely expected that future architectures will be a mix of CPUs, GPUs and domain-specific accelerators, each optimized for a specific function, as shown in Figure 3. Such diverse architectures also make it imperative for the industry and academia to come together and define uniform interfaces across hardware and software to model, estimate, measure and analyze power, performance and energy consumption across layers. Efforts such as the IEEE Rebooting Computing initiative [85] could be extended to consider this aspect as well in addition to its existing charter.

2.9 Power delivery miniaturization, reconfigurable power delivery networks

From a power delivery perspective, voltage regulators have shrunk and SOCs today have on-die voltage delivery that can deliver fine grained power to different parts of the chip, all of which are controlled through hardware and firmware (and in some architectures, to the OS level as well). SOCs are organized into "power domains" or "voltage islands", which allow for several individual areas of the chip to be powered on/off or run at different clock frequencies/voltage. Haj-Yahya et al. [78] review on-chip, integrated voltage regulator (IVRs) and presents a thorough and quantitative evaluation of different power delivery networks for modern microprocessors. Miniaturization of power delivery has led to another important area - reconfigurable power delivery networks (Lee [103]). This comprises of a network of voltage/frequency converters, a switch network and a controller that can dynamically route power to different areas of the chip to realize fine-grained (zone-specific) voltage/frequency scaling. This is an emerging area across circuit, architecture, and system-level approaches to optimize power delivery to parts of a chip or the entire system based on the current workload(s).

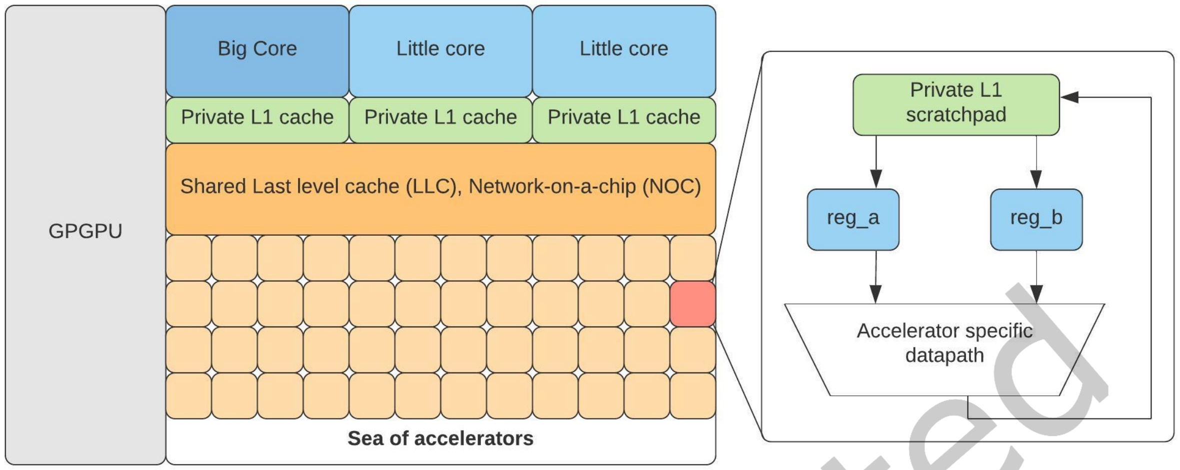

Fig. 3. Future heterogeneous architectures [144]

2.10 Programmable architectures

Field Programmable Gate Arrays (FPGAs) were once applicable to very specific domains and industries. This has changed in the last few years with FPGAs now being a critical component of data center and cloud systems, as well as edge computing systems (Ovtcharov et al. [129]). FPGAs are highly programmable in nature as they contain an array of programmable logic blocks, and a hierarchy of "reconfigurable interconnects". The blocks can be "wired together", like many logic gates that can be inter-wired in different configurations, thus making them ideal candidates for *reconfigurable computing systems* that can run highly diverse workloads. However, energy efficiency of such systems is still in its infancy with no easy or standard ways of hardware/software power management across traditional compute and FPGA subsystems.

2.11 Energy Proportional Computing

In 2007, the concept of *energy proportional computing* was first proposed by Google engineers Luiz AndrÃľ Barroso and Urs HÃűlzle [17]. Energy proportionality is a measure of the relationship between power consumed in a computer system, and the rate at which useful work is done (its utilization, which is one measure of performance). If the overall power consumption is proportional to the computer's utilization, then the machine is said to be energy proportional. Up until recently, computers were far from being energy proportional for three primary reasons. The first is high static power, which means that the computer consumes significant energy even when it is idle. High static power is common in servers owing to their design, architecture, and manufacturing optimizations that favor high performance instead of low power. The second reason is that the various hardware operating states for power management can be difficult to use effectively due to complex latency/energy tradeoffs. This is because deeper low power states tend to have larger transition latency and energy costs than lighter low power states. For workloads that have frequent and intermittent bursts of activity, such as cloud microservices, systems do not use deep lower power states due to significant latency penalties, which may be unacceptable for the application(s). The third reason is that beyond the CPU(s), very few system components are designed with fine grained energy efficiency in mind. The fact that the nature of the data center has changed significantly from being compute bound to being more heterogeneous has now exacerbated the problem and energy proportionality of all components will be an important area of research.

2.11.1 Data center energy efficiency. One of the biggest challenges for large server farms and data center operators is the increasing cost of power and cooling. Over the past decade, the cost of power and cooling has increased tremendously, and these costs are expected to continue to rise. As reported in 2015, (Hamilton [81]), power distribution and cooling accounts for 18% of costs in data centers. The Green Grid consortium [69] defines Power Usage Effectiveness (PUE) a metric used to capture the efficiency of a data centerâĂŹs cooling and power delivery mechanisms. PUE is defined as the ratio of total amount of energy used by a computer data center facility to the energy delivered to computing equipment.

$$\text{PUE} = \frac{\text{Total\_Power\_Consumption}}{\text{IT\_Power\_Consumption}}$$

An ideal PUE is 1.0. Any energy consumption that goes towards a non computing device in a data center falls in the category of facility energy consumption, or IT power consumption. PUE has become the most commonly used metric for reporting energy efficiency of data centers, with many public cloud vendors like Google, Microsoft and Facebook reporting PUE regularly. However, one problem with this metric is that PUE does not account for the climate in the region the data center is built in. In particular, it does not account for different normal temperatures outside the data center. So, a data center running in a tropical region may have a higher PUE than one running in Alaska, but it may actually be running more efficiently.

PUE was published in 2016 as a global standard under ISO/IEC [128] as well as a European standard [127].

Recent research has looked at the impact of the recent explosion in the range of cloud workloads in data centers. In Gan and Delimitrou [63], the authors investigate the architectural implications of microservices in the cloud, specifically system bottlenecks and implications to server design. Gan et al. [64] present an open source benchmark for microservices, DeathStarBench, that can measure hardware-software implications for data center systems. In Ayers et al. [15], the authors present asmDB, which looks at the source of front end stalls (cache misses, instruction cache misses, etc.) in large warehouse-scale computers, and present some optimizations that can help mitigate such system bottlenecks. Mirhosseini et al. [116] explore *killer microseconds* - microsecond-scale "holes" in CPU schedules caused by I/O stalls or idle periods between requests in high throughput microservices that are typical in data centers. They then propose enhancements to server architectures to help mitigate such effects. At a system level, Ilager et al. in [86] explore using ML techniques for thermal prediction for energy efficient management of cloud computing systems.

2.12 Advanced Packaging, 3D stacking, chiplets

While Moore's Law has slowed down, we have found ways to continue the scaling towards lower process nodes (sub-10nm) using technologies like 3D stacking and Through-Silicon-via (TSV - a via being a vertical chip-to-chip connection) (Lim [107]), Near and sub Threshold Voltage (NTV) designs (Borkar et al. [94]), newer memory integration technologies, and more recently chiplets. Intel's Foveros (chiplets) [31] is a new silicon stacking technique that allows different chips to be connected by TSVs so that the the cores, onboard caches/memory and peripherals can be manufactured as separate dies and can be connected together. By picking the best transistor for each function âĂŞ CPU, IO, FPGA, RF, GPU and accelerator âĂŞ the system can be optimized for power, performance and thermals. Additionally, by stacking chiplets vertically Intel expects that it will be able to get around a major bottleneck in high-performance system-in-package design âĂŞ memory proximity. While these technologies provide advanced packaging capabilities, cooling methods for such chips is currently a crucial area of development in the industry and will be an ongoing challenge.

2.13 Thermodynamic computing, Landauer Limit and Quantum Computing

Richard Feynman, in his classic work [58] laid down the foundations of thermodynamic and quantum computing, which are now on the horizon. As detailed in the recent report on thermodynamic computing (Conte et al. [26]), in today's "classical" computing systems that are based on transistors, quantum mechanical effects of sub-7/sub-5 nm are addressed by âĂIJaveraging themâĂİ by appropriate tools and technologies. In such systems, components such as transistors are engineered such that their small-scale dynamics are isolated from one another. In the quantum computing domain, quantum effects are avoided by âĂIJfreezing themâĂİ at very low temperatures. In the thermodynamic domain, fluctuations in space and time are comparable to the scale of the computing system and/or the devices that comprise the computing system. This is the domain of non-equilibrium physics and cellular operations, which is highly energy efficient. For example, proteins fold naturally into a low-energy state in response to their environment.

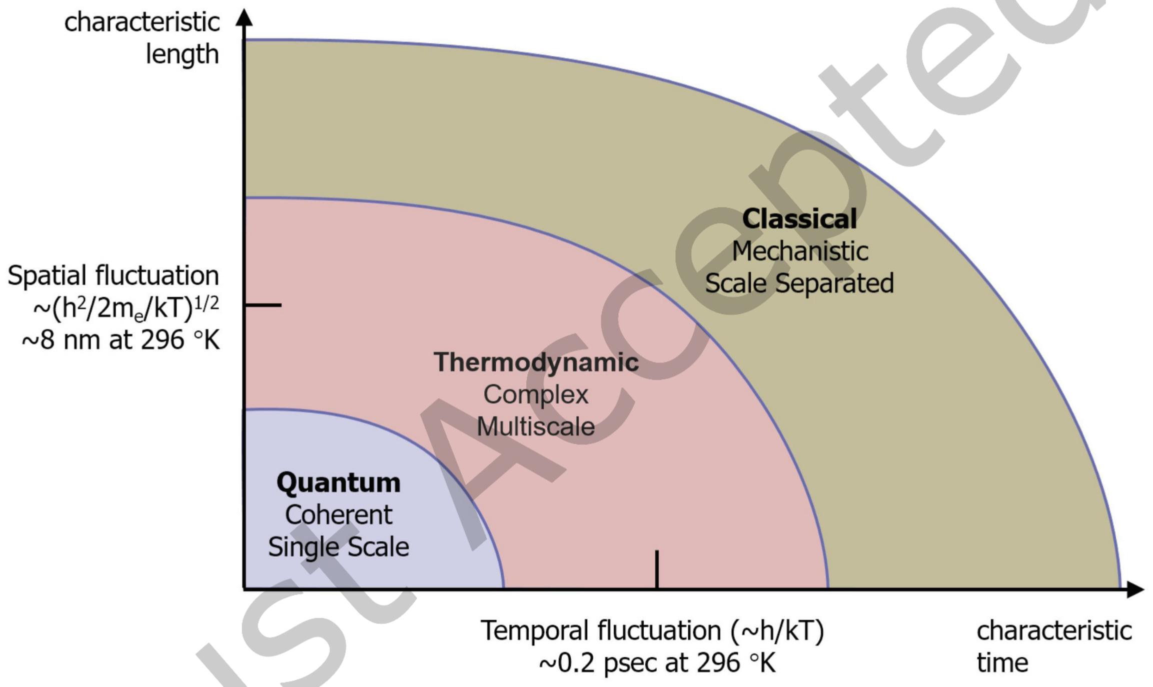

Fig. 4. Comparing scales of classical, quantum and thermodynamic computing [26]

Rolf Landauer, motivated by John von Neumann's considerations of entropy involved in computation, reasoned that when a bit of information is irreversibly transformed (erased, for example), or when two bits combine logically to yield a single bit (logic operations, for example), some information is lost, thereby resulting in a change in entropy of the system. *Landauer's principle* [99] asserts that there is a minimum possible amount of energy required to erase one bit of information, known as the Landauer limit. Some recent work [153] has demonstrated nanomagnetic logic structures that operate near the Landauer Limit, thereby raising the possibility of developing highly energy efficient computing systems in the future.

Quantum computing is another important architectural trend with different kinds of quantum hardware being built along with varying systems architectures, languages, runtime and workloads, as reported in Bertels et al.

[18] and Gyongyosi et al. [72]. Getting such systems to work is the immediate focus across research and industry, and energy efficiency will be an important topic for the future. These topics are however, beyond the scope of this survey.

3 BASICS OF POWER/THERMAL DISSIPATION, ENERGY EFFICIENCY

This section, produced as online Appendix A, provides a brief background of the basics of power /thermal dissipation, energy metrics, and energy efficiency techniques. The different components of power dissipation in CMOS devices are exhaustively covered in Kaxiras and Martonosi [95].

4 MICROARCHITECTURAL TECHNIQUES

The fundamental techniques for energy efficiency involve fine-grained clock/power gating, dynamic frequency scaling (DFS) and dynamic voltage frequency scaling (DVFS). The basics of these techniques and thermal dissipation/management are described in exhaustive detail in Kaxiras and Martonosi [95]. Given the vast amount of work done in the area of energy efficiency techniques implemented in microarchitecture, we do not attempt to survey them all here. Instead, we focus only on those techniques that are visible and controllable by higher layers of the firmware/OS/software stack. This section focuses on such microarchitectural techniques for energy efficiency across CPU, memory and accelerators like GPUs and deep learning chips, and is produced as online Appendix C.

5 SPECIFICATION

Energy efficiency techniques at hardware / RTL level (clock gating, multi-voltage design, power gating and DVFS) are specified using industry standards like IEEE 1801 Unified Power Format (UPF). At the hardware-firmware-OS level, a set of specifications are used to describe underlying hardware, power, performance and thermals. Further up the stack, the OS and applications use these abstractions to implement various energy efficiency techniques, such as the Linux Idle and Runtime PM framework, DVFS governors, thermal management algorithms and Windows Connected Standby. The specifications and abstractions used at, and across, each level are described in this section, and produced as online Appendix D.

6 MODELING AND SIMULATION

The main goal of simulation is to model new research ideas for parts of a system (processor, memory, accelerator and others) or a complete system (SOC or server) and estimate metrics such as performance and energy. In this section, we focus primarily on power, energy and thermal modeling/estimation tools for multicore processors, domain-specific accelerators, and SOC/full chip systems. This section is produced as online Appendix E.

7 VERIFICATION

Verifying energy efficiency features of complex SOCs is a big challenge from hardware as well as a system level perspective, since power management flows span the entire platform. Ideally, each system component (hardware, firmware, software) needs to be verified for its power management capability both individually as well as how they work in relation to other components, and with real workloads. In addition, system-level power flows (low power idle/standby states) also need to be verified before silicon tape-in is achieved. This section, produced as online Appendix F, discusses verification at the gate, RTL/architectural and system level.

8 SYSTEM LEVEL TECHNIQUES FOR ENERGY EFFICIENCY

In this section we look at how underlying architectural and microarchitectural techniques are used at higher levels of the software hierarchy (firmware, operating system and applications) and how energy efficiency is implemented at the entire system. Depending on the constraints of the system (IoT, wearable, smartphone, or server) several of these techniques may be used to fine tune the system for specific workloads. Since it is hard to discuss system level techniques without being specific about the underlying system architecture, we elaborate on ARM and x86 systems. We will first cover the system level techniques implemented in these systems and then discuss how software uses these features to optimize for energy efficiency.

8.1 ARM System Architecture and Energy Efficiency Features

8.1.1 Clock Gating, Dormant Mode and Power Collapse. ARM processors implement clock gating for the CPU using the Wait-For-Idle (WFI) instruction. Most ARM cores also provide the capability to clock gate the L2 cache, debug logic, and other components using co-processor instructions. Dormant Mode allows for cache controller and CPU to be powered down with the cache memories remaining powered on. The cached RAMs may be held in a low-power retention state where they keep their contents but are not otherwise functional. This mode helps achieve power savings by turning off the cache masters at the same time preventing any performance hit due to invalidation/flush of the caches. Power gating a core results in the context having to be reset at resume. ARM based platforms may have multiple clusters of cores, with each cluster having a shared L2. Power collapse of all CPU cores in a cluster results in a cluster power down which includes disabling cache snoops and power gating the L2 cache. A System Control Processor (SCP) provides several PM functions and services âĂŞ (a) Managing clocks, voltage regulators to support DVFS (b) Power state management for SoC domains and (c) Maintain/enforce consistency between device states within the system.

8.1.2 DVS/DVFS/AVFS. All modern ARM SoCs usually support software controlled DVFS. Apart from a maximum sustained frequency, several ARM SoC vendors add a boost mode where the CPU can be overclocked if required. For Symmetric Multi Processors (SMP) and Hetergeneous Multi Processor (HMP) systems with multiple (hetero) cores, the most common configuration is having a single voltage rail for all the cores in a cluster. Per-core voltage rail implementations are rare due to design complexity. Per-core clock lines are available on some SoCs allowing for independent control of core frequency with glue logic handling the voltage synchronization for the common voltage rail. ARM11 introduced a new *Intelligent Energy Manager (IEM)* that could dynamically predict the lowest voltage. This is *Adaptive Voltage Frequency Scaling (AVFS)* - a closed-loop system which continuously monitors system parameters through sensors. The IEM lowers the voltages below the values of the stock voltage tables when silicon characteristics reported by sensors permit it. Some ARM-based SOCs use power-efficient and high performance hetero cores in a single SoC as separate clusters, called *BIG.LITTLE* systems. The standard pattern of usage on mobile devices is that of periods of high processing and longer periods of light load. The core idea is that with appropriate task placement and packing on the HMP clusters, performance and power criteria both can be met. The recent DynamIQ is similar - it bundles both high performance big CPUs and high efficiency LITTLE CPUs into a single cluster with a shared coherent memory. All task migrations between big and LITTLE CPUs take place within a single CPU cluster through a shared memory, with the help of an upgraded snoop management system, resulting in improved energy efficiency. The transfer of shared data between BIG and LITTLE cores takes place within the cluster reducing the amount of traffic being generated and in turn the amount of power spent.

8.1.3 Device PM and Power Domains. ARM SoCs are typically partitioned into multiple voltage domains allowing for independent power control of devices and independent DVFS. Additionally voltage regulators are organized hierarchically so that the Linux Regulator framework can be used by software to indicate when components are idle and do not need clock/power. This allows for system level power collapse. Power collapse of an IP or group of IPs is made possible by this partitioning and hierarchical clock and voltage framework. The focus is always to reduce the number of always-on power domains on a platform and allow as many domains as possible to be turned off. Software orchestrates these dynamic power plane management based on the usage scenario - device drivers manage the clock and power to respective hardware and OS software manages system level power domains. The common system low power states on ARM SoCs are:

- **S2R**: Here the entire system is off except for components like wake-up logic and internal SRAMs
- **Low Power Audio**: Most SoCs support a special low power audio state to minimize power consumption for use cases like âĂIJscreen off user listening to musicâĂİ. The internal audio SRAM, DRAM, DMA and I2S Controller are only active (audio power domain is ON). CPU/dedicated DSP wakes up periodically to process the audio data and the display remains off.
- **Low Power Display**: Another common use case is when the modem, display and audio are only active during a voice call. This is handled by a low power display state.

Several other similar low power states are supported based on the low power usage scenario (low power sensing, low power voice call). Suspend-to-Disk, which is a common feature in larger laptops and desktops, is generally not supported on ARM based tablets/mobiles due to large resume latencies.

8.2 Intel x86 Power Management

Intel x86 SOCs provide fine-grained knobs for device and system level power management. OS Power managers like ACPI traditionally directs the platform to various power states (S3/S4, for example) depending on different power policy set by the user. Intel SOCs have components in OS and firmware that guide the power states for the CPU, devices, other subsystems and the system as a whole. A combination of hardware (dedicated power management units) and software (OS, kernel drivers, software) orchestrate the transition of the system into low power states. The overall power management architecture is built around the idea of aggressively turning off subsystems without affecting the end user functionality and usability of the system. This is enabled by several platform hardware and software changes:

- *On die clock/power gating* - applicable to all subsystems, controllers, fabrics and peripherals.
- *CPU C-states* - C-states are the CPU cores' low power states and a state Cx, means one or more subsystems of the CPU is at idle, powered down. For example, C1 is a AUTOHALT state, C3 means that the processor caches are flushed and the processor clocks are shutoff. In C6, the CPU core voltage can be shut off. More details are in the Intel x86 developer manual [32]. Higher levels of software (operating system) can initiate entry into some of these states and monitor residencies in different states.
- *CPU P-states* - CPU P-states are performance states, and each Px state represents a specific operating frequency and a corresponding voltage it needs to run at. More details are in the Intel x86 developer manual [32]. Selecting an appropriate P-state can be done through architectural registers, and there are several software and hardware-software techniques to do this, which we will describe shortly.
- *Subsystem active idle states* âĂŞ applicable to all OS/driver controlled components. These states, called **D0ix**, are managed either in hardware or using the Linux Runtime PM framework (in the kernel) and the device drivers (in the OS).
- *Platform idle states* - extending idleness to the entire platform when all devices are idle. These are termed **S0ix** states. In these states, many platform components are transitioned to an appropriate lower power state (CPU in low power sleep state, memory in self refresh, and most components are clock or power gated).
- Microcontrollers for power management of north (CPU, GPU) and south complex IPs (peripherals) respectively. The microcontrollers coordinate device and system transitions, voltage rail management, and system wake processing.
- *Integrated Voltage Regulators (IVR)*: On-die and on-chip voltage regulators provide fine-grained power delivery to different parts of the chip and this is managed by hardware and/or firmware/software.

Many Intel SoCs have CPU cores organized in a hierarchical structure, which has three levels: core, module, and package. A package contains two modules, each of which groups two cores together. This topology allows two levels of task consolidation: in-package and in-module. With in-package consolidation, the workload runs on either the first module or both modules, i.e., all of the four cores. Intel CPUs support DVFS or performance states (or P-states) for OS controlled management of processor performance. The P-states are exposed via ACPI tables to the OS. OS Software requests a P-State based on performance needs of the application (in Linux/Android, this is via the cpufreq-based governors). Atom cores also support Turbo frequencies akin to boost on ARM SoCs. Turbo allows processor cores to run faster than the âĂIJguaranteedâĂİ operating frequency if the processor is operating below rated power, temperature, and current specification limits of the system. Turbo takes advantage of the fact that the rated maximum operating point of a processor is based on fairly conservative conditions which occur infrequently.

8.2.1 System low power states. Intel SoCs support the following transient low power system states:

(1) S0ix: Shallow idle state for the entire SOC
(2) S0ix-Display: display can be kept in a shallow low power state, with display controller periodically waking up to feed the contents of the display panel and the rest of the SOC fully powered off.
(3) S0ix-Audio: SOC in low power state except audio block.
(4) S0ix-Sensing: SOC in low power state except sensor hub to support several low power sensing modes such as pedometer
(5) S0i3: Entire SOC is in low power state, except for wake logic/sequencing and a small amount of memory to store code for restoring the SOC back to operating state.

All these states are transparent to applications and are entered/exited by close orchestration between operating system, firmware, microcontrollers and hardware and have different entry/exit latencies. In addition to these, systems generally support **Suspend-to-RAM**, where the entire system is off except for minor exceptions such as wake-up logic, internal SRAMs etc. and **Suspend-to-Disk**, that has larger entry/exit latencies but also deeper power savings.

8.3 OS and Software Techniques

Linux [108] has developed several energy efficiency features in the last two decades and the following have been among the most important ones:

(1) **Timers and Tickless Scheduling**: The scheduler allocates CPU time to individual processes via interrupts. Programmable timer interrupts keep track of, and handle future events. In traditional systems we had a periodic tick i.e. the scheduler runs at a constant frequency. This resulted in periodic wake-ups and poor energy efficiency. Linux evolved to use three primary mechanisms, as described in Siddha et al. [148] and [46] - (a) *Dynamic tick* - program the next timer interrupt to happen only when work needs to be done, (b) *Deferrable timers* - bundle unimportant timer events with the next interrupt (c) *Timer migration* - move timer events away from idle CPUs. Some CPUs also support *power-aware interrupt redirection (PAIR)*, that ensures that interrupts are directed to already-awake CPU cores, rather than wake up a sleeping core.
(2) **CPUFreq**: This is a standard Linux framework used for CPU Dynamic Voltage and Frequency Scaling (DVFS). Processors have a range of frequencies and corresponding voltages over which they may operate. The CPUFreq framework allows for control of these voltage-frequency pairs according to the load through components called *governors*. There are several different governors based on how the algorithm can be controlled and implemented. The performance governor is used for optimizing CPU performance whereas the power-save governor aims to conserve energy. The user-mode governor allows a user space application to control the DVFS states. The on-demand governor was one of the most popular governors, described in

Pallipadi et al. [132]. More recently, the interactive governor was developed for mobile devices that require optimized burst performance for on-screen usages. The Intel P-state driver is slightly different - it can operate in two different modes, active or passive. In the active mode, it uses its own internal performance scaling governor algorithm or allows the hardware to do performance scaling by itself, while in the passive mode it responds to requests made by a generic CPUFreq governor implementing a certain performance scaling algorithm. All of these are described in detail in the Linux kernel documentation [43] and the Intel P-state driver is described in more detail in [96].

(3) **CPU Idle**: This is a Linux kernel subsystem that manages the CPU when it is idle and the core idea is to *do nothing, efficiently* (Pallipadi et al. [131]). Usually, several idle states, known as C-states, are supported by the processor. The convention for C-state naming is that 0 is active state and a higher number indicates a deeper idle state e.g. C1-Clock Gating. Deeper idle states mean larger power savings as well as longer entry/exit latencies. The inputs required by the framework for C-state entry are âĂŞ CPU idleness, next expected event, latency constraints, break-even time and exit latency. Based on the inputs, a specific C-state is entered via architecture specific instructions such as MWAIT in x86.

(4) **PM Quality of Service**: PM QOS is a latency and performance control framework in Linux [47]. It provides a synchronization mechanism across power managed resources with a minimum performance need as expressed by a device. The kernel infrastructure facilitates the communication of latency and throughput needs among devices, system, and users. QoS can be used to guarantee a minimum CPU frequency level to meet video playback performance or to limit the max device frequency to reduce skin temperature, and similar constraints.

(5) **Voltage Regulator framework** is a standard kernel interface to control voltage/current regulators [45]. It is mostly used to enable/disable a regulator output or control the output voltage and or current. The intention is to allow systems to dynamically control regulator power output in order to save power and prolong battery life. This applies to both voltage regulators (where voltage output is controllable) and current sinks (where current limit is controllable). Many drivers use this framework to enable/disable voltage rails or control the output of low drop out oscillators (LDOs) or buck boost regulators.

(6) **Runtime PM framework** is a widely used framework in the Linux kernel [48] to reduce the individual device power consumption when the device is idle through clock gating, gating the interface clock, power gating or turning off the voltage rail. In each of the cases we need to ensure that before we move the device to a low power state, any dependent devices are also considered. The framework allows for understanding and defining this tree for hierarchical control.

(7) **Devfreq** framework is used for handling DVFS of non-CPU devices such as GPU, memory and accelerator subsystems [44]. Devfreq is similar to cpufreq but cpufreq does not allow multiple device registration and is not suitable for heterogeneous devices with different governors. It exposes controls for adjusting frequency through sysfs files which are similar to the cpufreq subsystem.

(8) **System sleep states** provide significant power savings by putting much of the hardware into low power modes. The sleep states supported by the Linux kernel are power-on standby, suspend-to-RAM (S2R), suspend to idle (S2I) and suspend to disk (hibernate) [49]. Suspend to idle is purely software driven and involves keeping the CPUs in their deepest idle state as much as possible. Power-on standby involves placing devices in low power states and powering off all non-boot CPUs. Suspend to RAM goes further by powering off all CPUs and putting the memory into self-refresh. Lastly, suspend to disk gets the greatest power savings through powering off as much of the system as possible, including the memory. The contents of memory are written to disk at suspend, and on resume this is read back into memory.

(9) **Power Capping Framework**: The Linux power capping framework provides a consistent interface between the kernel and the user space that allows power capping drivers to expose the settings to user space in a uniform way [42]. Power zones represent different parts of the system, which can be controlled and

monitored using the power capping method determined by the control type the given zone belongs to. They each contain attributes for monitoring power, as well as controls represented in the form of power constraints. With the power capping framework, it is possible to apply power capping to a set of devices together. Intel RAPL [Section 9.1.5] is one form of a power capping framework.

(10) **Multi-cluster PM and Energy Aware Scheduler**: The Multi Cluster PM (MCPM) layer supports power modes for multiple clusters. It implements powering up/down transitions of clusters including the necessary synchronization. The Linux scheduler traditionally placed importance on CPU performance and did not consider the different power curves if disparate cores exists in one system. The Energy Aware Scheduler (EAS) links several otherwise independent frameworks such as CPUFreq, CPUIdle, thermal and scheduler to be more energy efficient even for disparate cores. A scheduler directed CPUFreq governor called schedutil has been introduced which takes optimal decisions regarding task placements, CPU idling, frequency level to run, among other parameters. Based on a SoC specific energy model, EAS realizes a power efficient system with minimal performance impact. This is commonly implemented today on several ARM based systems [112].

8.4 System and OS Techniques for Energy Efficiency in GPUs

The techniques for improving energy efficiency of GPUs overlaps with those used for CPUs and a detailed survey is presented in Mittal et al. [117]. Some key techniques are highlighted here:

(1) **Workload-based dynamic resource allocation**: This is based on the observation that the power consumption of GPUs is primarily dependent on the ratio of global memory transactions to computation instructions and the rate of issuing instructions. The two metrics decide whether an application is memory intensive or computation intensive respectively. Based on the metrics, the frequency of GPU cores and memory is adjusted to save energy. Some systems use an integrated power and performance prediction system to save energy in GPUs. For a given GPU kernel, their method predicts both performance and power and then uses these predictions to choose the optimal number of cores that can lead to the highest performance per watt value. Based on this, only the desired number of cores can be activated, while the remaining cores can be turned off using power gating.

(2) **CPU-GPU Work division**: Research has shown that different ratios of work division between CPUs and GPUs may lead to different performance and energy efficiency levels. Based on this observation, several techniques have been implemented that dynamically choose between CPU and GPU as a platform of execution of a kernel based on the expected energy efficiency on those platforms.

(3) **CPU-GPU Power Sharing**: In several recent CPU-GPU systems, dynamic power sharing is implemented at the firmware, microkernel and/or OS level to dynamically balance the power being consumed by the CPUs and GPUs. For example, in [29], the power sharing framework is used to balance the power between high performing processors and graphics subsystem. It helps to manage temperature, power delivery and performance state in real time and allows system designers to adjust the ratio of power sharing between the processor and graphics based on workloads and usages.

9 RECENT ADVANCES IN SOC AND SYSTEM LEVEL ENERGY EFFICIENCY

The last few years have seen rapid innovations in SOC design/microarchitecture and system level power/performance optimizations across x86 and non-x86 architectures, including the rise of custom-designed ARM chips by different companies such as Apple, Amazon, Google, Ampere, etc. In this section, we review some of the key technologies across these architectures and systems.

Table 3. Summary of recent energy efficiency techniques in Intel and AMD x86 processors

Technique	Processor/SOC family
Per Core P-States, Uncore Frequency Scaling, Integrated Voltage Regulator (IVR), Running Average Power Limiting (RAPL)	Intel Haswell (22nm) [75]
Intel Speedshift (Hardware P-states), Energy-aware Race to Halt (EARtH), Energy Performance Bias (EPB) / Energy Performance Preference (EPP), Hardware/SOC Duty Cycle, Memory DVFS, enhanced RAPL, IccMax/Peak current management	Intel Skylake (14nm) [50]
Dynamic Tuning (ML-based Turbo)	Intel Ice Lake (10nm) [6]
Autonomous Fabric and Memory DVFS, independent clock and power domains for Graphics, Memory, PCIe, USB, Thunderbolt	Intel Tiger Lake (10nm) [10]
Heterogeneous x86 cores	Intel Lakefield [31]
Locally Efficient Application Power Management (LEAPM), Globally Efficient APM (GEAPM), Core Bound Boost (CBB), Memory-Bound Boost (MBB)	AMD (28nm) [19]

9.1 Energy Efficiency in Intel Processors/SOCs

Starting with the Broadwell, Intel architectures implemented several system level techniques for energy efficiency while pushing the performance envelopes for newer workloads and all-day battery life for active scenarios. Similarly, AMD processors also evolved several techniques across client and server processors. Some of the most important features and techniques are described here and are summarized in Table 3.

9.1.1 Intel Speed Shift Technology (Hardware P-States). Skylake (Doweck et al. [50]) is a SOC consisting of 2-4 CPU cores, Graphics, media, a ring interconnect, an integrated system system, and a Power Control Unit (PCU) that houses the power management firmware logic and provides interfaces to higher power management hierarchies (BIOS, OS, device drivers, etc.). Speed Shift is a faster response vehicle to frequency requests and race to sleep by migrating the control from the operating system back down to the hardware. Current implementation of OS-guided P-states can take up to 30 milliseconds to adjust, whereas if they are managed by the processor, it can be reduced to about 1 millisecond. At any time the OS can demand control of the states back from the hardware if specific performance is needed. The key concept behind autonomous processor level control is to find the power state that uses the least total system system power, and stay in that state as often as possible.

9.1.2 Workload Aware Power Balancer. For active workloads, Intel Skylake looks to balance the power across CPU cores, Graphics, and other subsystems (memory and uncore). A feedback-based control system monitors the different units (CPU, Graphics, memory, uncore, imaging/camera subsystem) and uses that information to understand the nature of the workload. That information is used to split the available system power between CPU, Graphics and other subsystems. By default, such power budget allocation could be fixed, corresponding to worst-case performance demands/workloads, even if the domains are under utilized. This unfair allocation is sub-optimal and can hamper overall system performance and throughput. In SysScale [77], the authors introduce an algorithm to predict the performance demands (bandwidth, latency) of the SOC domains and implements a new DVFS algorithm to distribute SOC power based on predicted performance demands. Furthermore, in addition to a global DVFS mechanism, SysScale optimizes the DVFS of each domain from an energy efficiency perspective.

9.1.3 SOC/ Hardware Duty Cycling. One of the fundamental concepts for saving power is *race to idle or sleep* - get the job done as soon as possible, and put the CPU into an idle state. As process technology starts encountering fundamental physics limits, and due to the fact that transistors cannot operate reliably below a certain threshold voltage, idling the processor at lower frequencies starts providing diminishing returns once we get closer to the threshold voltage. Intel Broadwell and Haswell processors introduced the idea of Duty Cycling Control (DCC) for the integrated graphics unit, which meant that the GPU would be cycling between on and off states. Skylake introduced this concept for the CPU cores as well, and rapidly transitions the CPU cores between on and off states. This technique has shown to save large amounts of power for a range of workloads.

9.1.4 Energy Aware Race to Halt. Intel Skylake also introduced a new algorithm called Energy Aware Race to Halt (EARtH) as described in Deweck et al. [52]. The motivation behind this is based on the observation that controlling CPU power has limited impact on the overall energy efficiency of the computing platform due to energy consumption of other platform components. When the CPU power dominates total power, the minimum energy is achieved when the CPU operates at the lowest frequency mode (LFM). When the rest of the platform consumes significantly higher power than the CPU, the most energy efficient policy is Race To Halt (RtH). In many real systems, however, power is balanced between CPU and the rest of the platform for different workloads. In such systems the minimum energy point may happen at some intermediate frequency. The authors in this paper demonstrate this observation in real systems and with real production workloads and present an Energy Aware Race to Halt (EARtH) algorithm that identifies that minimum energy point at run time. Starting with Skylake, this algorithm (with some enhancements) is now available in most Intel Core processors including the latest Ice Lake and Tiger Lake SOCs.

9.1.5 Running Average Power Limiter (RAPL). Intel's RAPL provides a set of counters providing energy and power consumption information using a software power model that estimates energy usage by using hardware performance counters and I/O models [89]. The key idea behind RAPL is that of Thermal Design Power (TDP). The TDP of a system represents the maximum amount of power the cooling system in a computer is required to dissipate. For example, for a processor with TDP of 35W, Intel guarantees the OEM that if it implements a chassis and cooling system capable of dissipating that much heat, the chip will operate as intended. This is the power budget under which the system needs to operate. But this is not the same as the maximum power the processor can consume. It is possible for the processor to consume more than the TDP power for a short period of time without it being âĂIJthermally significantâĂİ. Using basic physics, heat will take some time to propagate, so a short burst may not necessarily violate TDP. RAPL provides a set of counters providing energy and power consumption information. RAPL is not an analog power meter, but rather uses a software power model. This software power model estimates energy usage by using hardware performance counters and I/O models.

RAPL provides a way to set power limits on processor packages and DRAM. This will allow a monitoring and control program to dynamically limit max average power, to match its expected power and cooling budget. In addition, power limits in a rack enable power budgeting across the rack distribution. By dynamically monitoring the feedback of power consumption, power limits can be reassigned based on use and workloads. Because multiple bursts of heavy workloads will eventually cause the ambient temperature to rise, reducing the rate of heat transfer, one uniform power limit canâĂŹt be enforced. RAPL provides a way to set short term and longer term averaging windows for power limits. These window sizes and power limits can be adjusted dynamically.

9.1.6 Intel P-state driver. Starting with Intel's Sandybridge, this driver provides an interface to control the P-State selection for the processors. The underlying driver in the kernel is essentially a Proportional Integral Derivative (PID) controller with software-tunable interfaces to control each of the P, I, and D parameters [96]. The driver decides what P-State to use based on the requested policy from the OS's cpufreq core. If the processor is capable of selecting its next P-State internally, then the driver will offload this responsibility to the processor (Hardware P-States). If not, the driver implements algorithms to select the next P-State. The P-state driver is primarily supported only for Linux based platforms.

9.1.7 Dynamic Current Management, peak current and thermal protection. Intel processors have two modes of current and thermal protection: throttling,and automatic shutdown. As described in [88] and [141], when a core exceeds the set throttle temperature, it will start to reduce power to bring the temperature back below that point. The throttle temperature can vary by processor and BIOS settings, and the throttling actions can be different (turning down display, turning off charging for example). Peak current violations are handled similarly. If the conditions are such that throttling is unable to keep the temperature down, such as a thermal solution failure or incorrect assembly, the processor will automatically shut down to prevent permanent damage. The peak current and peak thermal limits respectively are controlled by Processor Core IccMax and Thermal Limit PL1/PL2/PL3 settings in the BIOS.

9.1.8 Connected Standby. Connected Standby is a feature used in laptops, tablets, and smartphones in order to reduce energy consumption when the device is fully idle, while remaining connected to communication channels. A mobile device enters the deepest-runtime-idle-power state (DRIPS), which minimizes power consumption and retains fast wake-up capability. Haj-Yahya et al. [79] look at ways to increase battery life in the connected-standby mode and implement an optimized DRIPS (ODRIPS) mechanism. ODRIPS is based on 2 key ideas: (1) offload wake event monitoring to low-power off-chip circuitry, which enables turning off most of the SOC 2) offload processor context to off-chip storage (DRAM), thus eliminating the need for on-chip high-leakage SRAMs and thereby reducing leakage power.

9.1.9 Thermal Management - Intel DPTF and ARM Intelligent Power Allocator. Smart system level thermal management has improved over the last decade as form factors and workloads have impacted platform thermals significantly. Platforms today encompass several thermal sensors - per-CPU, per-GPU, for the connectivity radios, USB subsystem, etc. An intelligent thermal manager needs to comprehend the data from thermal sensors, estimate possible platform level impact (skin temperature of a device needs to be calculated using different equations based on the individual thermal sensor readings), and then impose policies (such as throttling the CPU, dim the display, or disable charging) to ensure the system can continue to function.

The Intel Dynamic Platform and Thermal Framework (Intel DPTF) is implemented on both Linux and Windows platforms [2]. It includes the DPTF Framework Manager, Policies, and Participants. The DPTF manager is responsible for all communication into the user space code, and serves as the interface to Eco-System Independent Framework (ESIF). It manages events and notifications to/from the ESIF layer and is responsible for high level arbitration of policies to ensure system level thermal management. DPTF policies are the intelligent plug-in glue that determines what needs to be done to address specific thermal situations. DPTF participants are the entities that expose telemetry (CPU temperature, for example) and provide controls (throttling the CPU P-states).

Similarly, ARM's Intelligent Power Allocator (IPA) [11] performs proactive power and thermal management by continuously adapting response based on power consumption and thermal headroom. It implements a closed-loop Proportional Integral Derivative (PID) controller for accurate temperature control. The Dynamic Power Partitioning component optimally allocates power to CPU and GPU based on the current workload based on a SoC power model, which is composed of voltage/frequency operating point for each key IP block (e.g. CPU, GPU). IPA thus maps between runtime power consumption (measured through on-chip monitoring counters) and theoretical operating points of each component.

9.2 Energy Efficiency in AMD Processors/SOCs

AMD processors and SOCs support several energy efficiency features including clock and power gating, DFS, DVS, DVFS, link level power management, etc. Some of the recent advances are noted here.

AMD SOC's power management is described in detail in Bircher et al. [19]. Here the authors describe several aspects of AMD's SOC/system level power management. The power manager is implemented on an on-die microcontroller that uses power and thermal feedback from the SOC through digital power monitors. The power monitor accounts for fluctuations in dynamic power caused by the workload and also accounts for the effects of voltage, frequency and temperature using built-in models. To provide consistent repeatable performance, the models are calibrated for each version/model of the SOC. The power manager contains three performance controllers: Global Efficient Application Power Management (GEAPM), Core-Bound Boost (CBB) and Memory-Bound Boost (MBB). Another feature, called Locally Efficient APM (LEAPM) is also implemented for IP-level power management.

(1) GEAPM optimizes the balance of power between CPU and GPU within an SOC. It is global in the sense that it seeks to maximize the SOC-level performance rather than the individual (local) performance of either CPU or GPU.
(2) The CBB and MBB features improve performance of CPU-centric workloads. CBB increases CPU performance by shifting power from the memory subsystem to the CPU for core-bound workloads (with little memory dependence).
(3) The MBB feature detects memory latency-sensitive workloads and shifts power to the memory controller.
(4) LEAPM works on the premise that some power management decisions can be made using only information local to the IP and it essentially uses different ways of tracking IP-level utilization to determine when to shift power away from an IP.

All of these features shift power from other parts of the system that have less impact on performance to those with more power requirements thus providing higher performance in a constrained environment. AMD also optimizes power consumption for different workloads through different processor/SOC settings based on die temperature, expected leakage (as leakage depends on temperature), part-to-part variations in the die itself, as documented in Suggs et al. [147], Arora et al. [12] and [4].

9.3 The rise of ARM in HPC and the Cloud

The last couple of years has also seen the rise of ARM architectures in data center and cloud systems that were traditionally x86-based, which was primarily due to the unmatched performance of Intel and AMD processors.

Amazon Web Services have released custom-build high performance ARM processors for the cloud, named Graviton, Graviton 2, and more recently, Graviton 3. These are available as AWS's EC2 instances. Graviton is an ARM64 processor based on A72 microarchietcture. In [92], the authors perform detailed performance analysis of AWS's Graviton A1 against similar class of Intel Xeon processors and observe that the A1 achieves almost similar performance in web services, with significant cost savings across various video and database workloads. Graviton2 improves on this and delivers enhanced price-performance by 40% in comparison to present generation x86-fueled processors. At the time of this writing, Graviton 3 is touted to be 25 percent faster than Graviton 2, with 2x faster floating-point performances, and a 3x speedup for machine learning workloads, with a 60X energy reduction [137]. With its highly improved power/performance/cost benefits, ARM architectures and processors have also made a big headway to HPC and supercomputing systems [160].

10 ENERGY EFFICIENCY STANDARDS, BENCHMARKS AND CROSS LAYER ENERGY EFFICIENCY

Over the last couple of decades, it has become abundantly clear that the *power wall* is going to define the next generation of sustainable computing across the entire spectrum of devices and systems, from the very small (ultra-low power, IoT, sensors, etc.) to the very large (HPC, data center, exascale and beyond). This section, will discuss important industry consortiums, standards, benchmarks and regulations for energy efficient and sustainable computing, and is produced as online Appendix G.

11 THE ROAD AHEAD AND NEW TRENDS

The semiconductor industry has gone through several decades of evolution; compute performance has increased by orders of magnitude that was made possible by continued technology scaling, improved transistor performance, increased integration to realize novel architectures, extreme form factors, emerging workloads, and reducing energy consumed per logic operation to keep power and thermal dissipation within limits. We have worked around fundamental issues like ILP limits, end of Dennard scaling, and Amdahl's limit on multi-core performance. More recently, and expectedly, there has been a slowdown of Moore's Law. The following trends will continue to inexorably push computing beyond current limits:

(1) **Lower process nodes**: The industry is currently in the sub-10nm node, and a shift to 5nm and 3nm will provide a few generations of performance gains and energy efficiency, but requiring new transistor architectures like nanosheets and nanowires beyond today's FinFETs. Stacking nanosheets will provide perhaps the last step step in Moore's Law (Ye et al. [135]).
(2) **Heterogeneous architectures**: Mainstream computing will continue to see heterogeneous architectures comprising of CPUs, GPUs, domain specific accelerators and programmable hardware (FPGAs) across the spectrum with tightly integrated solutions.
(3) **Exascale and beyond**: Research and industry will continue the push to build exascale systems using new architectures (Borkar et al. [20]) and computing paradigms like mixing von Neumann and non-von Neumann models [30].
(4) **Sub-threshold voltage designs**: At the other end of the spectrum, sub-threshold and near-threshold voltage designs and techniques will enable ultra low power IoT and embedded/wearable markets that consume drastically lower power than traditional chips[119], [60]. Companies such as Ambiq Micro, PsiKick and Minima Processor, among others, have matured techniques developed in academia (Univ of Michigan, MIT and VTT Technical Research Center at Finland, respectively) to develop ultra low power chips that operate at 0.1-0.2 V range, with wide dynamic range as well, all the way up to 0.8 V [126].
(5) **Rise of non-x86 architectures and custom chips**: The last couple of years has also seen the rise of ARM architectures in enterprise systems (laptops), data center and cloud systems that were traditionally x86-based, which was primarily due to the unmatched performance of Intel and AMD processors. Recently, we have seen Apple's M1 chip [9] in the personal PC domain and chips such as Amazon Web Services' Graviton2 [8] for the data center and ARM in HPC [160]. We believe this trend of custom silicon will continue to push the boundaries of architectural innovation.
(6) **Energy efficient hardware**: We will see newer, open standards based (RISC-V, for eg.), energy-efficient architectures as computer architecture becomes more multi-disciplinary cross cutting computer science and cognitive science as our understanding of nature and the human mind evolves (neuromorphic and bio-inspired chips, for example). TinyML [154] is an important emerging area of machine learning under the 1mW power envelope. Similarly, software-defined hardware [53] is an important area of reconfigurable systems.

(7) **Energy-aware software**: Software and operating systems will need to evolve in lock-step fashion to utilize energy efficient hardware across different categories of systems and under varying energy efficiency/thermal constraints and challenges such as dark silicon and accelerator limits.
(8) **Cross Layer Energy Efficiency, Standards**: Systems will necessitate a tight interplay between energy efficient hardware and energy aware software through standardized cross layer abstractions across architecture, design, modeling and simulation, implementation, verification and optimization of complete systems.
(9) **Domain-specific stacks**: Across different computing domains (ultra low power/IoT, edge, mainstream, cloud, HPC and exascale), the industry will see highly optimized domain-specific stacks that are built using modular, standardized hardware-software interfaces and components. For example, Tesla's full self-driving solution (FSD) (Talpes et al. [149]), which is a tightly integrated, domain-specific system for autonomous driving with a TDP of under 40W.
(10) **Neuromorphic Computing and other non-von Neumann systems**: Several industrial systems are available now that implement non-von Neumann architectural and programming models such as IBMâĂŹs TrueNorth [115] and Intel's Loihi (Davies et al. [37]). Both are based on Spiking Neural Networks to demonstrate neuromorphic architectures. Poihiki Springs took this further with a rack-mounted chassis enclosing 768 Loihi chips, FPGA interface boards, and an integrated IA host CPU. Davies et al. [38] describes the latest results of how specific types of deep learning algorithms (brain-inspired networks) perform with orders of magnitude lower latency and energy. Such architectures and systems will continue to push the boundaries of non-von Neumann computing.
(11) **Quantum Computing**: Quantum computing is on the horizon now, with several experimental quantum computing architectures being built and used for specific optimization problems. Amazon Web Services provides Braket, a fully managed cloud service that allows scientists, researchers, and developers to begin experimenting with computers from multiple quantum hardware providers in a single place [143]. Similarly, Microsoft provides a quantum development kit for the Q# quantum programming language and Azure Quantum hardware based on topological quantum computing. Intel is aiming for "quantum practicality" [90] with its attempt to use silicon spin qubits that look exactly like a transistor; this would enable high volume fabrication for silicon-based quantum computing. For the foreseeable future, quantum computers will at best be accelerators that will interface with classical computers [18]. Getting such systems to work is the immediate focus across research and industry.
(12) **Thermodynamic computing**: As we push the boundaries of computing and look at how to make computers function more efficiently, researchers are probing the foundations of *thermodynamic computing* [26] based on the observation that thermodynamics drives the self-organization and evolution of natural systems and, therefore, thermodynamics might drive the self-organization and evolution of future computing systems, making them more capable, more robust, and highly energy efficient. It is not very clear what thermodynamic computing will look like at this time of writing.

12 SUMMARY AND CONCLUSIONS

Computing systems have undergone a tremendous change in the last few decades with several inflexion points. While Moore's law guided the semiconductor industry to cram more and more transistors and logic into the same volume, the limits of instruction-level parallelism (ILP) and the end of Dennard's scaling drove the industry towards multi-core chips; we have now entered the era of domain-specific architectures, pushing beyond the memory wall. However, challenges of dark silicon and other limits will continue to impose constraints. Overall energy efficiency encompasses multiple domains - hardware, SOC, firmware, device drivers, operating system runtime and software applications/algorithms and therefore must be done at the entire platform level in a holistic way and across all phases of system development.

This survey brings together different aspects of energy efficient systems, through a systematic categorization of *specification*, *modeling* and *simulation*, *energy efficiency techniques*, *verification*, *energy efficiency benchmarks, standards, consortiums and cross layer efforts* that are crucial for next generation computing systems. Future energy efficient systems will need to look at all these aspects holistically, through cross-domain, cross-layer boundaries and bring together energy efficient hardware and energy-aware software.

Trends indicate that systems will continue to evolve, pushing the boundaries of technology, architecture, design and manufacturing. For future systems, the power wall will be the boundary condition around which computing systems will evolve across the ends of the computing spectrum (ultra low power devices to large HPC/exascale systems), through a tight interplay between energy efficient hardware and energy-aware software.

REFERENCES

[1] 2011. AMD Claims 'Fastest Graphics Card in the World'. https://www.tomshardware.com/news/Radeon-HD-6990-DirectX-11-Dual-BIOS-TeraScale-3-dual-GPU,12351.html.

[2] 01.org. 2020. IntelÂő Dynamic Platform and Thermal Framework (DPTF). (2020). https://01.org/intel-dynamic-platform-and-thermal-framework-dptf-chromium-os/documentation/implementation-design-and-source-code-organization

[3] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. 2015. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 10 (2015), 1537–1557.

[4] AMD. 2019. Workload Tuning Guide for AMD EPYCâĎć 7002 Series Processor Based Servers. (2019). https://developer.amd.com/resources/epyc-resources/epyc-tuning-guides/

[5] G. M. Amdahl. 1967. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings*, Vol. 30. AFIPS Press, Reston, VA, 483–485.

[6] Anandtech. 2019. Examining Intel's Ice Lake Processors: Taking a bite of the Sunny Cove Microarchitecture. (2019). https://www.anandtech.com/show/14514/examining-intels-ice-lake-microarchitecture-and-sunny-cove

[7] Anandtech. 2019. The Intel Optane Memory SSD Review. (2019). https://www.anandtech.com/show/11210/the-intel-optane-memory-ssd-review-32gb-of-kaby-lake-caching

[8] AnandTech. 2020. Amazon's ARM-based Graviton2 Against AMD and Intel: Comparing Cloud Compute. (2020). https://www.anandtech.com/show/15578/cloud-clash-amazon-graviton2-arm-against-intel-and-amd

[9] AnandTech. 2020. Apple Announces the Apple M1 Silicon. (2020). https://www.anandtech.com/print/16226/apple-silicon-m1-a14-deep-dive

[10] Anandtech. 2020. Intel's 11th Gen Core Tiger Lake SOC Detailed: Super Fin, Willow Cove and Xe-LP. (2020). https://www.anandtech.com/show/15971/intels-11th-gen-core-tiger-lake-soc-detailed-superfin-willow-cove-and-xelp

[11] ARM. 2020. Intelligent Power Allocation. (2020). https://developer.arm.com/tools-and-software/open-source-software/linux-kernel/intelligent-power-allocation

[12] Sonu Arora, Dan Bouvier, and Chris Weaver. 2020. AMD Next Generation 7NM Ryzen™ 4000 APU "Renoir". In *IEEE Hot Chips 32 Symposium, HCS 2020, Palo Alto, CA, USA, August 16-18, 2020.* IEEE, 1–30.

[13] IEEE Standards Association. 2016. IEEE P2415 - Standard for Power Modeling to Enable System Level Analysis. (2016). https://standards.ieee.org/project/2415.html

[14] IEEE Standards Association. 2019. IEEE P2416 - Standard for Power Modeling to Enable System Level Analysis. (2019). https://standards.ieee.org/project/2416.html

[15] Grant Ayers, Nayana Prasad Nagendra, David I. August, Hyoun Kyu Cho, Svilen Kanev, Christos Kozyrakis, Trivikram Krishnamurthy, Heiner Litz, Tipp Moseley, and Parthasarathy Ranganathan. 2019. AsmDB: Understanding and Mitigating Front-End Stalls in Warehouse-Scale Computers. Association for Computing Machinery, New York, NY, USA.

[16] Luiz Andre Barroso and Urs Hoelzle. 2009. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines* (1st ed.). Morgan and Claypool Publishers.

[17] Luiz André Barroso and Urs Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (Dec. 2007), 33âĂŞ37.

[18] Koen Bertels, Aritra Sarkar, A Mouedenne, Thomas Hubregtsen, A Yadav, Anneriet Krol, and Imran Ashraf. 2019. Quantum Computer Architecture: Towards Full-Stack Quantum Accelerators. (09 2019).

[19] W. L. Bircher and S. Naffziger. 2014. AMD SOC power management: Improving performance/watt using run-time feedback. In *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference.* 1–4.

[20] Shekhar Y. Borkar. 2010. The Exascale challenge. *Proceedings of 2010 International Symposium on VLSI Design, Automation and Test* (2010), 2–3.

[21] Kevin K. Chang, A. Giray Yağlıkçı, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu. 2017. Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms. 1, 1 (2017).

[22] Vincent Chau, Xiaowen Chu, Hai Liu, and Yiu-Wing Leung. 2017. Energy Efficient Job Scheduling with DVFS for CPU-GPU Heterogeneous Systems. Association for Computing Machinery, New York, NY, USA.

[23] Samit Chaudhuri and Asmus Hetzel. 2017. SAT-based compilation to a non-von neumann processor. In *Proceedings of the 36th International Conference on Computer-Aided Design.* IEEE Press, 675–682.

[24] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, Vol. 44. IEEE Press, 367–379.

[25] Compute Express Link Consortium. 2020. Compute Express Link. (2020). https://www.computeexpresslink.org/

[26] Tom Conte, Erik DeBenedictis, Natesh Ganesh, Todd Hylton, Susanne Still, John William Strachan, Stan Williams, Alexander Alemi, Lee Altenberg, Gavin Crooks, James Crutchfield, Lidia Rio, Josh Deutsch, Michael DeWeese, Khari Douglas, Massimiliano Esposito, Michael

Frank, Robert Fry, Peter Harsha, and Yan Yufik. 2019. Thermodynamic Computing: A Report Based on a Computing Community Consortium (CCC) Workshop. (11 2019).
[27] Intel Corporation. 2017. Intel® Movidius^{TM} Myriad^{TM} VPU 2: A Class-Defining Processor. https://www.movidius.com/myriad2.
[28] Intel Corporation. 2017. Intel® Movidius^{TM} SHAVE v2.0 - Microarchitectures - Movidius. https://en.wikichip.org/wiki/movidius/microarchitectures/shave_v2.0.
[29] Intel Corporation. 2018. Intel and AMD working on Power Sharing Across CPU and GPU for Optimal Performance. https://www.spokenbyyou.com/intel-amd-working-power-sharing-across-cpu-gpu-optimal-performance/.
[30] Intel Corporation. 2018. Intel's Exascale Dataflow Engine drops x86 and von Neumann. https://www.nextplatform.com/2018/08/30/intels-exascale-dataflow-engine-drops-x86-and-von-neuman/.
[31] Intel Corporation. 2019. Lakefield: Hybrid CPU with Foveros Technology. https://newsroom.intel.com/press-kits/lakefield/.
[32] Intel Corporation. 2020. Intel 64 and IA-32 Architectures Software Developer's Manual. (2020). https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-system-programming-manual-325384.pdf
[33] Intel Corporation. 2022. New Mobileye EyeQ Ultra will Enable Consumer AVs. https://www.intel.com/content/www/us/en/newsroom/news/mobileye-ces-2022-tech-news.html.
[34] Microsoft Corporation. 2019. What is Modern Standby? https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/modern-standby.
[35] David E Culler. 1986. Dataflow architectures. *Annual review of computer science* 1, 1 (1986), 225–253.
[36] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. 2011. Memory Power Management via Dynamic Voltage/Frequency Scaling. Association for Computing Machinery, New York, NY, USA.
[37] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.
[38] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. 2021. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook. *Proc. IEEE* 109, 5 (2021), 911–934.
[39] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. 2012. CoScale: Coordinating CPU and Memory System DVFS in Server Systems. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*.
[40] Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F. Wenisch, and Ricardo Bianchini. 2011. MemScale: Active Low-Power Modes for Main Memory. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1950365.1950392
[41] R.H. Dennard, F.H. Gaensslen, Hwa-Nien Yu, V.L. Rideout, E. Bassous, and A.R. LeBlanc. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9, 5 (1974), 256–268.
[42] Linux Kernel Documentation. 2020. Power Capping Framework. (2020). https://www.kernel.org/doc/Documentation/power/powercap/powercap.txt
[43] Linux Kernel Documentation. 2021. CPU Frequency Scaling. (2021). https://wiki.archlinux.org/index.php/CPU_frequency_scaling
[44] Linux Kernel Documentation. 2021. Device Frequency Scaling. (2021). https://www.kernel.org/doc/html/latest/driver-api/devfreq.html
[45] Linux Kernel Documentation. 2021. Linux voltage and current regulator framework. (2021). https://www.kernel.org/doc/Documentation/power/regulator/overview.txt
[46] Linux Kernel Documentation. 2021. NO_HZ: Reducing Scheduling-Clock Ticks. (2021). https://www.kernel.org/doc/Documentation/timers/no_hz.rst
[47] Linux Kernel Documentation. 2021. PM Quality of Service Interface. (2021). https://www.kernel.org/doc/Documentation/power/pm_qos_interface.txt
[48] Linux Kernel Documentation. 2021. Runtime Power Management Framework for I/O Devices. (2021). https://www.kernel.org/doc/Documentation/power/runtime_pm.txt
[49] Linux Kernel Documentation. 2021. System Power Management Sleep States. (2021). https://www.kernel.org/doc/Documentation/power/states.txt
[50] J. Doweck, W. Kao, A. K. Lu, J. Mandelblat, A. Rahatekar, L. Rappoport, E. Rotem, A. Yasin, and A. Yoaz. 2017. Inside 6th-Generation Intel Core: New Microarchitecture Code-Named Skylake. *IEEE Micro* 37, 2 (2017), 52–62.
[51] Jonathan Eastep, Steve Sylvester, Christopher Cantalupo, Brad Geltz, Federico Ardanaz, Asma Al-Rawi, Kelly Livingston, Fuat Keceli, Matthias Maiterth, and Siddhartha Jana. 2017. Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions. In *High Performance Computing*, Julian M. Kunkel, Rio Yokota, Pavan Balaji, and David Keyes (Eds.). Springer International Publishing, Cham, 394–412.
[52] R. Efraim, R. Ginosar, C. Weiser, and A. Mendelson. 2014. Energy Aware Race to Halt: A Down to EARtH Approach for Platform Energy Management. *IEEE Computer Architecture Letters* 13, 1 (2014), 25–28.
[53] Semiconductor Engineering. 2020. Software Defined Hardware gains ground again. https://semiengineering.com/software-defined-hardware-gains-ground-again/.

[54] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2012. Power Limitations and Dark Silicon are Challenging the Future of Multicore. *ACM Transcations on Computer Systems (TOCS)* (2012).
[55] Daniel Etiemble. 2018. 45-year CPU evolution: one law and two equations. arXiv:1803.00254 [cs.AR]
[56] Giorgos Fagas, John P. Gallagher, Luca Gammaitoni, and Douglas J. Paul. 2017. Energy Challenges for ICT. In *ICT - Energy Concepts for Energy Efficiency and Sustainability*, Giorgos Fagas, Luca Gammaitoni, John P. Gallagher, and Douglas J. Paul (Eds.). IntechOpen, Rijeka, Chapter 1.
[57] Clément Farabet, Berin Martini, Benoit Corda, Polina Akselrod, Eugenio Culurciello, and Yann LeCun. 2011. Neuflow: A runtime reconfigurable dataflow processor for vision. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on.* IEEE, 109–116.
[58] Richard Phillips Feynman, Anthony J. Hey, and Robin W. Allen. 2000. *Feynman Lectures on Computation.* Perseus Books, USA.
[59] J. A. Fisher. 1981. Trace Scheduling: A Technique for Global Microcode Compaction. 30, 7 (1981).
[60] Sagi Fisher, Adam Teman, Dmitry Vaysman, Alexander Gertsman, Orly Yadid-Pecht, and Alexander Fish. 2008. Digital subthreshold logic design - motivation and challenges. In *2008 IEEE 25th Convention of Electrical and Electronics Engineers in Israel.* 702–706.
[61] Adi Fuchs and David Wentzlaff. 2019. The Accelerator Wall: Limits of Chip Specialization. *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2019), 1–14.
[62] Steve Furber. 2014. SpinNNaker: The world's biggest NoC. In *Networks-on-Chip (NoCS), 2014 Eighth IEEE/ACM International Symposium on.* IEEE, ii–ii.
[63] Yu Gan and Christina Delimitrou. 2018. The Architectural Implications of Microservices in the Cloud. *CoRR* abs/1805.10351 (2018).
[64] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. 2019. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud and Edge Systems. Association for Computing Machinery, New York, NY, USA.
[65] Antara Ganguly, Rajeev Muralidhar, and Virendra Singh. 2019. Towards Energy Efficient non-von Neumann Architectures for Deep Learning. *20th International Symposium on Quality Electronic Design (ISQED)* (2019), 335–342.
[66] Mingyu Gao, Jing Pu, Xuan Yang, Mark Horowitz, and Christos Kozyrakis. 2017. Tetris: Scalable and efficient neural network acceleration with 3d memory. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems.* ACM, 751–764.
[67] Sukhpal Singh Gill and Rajkumar Buyya. 2018. A Taxonomy and Future Directions for Sustainable Cloud Computing: 360 Degree View. *ACM Comput. Surv.* 51, 5, Article 104 (Dec. 2018), 33 pages.
[68] O. Golonzka, J. . Alzate, U. Arslan, M. Bohr, P. Bai, J. Brockman, B. Buford, C. Connor, N. Das, B. Doyle, T. Ghani, F. Hamzaoglu, P. Heil, P. Hentges, R. Jahan, D. Kencke, B. Lin, M. Lu, M. Mainuddin, M. Meterelliyoz, P. Nguyen, D. Nikonov, K. O'brien, J. O. Donnell, K. Oguz, D. Ouellette, J. Park, J. Pellegren, C. Puls, P. Quintero, T. Rahman, A. Romang, M. Sekhar, A. Selarka, M. Seth, A. J. Smith, A. K. Smith, L. Wei, C. Wiegand, Z. Zhang, and K. Fischer. 2018. MRAM as Embedded Non-Volatile Memory Solution for 22FFL FinFET Technology. In *2018 IEEE International Electron Devices Meeting (IEDM).* 18.1.1–18.1.4.
[69] The Green Grid. 2020. https://www.thegreengrid.org/.
[70] The Energy Efficient High Performance Computing (EEHPC) Group. 2019. https://eehpcwg.llnl.gov/index.html.
[71] John Gurd, Wim Bohm, and Yong Meng Teo. 1987. Performance issues in dataflow machines. *Future Generation Computer Systems* 3, 4 (1987), 285–297.
[72] Laszlo Gyongyosi and Sandor Imre. 2019. A Survey on quantum computing technology. *Computer Science Review* 31 (02 2019), 51–71.
[73] Heonjae Ha. 2018. Understanding and Improving the Energy Efficiency of DRAM, PhD Thesis, Stanford University. (2018). https://searchworks.stanford.edu/view/12819402
[74] Heonjae Ha, Ardavan Pedram, Stephen Richardson, Shahar Kvatinsky, and Mark Horowitz. 2016. Improving Energy Efficiency of DRAM by Exploiting Half Page Row Access. IEEE Press.
[75] D. Hackenberg, R. SchÃűne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop.* 896–904.
[76] Ramyad Hadidi, Bahar Asgari, Burhan Ahmad Mudassar, Saibal Mukhopadhyay, Sudhakar Yalamanchili, and Hyesoon Kim. 2017. Demystifying the characteristics of 3D-stacked memories: A case study for Hybrid Memory Cube. In *2017 IEEE International Symposium on Workload Characterization (IISWC).*
[77] Jawad Haj-Yahya, Mohammed Alser, Jeremie Kim, A. Giray Yağlıkçı, Nandita Vijaykumar, Efraim Rotem, and Onur Mutlu. 2020. SysScale: Exploiting Multi-Domain Dynamic Voltage and Frequency Scaling for Energy Efficient Mobile Processors. IEEE Press.
[78] J. Haj-Yahya, E. Rotem, A. Mendelson, and A. Chattopadhyay. 2019. A Comprehensive Evaluation of Power Delivery Schemes for Modern Microprocessors. In *20th International Symposium on Quality Electronic Design (ISQED).* 123–130.
[79] J. Haj-Yahya, Y. Sazeides, M. Alser, E. Rotem, and O. Mutlu. 2020. Techniques for Reducing the Connected-Standby Energy Consumption of Mobile Devices. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA).* 623–636.

[80] Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz. 2010. Understanding Sources of Inefficiency in General-Purpose Chips. Association for Computing Machinery, New York, NY, USA.

[81] James Hamilton. 2017. Data Center Power and Water Consumption. (2017). https://perspectives.mvdirona.com/2015/06/data-center-power-water-consumption/

[82] John L. Hennessy and David A. Patterson. 2019. A New Golden Age for Computer Architecture. *Commun. ACM* 62, 2 (Jan. 2019), 48âĂŞ60.

[83] D.S. Henry, B.C. Kuszmaul, and V. Viswanath. 1999. The Ultrascalar processor-an asymptotically scalable superscalar microarchitecture. In *Proceedings 20th Anniversary Conference on Advanced Research in VLSI*. 256–273.

[84] M.D. Hill and M.R. Marty. 2008. Amdahl's Law in the Multicore Era. *Computer* 41, 7 (july 2008), 33–38.

[85] IEEE. 2020. IEEE Rebooting Computing Initiative. https://rebootingcomputing.ieee.org/.

[86] Shashikant Ilager, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2021. Thermal Prediction for Efficient Energy Management of Clouds Using Machine Learning. *IEEE Transactions on Parallel and Distributed Systems* 32, 5 (2021), 1044–1056.

[87] Intel. 2007. From a Few Cores to Many: A Tera-scale Computing Research Overview. https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/intel-labs-tera-scale-research-paper.pdf/.

[88] Intel. 2020. Current and Power Limit Throttling Indicators in Intel Processors. (2020). https://www.intel.in/content/www/in/en/support/articles/000039154/processors/intel-core-processors.html

[89] Intel. 2020. The Intel Running Average Power Limiter. (2020). https://01.org/blogs/2014/running-average-power-limit

[90] Intel. 2020. What Intel Is Planning for The Future of Quantum Computing: Hot Qubits, Cold Control Chips, and Rapid Testing. (2020). https://spectrum.ieee.org/tech-talk/computing/hardware/intels-quantum-computing-plans-hot-qubits-cold-control-chips-and-rapid-testing

[91] Qualcomm Hexagon 685 DSP is a Boon for Machine Learning. 2017. https://www.xda-developers.com/qualcomm-snapdragon-845-hexagon-685-dsp/.

[92] Q. Jiang, Y. C. Lee, and A. Y. Zomaya. 2020. The Power of ARM64 in Public Clouds. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. 459–468.

[93] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*. IEEE, 1–12.

[94] Himanshu Kaul, Mark Anders, Steven Hsu, Amit Agarwal, Ram Krishnamurthy, and Shekhar Borkar. 2012. Near-Threshold Voltage (NTV) Design: Opportunities and Challenges. In *Proceedings of the 49th Annual Design Automation Conference* (San Francisco, California) *(DAC âĂŹ12)*. Association for Computing Machinery, New York, NY, USA, 1153âĂŞ1158.

[95] Stefanos Kaxiras and Margaret Martonosi. 2008. *Computer Architecture Techniques for Power-Efficiency* (1st ed.). Morgan and Claypool Publishers.

[96] Linux Kernel. 2021. The Intel P-state Driver. (2021). https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt

[97] M. Krstic, E. Grass, F. K. GÃijrkaynak, and P. Vivet. 2007. Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook. *IEEE Design Test of Computers* 24, 5 (2007), 430–441.

[98] E. KÃijltÃijrsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu. 2013. Evaluating STT-RAM as an energy-efficient main memory alternative. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 256–267.

[99] Rolf Landauer. 2000. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development* 44 (01 2000), 261–269.

[100] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2010. Phase Change Memory Architecture and the Quest for Scalability. 53, 7 (2010).

[101] Dong Uk Lee, Kyung Whan Kim, Kwan Weon Kim, Kang Seol Lee, Sang Jin Byeon, Jae Hwan Kim, Jin Hee Cho, Jaejin Lee, and Jun Hyun Chun. 2015. A 1.2 V 8 Gb 8-channel 128 GB/s high-bandwidth memory (HBM) stacked DRAM with effective I/O test circuits. *IEEE Journal of Solid-State Circuits* 50, 1 (2015), 191–203.

[102] Sukhan Lee, Yuhwan Ro, Young Hoon Son, Hyunyoon Cho, Nam Sung Kim, and Jung Ho Ahn. 2017. Understanding power-performance relationship of energy-efficient modern DRAM devices. In *2017 IEEE International Symposium on Workload Characterization (IISWC)*. 110–111.

[103] Woojoo Lee. 2016. Tutorial: Design and Optimization of Power Delivery Networks. *IEIE Transactions on Smart Processing and Computing* 5 (10 2016), 349–357.

[104] Jiajun Li, Guihai Yan, Wenyan Lu, Shuhao Jiang, Shijun Gong, Jingya Wu, and Xiaowei Li. 2018. SmartShuttle: Optimizing off-chip memory accesses for deep learning accelerators. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 343–348.

[105] P. Li and Y. Luo. 2016. P4GPU: Accelerate packet processing of a P4 program with a CPU-GPU heterogeneous architecture. In *2016 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. 125–126.

[106] W. Liang, S. Chen, Y. Chang, and J. Fang. 2008. Memory-Aware Dynamic Voltage and Frequency Prediction for Portable Devices. In *2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. 229–236.
[107] Sung Kyu Lim. 2010. 3D circuit design with through-silicon-via: Challenges and opportunities. In *IEEE Electronic Design Processes Symposium Workshop*.
[108] Linux. 2020. Linux Kernel Documentation. https://www.kernel.org/doc/html/latest/.
[109] Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu. 2012. RAIDR: Retention-Aware Intelligent DRAM Refresh. IEEE Computer Society, USA.
[110] C. A. Mack. 2011. Fifty Years of Moore's Law. *IEEE Transactions on Semiconductor Manufacturing* 24, 2 (2011), 202–207.
[111] J. A. Mandelman, R. H. Dennard, G. B. Bronner, J. K. DeBrosse, R. Divakaruni, Y. Li, and C. J. Radens. 2002. Challenges and future directions for the scaling of dynamic random-access memory (DRAM). *IBM Journal of Research and Development* 46, 2.3 (2002), 187–212.
[112] ARM Developer Manuals. 2020. Energy Aware Scheduling and Multi Cluster PM. (2020). https://developer.arm.com/tools-and-software/open-source-software/linux-kernel/energy-aware-scheduling
[113] Igor L. Markov. 2014. Limits on fundamental limits to computation. *Nature* 512, 7513 (Aug 2014), 147âĂŞ154.
[114] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios Vasilakos. 2015. Cloud Computing: Survey on Energy Efficiency. *Comput. Surveys* 47 (01 2015), 36.
[115] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673.
[116] Amirhossein Mirhosseini, Akshitha Sriraman, and Thomas F. Wenisch. 2019. Enhancing Server Efficiency in the Face of Killer Microseconds. In *Proceedings of the 25th International Symposium on High-Performance Computer Architecture (HPCA âĂŹ19)*. IEEE, 185–198.
[117] Sparsh Mittal and Jeffrey S. Vetter. 2014. A Survey of Methods for Analyzing and Improving GPU Energy Efficiency. *ACM Comput. Surv.* 47, 2, Article 19 (Aug. 2014), 23 pages.
[118] Sparsh Mittal and Jeffrey S. Vetter. 2015. A Survey of CPU-GPU Heterogeneous Computing Techniques. 47, 4 (2015).
[119] Jugdutt Singh Mohsen Radfar, Kriyang Shah. 2012. Recent Subthreshold Design Techniques. In *Active and Passive Electronic Components*.
[120] Gordon E. Moore. 1965. Cramming more components onto integrated circuits. *Electronics* 38, 8 (April 1965).
[121] G. W. K. Moore. 2003. No exponential is forever: but "Forever" can be delayed! [semiconductor industry]. *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC.* (2003), 20–23 vol.1.
[122] Onur Mutlu. 2013. Memory scaling: A systems architecture perspective. In *2013 5th IEEE International Memory Workshop*.
[123] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. 2020. A Modern Primer on Processing in Memory. *CoRR* abs/2012.03112 (2020). arXiv:2012.03112 https://arxiv.org/abs/2012.03112
[124] Chris Nicol. 2017. A coarse grain reconfigurable array (cgra) for statically scheduled data flow computing. *Wave Computing White Paper* (2017).
[125] Tony Nowatzki, Vinay Gangadhar, and Karthikeyan Sankaralingam. 2015. Exploring the Potential of Heterogeneous Von Neumann/Dataflow Execution Models. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture* (Portland, Oregon) *(ISCA '15)*. ACM, New York, NY, USA, 298–310.
[126] Internet of Things Agenda. 2020. Startups target subthreshold to solve IoT power consumption challenge. https://internetofthingsagenda.techtarget.com/feature/Startups-target-subthreshold-to-solve-IoT-power-consumption-challenge.
[127] International Standards Organization. 2016. Data centres facilities and infrastructure: Power usage effectiveness. (2016). https://www.en-standard.eu/csn-en-50600-4-2-information-technology-data-centre-facilities-and-infrastructures-part-4-2-power-usage-effectiveness/
[128] International Standards Organization. 2016. Data centres Key performance indicators: Power usage effectiveness. (2016). https://www.iso.org/standard/63451.html
[129] Kalin Ovtcharov, Olatunji Ruwase, Joo-Young Kim, Jeremy Fowers, Karin Strauss, and Eric Chung. 2015. Accelerating Deep Convolutional Neural Networks Using Specialized Hardware.
[130] S. Palacharla, N.P. Jouppi, and J.E. Smith. 1997. Complexity-Effective Superscalar Processors. In *Conference Proceedings. The 24th Annual International Symposium on Computer Architecture*. 206–218.
[131] Venkatesh Pallipadi. 2007. cpuidle - Do nothing, efficiently... http://ols.108.redhat.com/2007/Reprints/pallipadi-Reprint.pdf
[132] Venkatesh Pallipadi and Alexey Starikovskiy. 2006. The Ondemand Governor: Past, Present and Future. (2006). https://www.kernel.org/doc/ols/2006/ols2006v2-pages-223-238.pdf
[133] David Patterson and Andrew Waterman. 2017. *The RISC-V Reader: An Open Architecture Atlas.* Strawberry Canyon.
[134] Peripheral Component Interconnect Special Interest Group (PCI-SIG). 2020. PCI Specifications. (2020). https://pcisig.com/
[135] Thomas Ernst Peide D. Ye and Mukesh V. Khare. 2019. The Nanosheet Transistor Is the Next (and Maybe Last) Step in MooreâĂŹs Law. (2019). https://spectrum.ieee.org/semiconductors/devices/the-nanosheet-transistor-is-the-next-and-maybe-last-step-in-moores-law

[136] Luis A Plana, Steve B Furber, Steve Temple, Mukaram Khan, Yebin Shi, Jian Wu, and Shufan Yang. 2007. A GALS infrastructure for a massively parallel multiprocessor. *IEEE Design & Test of Computers* 24, 5 (2007).

[137] Next Platform. 2021. Amazon goes wide and deep with Graviton3 server chip. (2021). https://www.nextplatform.com/2021/12/02/aws-goes-wide-and-deep-with-graviton3-server-chip/

[138] IBM Research. 2020. IEDM 2020: Advances in memory, analog AI and interconnects point to the future of hybrid cloud and AI. (2020). https://www.ibm.com/blogs/research/2020/12/iedm2020-memory-analog-ai/

[139] Karl Rupp. 2018. 42 Years of Microprocessor Trend Data. https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/.

[140] Michael S. Schlansker and B. Ramakrishna Rau. 2000. EPIC: An Architecture for Instruction-Level Parallel Processors. (2000). https://www.hpl.hp.com/techreports/1999/HPL-1999-111.pdf

[141] Robert Schöne, Thomas Ilsche, Mario Bielert, Andreas Gocht, and Daniel Hackenberg. 2019. Energy Efficiency Features of the Intel Skylake-SP Processor and Their Impact on Performance. *CoRR* abs/1905.12468 (2019). arXiv:1905.12468 http://arxiv.org/abs/1905.12468

[142] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. *CoRR* abs/1907.10597 (2019). arXiv:1907.10597 http://arxiv.org/abs/1907.10597

[143] Amazon Web Services. 2020. Amazon Braket âĂŞ Get Started with Quantum Computing. (2020). https://aws.amazon.com/blogs/aws/amazon-braket-get-started-with-quantum-computing/

[144] Yakun Shao, Brandon Reagen, Gu-Yeon Wei, and David Brooks. 2014. Aladdin: A pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures. *Proceedings of International Symposium on Computer Architecture*, 97–108.

[145] Anirudh Sivaraman, Alvin Cheung, Mihai Budiu, Changhoon Kim, Mohammad Alizadeh, Hari Balakrishnan, George Varghese, Nick McKeown, and Steve Licking. 2016. Packet Transactions: High-Level Programming for Line-Rate Switches. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 15âĂŞ28. https://doi.org/10.1145/2934872.2934900

[146] Gurindar S. Sohi, Scott E. Breach, and T. N. Vijaykumar. 1995. Multiscalar Processors. Association for Computing Machinery, New York, NY, USA.

[147] D. Suggs, M. Subramony, and D. Bouvier. 2020. The AMD âĂIJZen 2âĂİ Processor. *IEEE Micro* 40, 2 (2020), 45–52.

[148] Siddha Suresh, Pallipadi Venkatesh, Van Arjan, and De Ven. 2007. Getting maximum mileage out of tickless. (01 2007).

[149] Emil Talpes, Atchyuth Gorti, Gagandeep Sachdev, Debjit Sarma, Ganesh Venkataramanan, Peter Bannon, Bill McGee, Benjamin Floering, Ankit Jalote, Chris Hsiong, and Sahil Arora. 2020. Compute Solution for Tesla's Full Self Driving Computer. *IEEE Micro* PP (02 2020), 1–1.

[150] Anand Tech. 2018. The iPhone XS and XS Max Review: Unveiling the Silicon Secrets. (2018). https://www.anandtech.com/show/13392/the-iphone-xs-xs-max-review-unveiling-the-silicon-secrets

[151] IMG Tech. 2018. Power VR Series 3NX Neural Network Accelerator. https://www.imgtec.com/vision-ai/powervr-series3nx/.

[152] Chetan Singh Thakur, Jamal Lottier Molin, Gert Cauwenberghs, Giacomo Indiveri, Kundan Kumar, Ning Qiao, Johannes Schemmel, Runchun Wang, Elisabetta Chicca, Jennifer Olson Hasler, Jae-sun Seo, Shimeng Yu, Yu Cao, AndrÃľ van Schaik, and Ralph Etienne-Cummings. 2018. Large-Scale Neuromorphic Spiking Array Processors: A Quest to Mimic the Brain. *Frontiers in Neuroscience* 12 (2018).

[153] Berkeley The University of California. 2011. Magnetic memory and logic could achieve ultimate energy efficiency. https://news.berkeley.edu/2011/07/01/magnetic-memory-and-logic-could-achieve-ultimate-energy-efficiency/.

[154] EE Times. 2020. AI at the very very Edge. https://www.eetimes.com/ai-at-the-very-very-edge/.

[155] R. M. Tomasulo. 1967. An Efficient Algorithm for Exploiting Multiple Arithmetic Units. *IBM Journal of Research and Development* 11, 1 (1967), 25–33.

[156] E. Vasilakis, I. Sourdis, V. Papaefstathiou, A. Psathakis, and M. G. H. Katevenis. 2017. Modeling energy-performance tradeoffs in ARM big.LITTLE architectures. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 1–8.

[157] Arthur H Veen. 1986. Dataflow machine architecture. *ACM Computing Surveys (CSUR)* 18, 4 (1986), 365–396.

[158] Vasanth Venkatachalam and Michael Franz. 2005. Power Reduction Techniques for Microprocessor Systems. *ACM Comput. Surv.* 37, 3 (Sept. 2005), 195âĂŞ237.

[159] David W. Wall. 1991. Limits of Instruction-Level Parallelism. *SIGPLAN Not.* 26, 4 (April 1991), 176âĂŞ188.

[160] HPC Wire. 2020. Arm Technology Powers the WorldâĂŹs Fastest Supercomputer. (2020). https://www.hpcwire.com/off-the-wire/arm-technology-powers-the-worlds-fastest-supercomputer/

[161] A. Yakovlev. 2011. Energy-modulated computing. In *2011 Design, Automation Test in Europe*. 1–6.

[162] Tien-Ju Yang, Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2017. A Method to Estimate the Energy Consumption of Deep Neural Networks. *Energy* 1, L2 (2017), L3.