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

Preprint · July 2020

CITATIONS
0

READS
6,594

3 authors:

Rajeev Muralidhar
University of Melbourne
8 PUBLICATIONS **71** CITATIONS
SEE PROFILE

Renata Borovica-Gajic
University of Melbourne
46 PUBLICATIONS **397** CITATIONS
SEE PROFILE

Rajkumar Buyya
University of Melbourne
1,023 PUBLICATIONS **104,571** CITATIONS
SEE PROFILE

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

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

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 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 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 - (a) *specification* - the ability to precisely specify the power intent, attributes or properties at different layers (b) *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, (c) *techniques* used for implementing energy efficiency at different levels of the stack, (d) *verification* techniques used to provide guarantees that the functionality of complex designs are preserved, and (e) *energy efficiency 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, Power Specification, Power Modeling, Low Power Optimizations, RTL Power Optimizations, Platform-Level Power Management, Dynamic Power Management, Survey

1 INTRODUCTION

The computing industry has gone through tremendous change in the last few decades. While Moore's law [82] drove the semiconductor industry to cram more and more transistors and logic into the same volume, the end of Dennard's scaling [36] limited how much we could shrink voltage and current without losing predictability, and the Instruction Level Parallelism (ILP) wall [119] defined the start of the multi-core and tera-scale era [61]. 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 [96] shows

Authors' addresses: Rajeev Muralidhar, rajeev.muralidhar@student.unimelb.edu.au,rajeevm@ieee.org, The University of Melbourne and Amazon Web Services, Parkville, Victoria, 3010; Renata Borovica-Gajic, renata.borovica@unimelb.edu.au, The University of Melbourne, Parkville, Victoria, 3010; Rajkumar Buyya, rbuyya@unimelb.edu.au, The University of Melbourne, Parkville, Victoria, 3010.

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 [57]. The graph, as well as studies such as [44], 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, power density continues to raise across the spectrum of computing devices. 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 GPUs, imaging, 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. Newer memory technologies like NVRAM, Intel's Optane, STT-SRAM, and interfaces such as Hybrid Memory Cube (HMC) [55] and High Bandwidth Memory (HBM) [70] that enable high-performance RAM interfaces have however, pushed the boundaries of the memory wall. The more recent Compute Express Link (CXL) [27] is a recent industry standard for integrating accelerators, memory and compute elements. Deep learning has also triggered relooking 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, as detailed in [48].

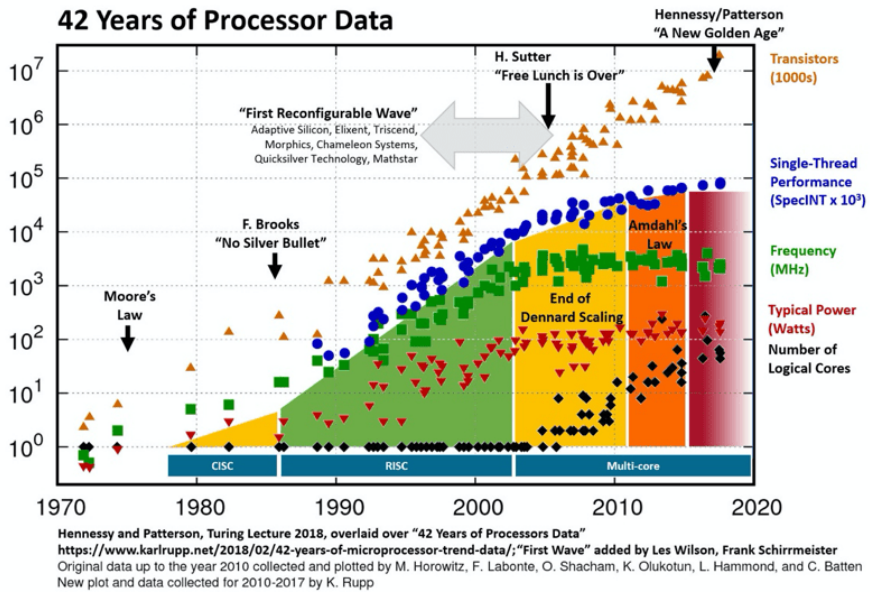

Fig. 1. 42 years of microprocessor trend data [96], [57], [63]

The nature of computing systems has thus changed across the spectrum of devices, from being pure compute-based to being a mixture of CPUs, GPUs, accelerators and FPGAs. Such heterogeneous capabilities are now also available on "edge devices" such as the Raspberry PI, Google's Coral Tensor Processing Unit [63] and Intel's Movidius [29]. From a hardware perspective, as devices have

Table 1. Summary of Energy Efficiency Related Surveys

Topic	Key survey or book
Energy efficiency/sustainability, metrics in cloud	[78], [50]
Energy efficiency techniques in hardware, circuits	[116]
Hardware techniques for energy efficiency in CPUs, GPUs	[116], [81]
Energy Efficiency of compute nodes	[67]
Energy efficiency at data center level	[11]

shrunk, the industry is struggling to eliminate the effects of thermodynamic fluctuations, which are unavoidable at lower technology nodes (sub-10nm scale). While we try to make energy efficient hardware architectures, recent research has shown that machine learning consumes significant energy [98]. 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 greatly accelerated the limitations imposed by fundamental physics and architectural, power and thermal walls. Decades of energy efficiency technologies across different areas have made such systems, form factors and workloads possible; however, energy consumption of computing systems is now on the rise as never before.

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 [106]). Systems have thus 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.

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 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. The research surveys conducted so far can be categorized as 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. 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 OS techniques of sleep state selection (via Linux idle power management idle governor[75], for example) when trying to evaluate low power sleep states, 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 [8]. 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 DVFS state 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 [7] 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.

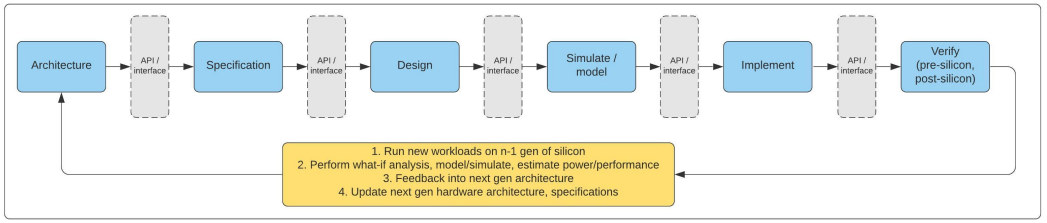

Fig. 2. Phases of energy efficient system design

Energy efficiency in HPC systems has also become important of late. The Energy Efficient HPC (EEHPC) [52] 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) [38] 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 most of these 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, 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) *Energy Efficiency Technique*: This could be at different levels of the hierarchy - circuit/RTL, microarchitecture, CPU, GPU or other accelerators, or at software/system level.
(2) *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).
(3) *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).
(4) *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.
(5) *Energy Standards*: 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

The rest of the 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 microarchitectural techniques used in CPUs, GPUs, memory and domain-specific accelerators.
(3) Section 4 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. We survey specifications and abstractions at different levels of the hierarchy.
(4) Section 5 covers *modeling and simulation* of power, performance and thermal dissipation across processors, GPUs, accelerators, SOC and complete systems. We describe some state of the art modeling and simulation tools and technologies in use today.
(5) In Section 6, we cover *system and software techniques* used for energy efficiency.
(6) Section 7 covers *verification* of power management design, techniques and transformations in large chips and systems.

Table 2. Trends in system architecture and energy efficiency

Architectural Trends	Energy Efficiency Features
Moore's Law, ILP wall, Dennard Scaling	Increased performance via superscalar, VLIW arch, Clock/power gating, processor, cache, memory sleep states, DVFS, power delivery improvements
Multi-cores, Amdahl's limit, on-die voltage regulators	OS guided / controlled sleep states, fine grained clock/power gating, per-core, per-module DVFS, on-die voltage regulators
Memory wall, newer memory technologies	Memory DVFS, compiler/software techniques
Domain-specific architectures	Chip/IP-level clock/power gating, DVFS
Dark silicon challenges	Fine grained power domains and islands
Specialized interconnects	Low power, high-bandwidth standards like CXL[27] and PCIe 5.0)
non-von Neumann architectures	Energy-aware dataflow architectures
Combining von Neumann and non-von Neumann chips	Emerging area, mix of different techniques
Power delivery miniaturization	On-die/chip voltage regulators, software control, reconfigurable power delivery
Programmable architectures - FPGAs	Energy-aware FPGAs, still in nascent stage
Energy Proportional Computing	Energy-aware data centers, system components
Near/sub-threshold voltage designs, 3D stacking, and chiplets	Ultra low voltage designs, Thermal algorithms
Thermodynamic computing, Landauer Limit and Quantum Computing	Emerging areas, system architectures unclear / evolving

(7) In Section 8, we survey *energy efficiency standards and consortiums* that are trying to address energy efficiency through regulations, standardization of abstractions, energy/performance models and cross-layer optimizations.
(8) In Section 9 and Section 10, we will discuss the road ahead for next generation of energy efficient systems.

2 ARCHITECTURAL TRENDS AND SYSTEM LEVEL ENERGY EFFICIENCY

John Hennessy and David Patterson, in their recent ACM Turing award lecture and publication [57] 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 [91]. In this section, we elaborate on some of the key observations highlighted in [91], 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 [77] 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 scaling, ILP Wall and the end of Dennard Scaling

MooreâĂŹs Law [82] (it is more an observation than a real law) 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 [83]. In addition to this, there have been other important laws that have shaped computer systems. One such is Dennard Scaling [36]. 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 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). Computer architects eventually ran out of ILP that could be exploited efficiently around 2003-2004 [119] thereby forcing the industry to switch from a single energy-hogging processors to multiple efficient processors or many cores per chip, ushering in the many/multi-core era. There are also hybrid designs that combined low power/low performance and high power/high performance cores, like ARM's BIG.LITTLE architecture [114] and the recent Intel Lakefield chip [33]. Dennard scaling thus ended about 30 years after it was first observed, primarily because current and voltage could not keep dropping while remaining dependable. Recently, near-threshold and sub-threshold voltage technologies [89] are attempting to push these boundaries.

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. The authors in [58] elaborate on the impact of this law on multi-core chips. Figure 3, from [62] illustrates the effect of these three laws on processor performance for the past 40 years. At the current rate, performance on standard processor benchmarks will not double before 2038, while transistor density, power consumption and power density continue to rise.

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 enormous 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 [42] 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*. Even at 22 nm, 21% of a fixed-size chip is powered off, and at 8 nm, this number grows to more than 50%. Software and operating system guided

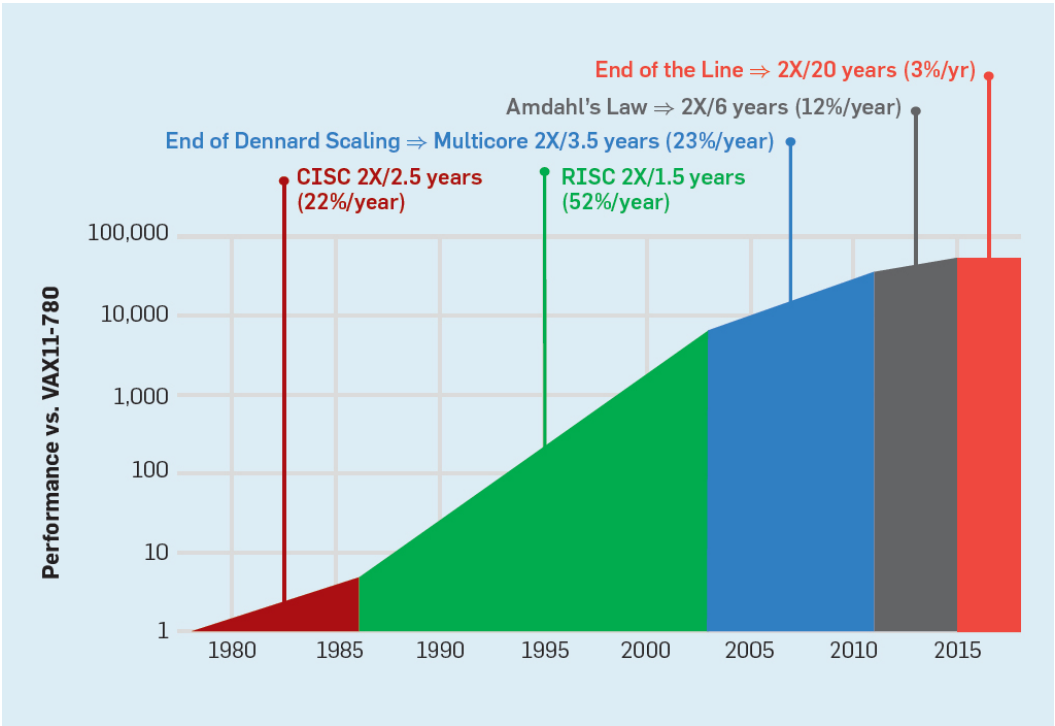

Fig. 3. Highest SPECCPUint performance per year for processor cores over the past 40 years [62]

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.

2.4 Memory wall, domain-specific architectures and the limits of chip specialization

Computational workloads like deep learning involve repetitive operations on large data sets. Moving data from memory to the processing unit and back turned out to be a limiting factor for both performance and power consumption. This 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. Newer technologies have now pushed the boundaries of this *memory wall* through newer memory techniques such as High Bandwidth Memory (HBM) [70], Hybrid Memory Cube (HMC) [55], in- and near-memory compute. Compute-in-memory architectures seek to remove this bottleneck by integrating memory and computation into a single circuit block, like the multiply-and-accumulate matrix operations required for neural network operations. Such trends have recently led to *domain-specific accelerators*, the key idea being to design 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 processors for software-defined networks (SDNs) for high speed packet processing. However, much of the benefits of chip specialization stems from optimizing a computational problem within a given chipâĂŹs transistor budget. As detailed in [47], 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.

2.5 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) [27] is an industry standard to integrating accelerators, memory and compute elements. 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.* Over the last couple of decades, operating system and software-guided power management infrastructures, frameworks, and algorithms have evolved rapidly to optimize these devices and systems. With Linux and embedded real time operating systems largely leading the way through different innovations like tickless operating systems, DVFS frameworks/governors, idle and runtime power management and system standby states (only to name a few), Windows has also implemented Connected Standby, Modern standby [34] and several more energy efficiency strategies and algorithms to manage idle and active workloads.

2.6 Advent of non-von Neumann architectures

Traditional architectures have largely followed the von Neumann computing model. Of the major deviations from von Neumann architectures, hardware dataflow machines proposed a couple of decades ago [35], [115] were designed to provide non-von Neumann architectural support to systems. However, they were severely limited by the availability of data movement infrastructures, effective software parallelism and functional units in hardware [53]. However, the revival of dataflow or near-dataflow architectures is driven by both advances in process/memory technologies and the nature of neural workloads. Some recent chips have implemented non-von Neumann computing models like dataflow, neuromorphic and spiking neural networks. 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 [121], 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. The authors in [48] discuss non-von Neumann architectures in more detail.

2.7 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 gaining traction. Nowatzki et al. [88] 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) [32] 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 the instruction set of a specific processor, 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 4. 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 [60] could be extended to consider this aspect as well in addition to its existing charter.

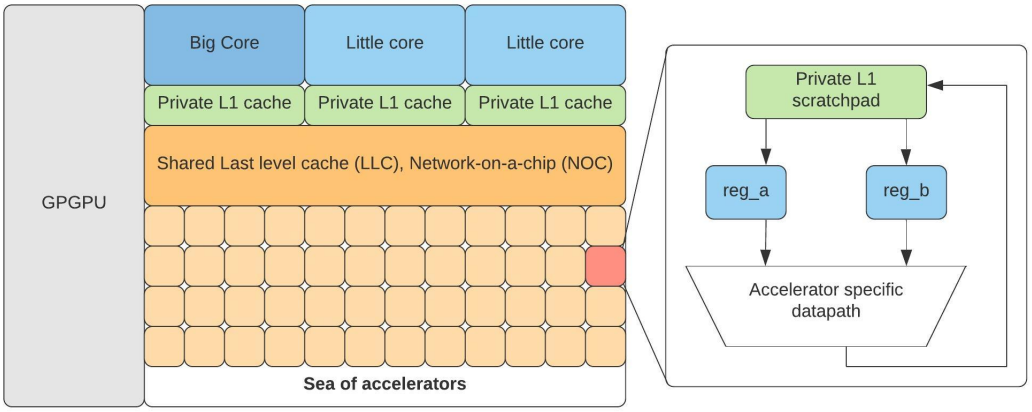

Fig. 4. Future heterogeneous architectures [99]

2.8 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. The authors in [56] 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 [71]. 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).

2.9 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 [43]. 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.

2.10 Energy Proportional Computing

In 2007, the concept of *energy proportional computing* was first proposed by Google engineers Luiz Andre Barroso and Urs Holzle [12]. 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 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)[74], Near and sub Threshold Voltage (NTV) [66] designs, newer memory integration technologies, and more recently chiplets. Intel's Foveros (chiplets) [33] 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.

2.12 Thermodynamic computing, Landauer Limit and Quantum Computing

Richard Feynman, in his classic work [46] laid down the foundations of thermodynamic and quantum computing, which are now on the horizon. As detailed in the recent report on thermodynamic computing [28], 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. The scale of these computing systems is shown in Figure 5. In the figure, spatial and temporal fluctuation scales are estimated in terms of thermal energy (kT) and corresponding electronic quantum coherence times and lengths.

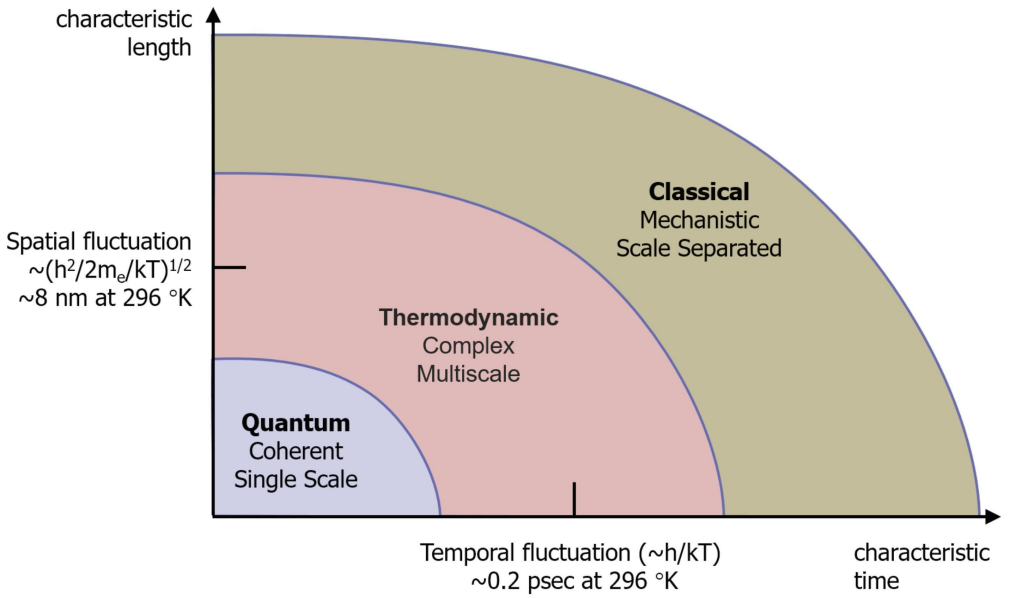

Fig. 5. Comparing scales of classical, quantum and thermodynamic computing [28]

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* [69] 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 [105] 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 [16] and [54]. 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 MICROARCHITECTURAL TECHNIQUES

The fundamental techniques for energy efficiency involve fine-grained clock/power gating and dynamic voltage frequency scaling (DVFS). The basics of these techniques and thermal dissipation/management are described in detail in Kaxiras and Martonosi [67]. In this section, we focus on key microarchitectural techniques for energy efficiency across CPU, caches, memory and domain specific accelerators like GPUs and deep learning chips.

3.1 Microarchitectural techniques for CPUs

Power management for microprocessors can be done over the whole processor, or in specific areas. CPUs can have their execution suspended simply by stopping the issuance of instructions or by turning off their clock circuitry. Deeper power states successively remove power from the processorâĂŹs caches, translation lookaside buffers (TLBs), memory controllers, and so on. Deeper power states incure higher latency, and therefore extra energy is required to save and restore the hardware contents, or restart it. Modern processors support multiple low power states that can be exploited either by hardware (hardware idle detection) or through hints from the operating system scheduler based on heuristics such as next expected timer/interrupt, transition latency of different low power states, and current QoS setting dictated by other kernel components. As CPUs have evolved over the generations from single monolithic cores to multi-domain, multi-module and hybrid many core architectures, energy efficiency has been incorporated into different aspects. CPUs employ the following energy efficiency techniques:

(1) *Clock gating*: In this, the clock distribution to an entire functional unit in the processor is shutoff, thus saving dynamic (switching) power.
(2) *Power gating*: Here, entire functional units of the processor are disconnected from the power supply, thus consuming effectively zero power.
(3) *Multiple voltage domains*: Different portions of the chip are powered by different voltage regulators, so that each can be individually controlled for DVFS scaling power gating. Recent designs use on-die and on-chip voltage regulators that can do fine-grained power management through CPU microcode or low level firmware [56].
(4) *Multi-threshold voltage designs*: Different transistors in the design use different threshold voltages to optimize delay and/or power.
(5) *Dynamic frequency scaling (DFS)*: The clock frequency is adjusted statically or dynamically to achieve different power/performance trade-offs.
(6) *Dynamic voltage scaling (DVS)*: The supply voltage of the processor is adjusted statically or dynamically to achieve different power/performance and reliability trade-offs.

(7) *Dynamic voltage and frequency scaling (DVFS)*: Both voltage and frequency are varied dynamically to achieve better power/performance trade-offs than either DFS or DVS alone can provide.

Beyond the CPU cores, uncore components like caches, translation lookaside buffer and others, also implement energy efficiency techniques as embedded microprocessors devote nearly 40% of their power budget to uncore/caches. Current cache implementations use several techniques. Smart sizing caches is done by the micro code in the processor core. In [112], the authors define application specific cache partitions, called cache molecules, that are resized to address performance targets for applications. Some other examples include drowsy caches, dynamic clock gating based on operand width and instruction compression, among others; these are detailed in the book [67].

3.2 Microarchitectural techniques for Memory

Memory technology has evolved across DDR3/4/5, LPDDR, and more recently non-volatile memory (NVM) and these have enabled different levels of performance and power management with features such as clock frequency control and varying degrees of shallow/deep self-refresh. Newer memories like non volatile memory (NVM) exhibit different power and energy efficiency characteristics across reads, writes and self-refresh states. Recent system design, application, and technology trends that require more capacity, bandwidth, efficiency, and predictability out of the memory system make the memory system an important system bottleneck. At the same time, DRAM and flash technologies are experiencing technology scaling challenges that make the maintenance and enhancement of their capacity, energy efficiency, performance, and reliability significantly expensive with conventional techniques.

Energy efficiency in memory is important in the context of workloads like deep neural networks (DNNs). System designs that enable accelerated processing of DNNs with improved energy efficiency but without trading off accuracy or increasing hardware costs have become indispensable. Computing of such applications is governed by data movements rather than the execution of algorithmic or logical functions. Hence, dependence of system performance on the efficiency of processor-memory interaction is seeing an all-time high as we have striven to push beyond the memory wall [120], [79]. With memory technologies like 3D-stacked memories [76] and non-volatile memories [123], the *memory wall* issue is being addressed to some degree. However, the high bandwidth and greater storage capacity of such alternatives to conventional DDR systems for main memory can be helpful only if they are intelligently utilized by the system. This requires a synergy of the resource requirement of the workload with the available bandwidth, parallelism and data access hierarchy of the underlying memory system via hardware-software techniques. Micron's Hybrid Memory Cube (HMC) has made a compelling case for realization of a high throughput and low energy solution for massively parallel computations with their extensive bandwidths [92] facilitated by through silicon via (TSV) technology [74] and near-data processing (NDP) [10] in the logic layer. An apt architectural design of memory layers as well as the logic layer of HMC can enable the effective bandwidth to be as close as possible to the maximum available bandwidth [55], [93].

Some systems use partial array self-refresh (PASR), where memory is divided into banks, each of which can be powered up/down independently. If any of those banks of memory are not needed, that memory (and its self- refresh mechanism) can be turned off. The result is a reduction in power use, but data stored in the affected banks is also lost. Correspondingly, this requires operating system support for intelligent memory allocation.

3.3 Microarchitectural techniques for GPUs

Modern GPUs consume a significant amount of power - anywhere from $\tilde{5}$0-300W (or even more). However, GPUs provide better performance-per-watt than CPUs for specific workloads. The techniques for improving energy efficiency of GPUs largely overlap with those used for CPUs, with some variations and additions. A detailed survey is presented in [81] and some of the key techniques are highlighted here:

(1) **GPU DVFS**: Many current GPUs have separate clocks and voltage domains, thereby making them ideal candidates for clock/frequency scaling, voltage scaling, or both through hardware/software orchestration. Typically, in low power GPUs (in handhelds, for example), the chip is divided into three power domains - vertex shader, rendering engine, and RISC processor, and DVFS is individually applied to each of the three domains, thereby allowing for finer orchestration of the power domains.
(2) **CPU-GPU orchestration**: Instead of using a single GPU with each CPU, using multiple GPUs with each CPU enables achieving speedup in execution time and improving the usage of the CPU, thereby improving the energy efficiency of the system. Further, since during the execution of the CUDA kernel the host CPU remains in the polling loop without doing useful work, the frequency of the CPU can be reduced for saving energy while ensuring that CPU frequency is optimal for the bus between the CPU and GPU. Since the range of CPU frequencies is generally larger than that of the bus, CPU frequency can be scaled without affecting GPU performance. Also, for specific workloads, using CPU DVFS can be employed while it stays in busy-waiting for the GPU to complete computations, thereby achieving energy savings with little performance loss. Most of these can be orchestrated through hardware and software components.
(3) **Energy efficiency in GPU components**: GPU components such as caches, global memory, pixel and vertex shader can all be managed through dynamic clock and power gating. Since GPUs employ a large number of threads, storing the register context of these threads requires a large amount of on-chip storage. Also, the thread scheduler in the GPU needs to select a thread to execute from a large number of threads, access large register files, etc. which consumes substantial energy. Similarly, instruction pipeline, shared registers, last-level caches can also be made more energy efficient through hardware and microarchitectural techniques.

3.4 Microarchitectural techniques for AI accelerators

An AI accelerator chip has three main elements âĂŤ a large amount of data, algorithms to process the data (configurable by software), and the physical architecture where data processing/calculation is carried out. Such accelerators tend to have regular architectures - large arrays with hundreds or thousands of processors, arranged in clusters repeated across the chip and consuming power in the order of tens or even hundreds of watts. The key energy efficiency techniques for such chips comprise of hardware/software partitioning of the workload, mapping of data structures into on-chip and off-chip memory, grouping of components into power domains, power management policy (race-to-halt typically), and enter idle states when parts of the chip are idle. Designs typically also include many temperature sensors across the die âĂŤ for example, one per processing cluster, to aid in aggressive thermal management.

Given the data-intensive nature of CNN algorithms (ML performance and power is dominated by data movement, not compute), several implementations have looked at accelerating the memory subsystem. Recent works like [49], [9], [68] have proposed CNN accelerator implementation in the logic layer of Hybrid Memory Cube (HMC). Here, in order to alleviate the bandwidth pressure on the data-path between the processor chip and the main memory chip, and to get rid of the large

on-chip local memory that occupy more than 50% of the chip [24], an array of processing elements and register files (as and where needed) are incorporated in the logic layer of the 3D-stacked DRAM module. The authors of [9] use HMC as a co-processor for CNN acceleration through synchronization free parallelism while the authors of [68] embed specialized state-machines within the vault controllers of HMC to drive data into the processing elements in the logic layer. Some accelerators use strategies such as optimized memory use and the use of lower precision arithmetic to accelerate calculations and increase throughput of computation, however, they tend to be designed for specific use cases and markets. Most of the accelerators support traditional clock and power gating; some of them support DFS / DVS / DVFS, making them amenable to standard energy efficiency algorithms through hardware software orchestration.

The data-intensive nature of CNN algorithms is in contrast with von Neumann execution models and this has motivated non-von Neumann models of computation like dataflow, spiking neural networks, and other forms of brain-inspired computing. The authors of [24] propose an optimized algorithmic dataflow for CNNs by exploiting local data reuse and optimization of intermediate data movement. The proposed design in [49] uses the dataflow model of [24] along with scheduling and partitioning in software to implement CNN acceleration in HMC. In [45], the authors present a compiler that transforms high level dataflow graphs into machine code representations. Another work [72] 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 [87] presents 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. Energy efficiency of deep learning accelerators is covered in more detail in [48].

4 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 microarchitectural level, techniques described in Section 3 are used and are specified using proprietary methods. At the hardware-firmware-OS level, a different 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 levels are now described and are illustrated in Figure 6 (the different colours are to delineate different layers and components).

4.1 IEEE 1801: Unified Power Format

The microarchitectural techniques for energy efficiency translate to hardware through a some important concepts at the RTL or lower levels:

(1) **Power domains**: These are independently powered domains, enabling the application of different power reduction techniques in each domain.
(2) **State retention**: It is important to save essential state when power is off, and to restore it when the power is turned back on. For this, special state-retention elements can be added to

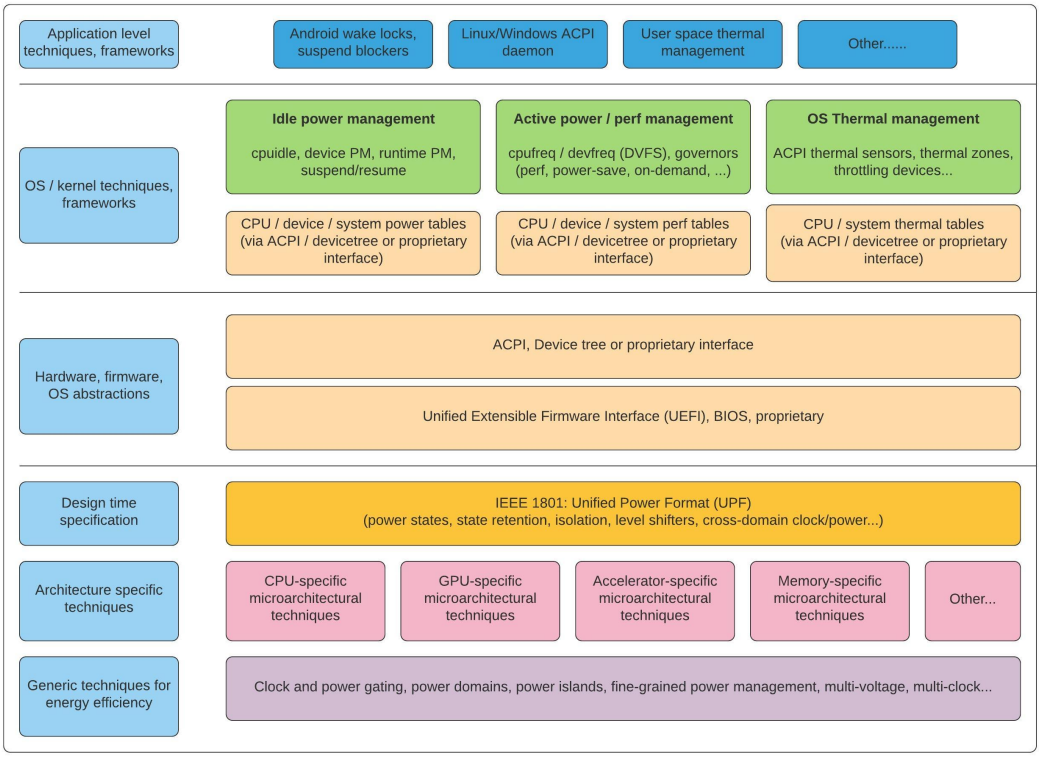

Fig. 6. Specifications and abstractions at different levels

keep a minimal amount of power available to registers whose contents must be preserved during power shutdowns.

(3) **Isolation**: This is to ensure correct logical and electrical interactions between domains belonging to different power states. To do this, a tool can insert isolation cells on signals coming from regions that are turned off.
(4) **Legal power states**: Only legal power state transitions must be allowed across components.
(5) **Level shifters**: To ensure communication between domains powered by different voltage levels, level shifters are added to signals crossing between regions with different voltages and different switching thresholds.

Across all these techniques, it is crucial to have a common, unambiguous representation of low power design intent across designers, verification engineers, design and verification tools.

IEEE 1801 Standard for the Design and Verification of Low Power Integrated Circuits, also called the Unified Power Format (UPF), is a standard for specifying the power intent and low power methods in early phases of design. UPF allows for specifying hardware systems with power as a key consideration and UPF scripts help describe power intent, or power management constructs / features. For example - which power rails are to be routed to individual blocks, when are blocks expected to be powered up or shut down, how voltage levels should be shifted between two different power domains and the type of measures taken for retention registers if the primary power supply to a domain is removed. Additionally, specifying power features in a standard format allows for several design and verification tools to validate the complex design. Beyond the obvious importance

of using standardized formats across all phases of design, the other importance of using UPF arises from the fact that often large blocks of hardware IP are re-used either in different systems-on-chip designs or several different generations of a particular system or even for porting a proven system to a different target technology. This is, therefore, a particularly important problem for hardware IP suppliers who need to be able to supply descriptions of power intent for products to their customers without having any information about what implementation-specific decisions might be taken by the customer, or how their IP is integrated into a different hardware / SOC design.

The latest standard, UPF 3.0, released in 2016, has improved capabilities for adding bottom-up implementation flow, power models, and high-level power analysis. The ability to develop energy-efficient platforms, including the hardware, software and system power management components of the platform, requires the ability to use appropriate levels of design abstraction for the task at hand. With UPF 3.0, architects can now model the salient power related characteristics of a piece of IP for use at the system level, thereby providing a foundation for building complex system level power models in a standardized manner. Using UPF-based hardware designs as reusable components in other SOCs is an important area for power/performance projections using power models of individual hardware comopnents leading up to system level power models. This is an important area of cross industry collaboration and standardization in the IEEE P2416 [8] working group.

4.2 UEFI, ACPI and DeviceTree

UPF is a design time specification for low power and it is disconnected from runtime management by system software. Over the years, several proprietary and industry consortiums have attempted to define abstractions for runtime management, which we will now describe.

4.2.1 Unified Extensible Firmware Interface (UEFI). UEFI [108] provides a well understood standard interface between low level hardware, firmware and operating system. The interface consists of data tables that contain platform-related information, boot and runtime service calls that are available to the operating system and its loader. Together, these provide a standard environment for booting an operating system and running pre-boot applications. The latest version of UEFI also adds support for REST APIs, thereby providing interoperability between computer systems on the internet. EFI has a specification, an open source BSD-licensed implementation, and the mainline project has both x86 and ARM support (and now RISC-V as well). UEFI is a generic hardware-firmware-OS interface and is not specifically related to energy efficiency.

4.2.2 Advanced Configuration and Power Interface (ACPI). ACPI [109] is a standard for runtime management of hardware. The scope of ACPI comprises system run-time configuration, power and thermal management as well as hardware error handling support. ACPI is, essentially, a standardized way to enable the operating system to discover, configure and initialize the system's hardware. It provides runtime tables for power management (among other things) - power states supported by the CPU(s), CPU hierarchy, DVFS states supported and associated transition latencies, thermal sensors supported on the platform, thermal states supported and thermal throttling order. The important thing to note is that UEFI is not tied to ACPI and will work with any firmware description. Similarly, ACPI does not depend on UEFI, and can work with any other low level device initialization framework as well such as U-Boot or BIOS. ACPI is a very active industry working group and is constantly being updated. Recent trends with GPUs and accelerators are not yet handled in ACPI.

4.2.3 Device Tree (DT). While ACPI was historically created for x86 platforms, the ARM ecosystem developed Device Tree to describe the same information for ARM-based devices. Thus, ACPI and

Table 3. Summary of Modeling and Simulation tools

Domain	Key work, surveys or books
Processor and multiprocessor simulators	gem5 [17], Multi2sim [107], MARSSx86[90], PTLsim [122] and ZSim [97], Surveys [3], [4], Book [39]
Cache Simulators	gem5 [17], CACTI [100], Survey [19]
Memory Simulators	Analytical modeling [2], Trace driven simulation [111]
GPU Simulators	[20]
Accelerator Simulators	Alladin [99], Minerva [95], FireSim [65], Survey [3]
SOC and full system simulators	PARADE [26], gem5 [17], McPAT [73], SoftSDV [110]
Power and Energy Simulators	Wattch [22], SimplePower [117], IBM PowerTimer tool [21], McPAT [73], PowerAnalyzer [84], Survey [6]
Thermal Simulators	Book [67], TEMPEST [37], Hotspot [101], SESCTherm [86], Power Blurring [124], Intel Docea [31], Survey [103]

DT overlap in that they both provide mechanisms for enumerating devices, attaching additional configuration data to devices (which can be used by higher layers of software). Rafael [94] goes into details of the commonalities between ACPI and Device Tree and the convergence between the two standards.

5 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. While initial generation of tools catered to building functional, timing/cycle-accurate models for performance estimation, subsequent tools incorporated power, energy and thermal modeling, simulation and estimation/projections and also the ability to run real, or close-to real workloads as well as full operating systems. Some key modeling/simulation tools across different kinds of hardware are illustrated in Table 3. 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.

5.1 Power and Energy Modeling / Simulation

Providing accurate power and performance estimations of future architectures is crucial for system architects to do what-if analysis of possible design tradeoffs. Wattch [22] was one of the first tools to provide accurate power estimation of processors. It developed a framework for analyzing and optimizing microprocessor power dissipation at the architecture-level thereby allowing architects to make high-level analysis of power tradeoffs. SimplePower [117] was introduced as a means of doing detailed whole processor analysis of dynamic power. It focused on in-order five-stage pipelines, with detailed models of integer ALU power as well as other regions of the chip. The Wattch tool built on cache modeling from Cacti [100], and provided parameterized activity factor-based estimates as well.

Both SimplePower [117] and Wattch [22] were both based on analytic power modeling techniques. The IBM PowerTimer tool [21] provides a processor simulator based on empirical techniques âĂŤ one can estimate the power consumption of a particular architectural module by using the measured power consumption in an existing reference processor, and applying appropriate scaling techniques for design and process technology. This tool thereby allows architects to estimate power of future generation designs early in the design phase. McPAT [73] can simulate timing, area and power of multicore processors. PowerAnalyzer [84] is a power evaluation tool suitable for calculating power consumption for complete computer systems. Power consumption of FPGAs is also an important area, hence modeling the power consumption of FPGA-based systems has also gained importance in recent years. In [6], Anderson et al. provide a survey of power estimation techniques for FPGAs. The authors formulate empirical prediction models for net activity for FPGAs.

5.2 Thermal Modeling

The ability to model thermal behavior is important especially for small form factor devices like smartphones and handhelds where the heat flows are critical in determining the usage of the device (and restrictions therein). Thermal modeling is also heavily used in large server farms and data centers to be able to administratively monitor and manage load across servers. Thermal modeling has several aspects ranging from designing thermals for a microprocessor alone to provisioning thermal sensors, and cooling of larger systems or data centers. In the past, the focus was on CPU thermal modeling, estimation and analysis; the focus has now moved to platform level thermal modeling, estimation and control mechanisms. Kaxiras and Martonosi [67] describe in detail the relationship between power and temperature and show the exponential dependence of power on temperature and the cyclic relationship âĂŤ thermals depend on power dissipation and density; on the other hand, power also depends on temperature.

TEMPEST [37] was one of the first thermal models, where temperature was modeled based on power dissipation and density values. It is a flexible, cycle-accurate microarchitectural power and performance analysis tool based on SimpleScalar [23]. The simulator generates power estimates based on either empirical data or analytical models and supports dynamic and leakage power and process technology scaling options as well as effects of clock throttling. The main drawback was that it modeled only the CPU, but not other regions or other architectural units. Skadron et al. [101] proposed and validated the HotSpot approach, a compact RC model for localized heating in high-end microprocessors. This was a complex model that considered both the lateral relationships between units on chip, as well as the vertical heating/cooling relationships between the active portion of the silicon die and the attached heat spreader and heat sink layers that seek to even out temperature and draw heat away from the active silicon. In recent SoCs, thermal modeling has taken up even higher prominence given that some of these smaller devices have no active cooling mechanisms like fans. Platform architects build hardware prototypes with heat generators that are modeled on actual physical components, and then test the prototypes in thermal chambers to analyze heat flow. SESCTherm [86] is a novel temperature modeling infrastructure that offers accurate thermal characterization. This framework is based on finite difference methods and equations. Power Blurring [124] is another temperature calculating model, which is developed based on a matrix convolution approach to reduce computation time as compared to the finite difference method. Power blurring (PB) uses a technique analogous to image blurring for calculating temperature distributions. Sarangi et. al [103] presents one of the most comprehensive and updated surveys of thermal estimation and modeling tools. The semiconductor industry has also developed several comprehensive thermal modeling and estimation tools. While many of these tend to be proprietary, some like Intel Docea [31] tool is available for experimental evaluation. Thermal simulation algorithms for calculating the on-chip temperature distribution in a multilayered substrate

structure rely on Green's function and discrete cosine transforms (DCT). In [113], the authors present NanoTherm, a solution to compute Green's function using a fast analytical approach that exploits the symmetry in the thermal distribution. Additionally, conventional methods fail to hold at the nanometer level, where it is necessary to solve the Boltzmann transport equation (BTE) to account for quantum mechanical effects, without which, there can be errors in temperature calculation of upto 60%. NanoTherm also provides a fast analytical approach to solve the BTE for nanometer chip designs.

5.3 Accelerator Simulators

With the rise of domain-specific accelerators, the need for power and performance modeling of such chips has become an important area of research. Accelerators could be GPUs, application specific integrated circuits (ASICs), digital signal processors (DSP), field programmable gate arrays (FPGA), near-data and in-memory processing engine, or any other similar component optimized for fixed functions. It is largely expected that future architectures will be a mix of CPUs, GPUs and domain-specific accelerators, each optimized for a specific function.

Alladin [99] is a pre-RTL power and performance modeling framework for accelerators. The framework takes high-level language descriptions of algorithms as inputs, and uses dynamic data dependence graphs (DDDG) as a representation of an accelerator without having to generate RTL. Starting with an unconstrained program DDDG, which corresponds to an initial representation of accelerator hardware, Aladdin applies optimizations as well as constraints to the graph to create a realistic model of accelerator activity and then overlays power and performance estimation. To accurately model the power of accelerators, Aladdin uses precise activity factors, accurate power characterization of different DDDG components, characterizes switching, internal, and leakage power from design compilers for each type of DDDG node (multipliers, adders, shifters) and registers. Minerva [95] is a highly automated co-design approach across the algorithm, architecture, and circuit levels to optimize DNN hardware accelerators. It allows for the modeling and simulation of ultra-low power DNN accelerators (in the range of tens of milliwatts), making it feasible to deploy DNNs in power-constrained IoT and mobile devices. FireSim [65] is an open-source simulation platform that enables cycle-exact microarchitectural simulation of large scale-out clusters by combining FPGA-accelerated simulation of silicon-proven RTL designs with a scalable, distributed network simulation. Unlike prior FPGA-accelerated simulation tools, FireSim runs on Amazon EC2 F1, a public cloud FPGA platform, which greatly improves usability, provides elasticity, and lowers the cost of large-scale FPGA-based experiments. The motivation for FireSim arises from recent trends in computer architecture that push the boundaries of hardware-software co-design at-scale. The authors of FireSim state that the platform is sufficiently mature to reproduce warehouse-scale workload performance phenomena. More accelerator simulators are described in detail in [3].

5.4 SOC and full system simulators

Largely, accelerators are integrated with processors on the same chip or on a system-on-chip (SoC). The simulation of accelerators in addition to processors thus aim to give a complete view of the performance of benchmarks on such full systems. PARADE [26] was the first cycle-accurate full-system simulation platform that simulates the whole system of the accelerator-rich architecture accurately, including X86 out-of-order cores, dedicated or composable accelerators, global accelerator manager, coherent cache/scratchpad with shared memory, and network-on-chip. It achieves cycle-accuracy by leveraging the existing cycle accurate gem5 simulator [17] for the CPU and cache memory hierarchy, and high-level synthesis (HLS) and register transfer level (RTL) simulation for the accelerator. In addition to performance simulation, PARADE also models the power, energy and area using existing toolchains including McPAT [73] for the CPU and HLS and RTL tools for the

accelerator. SoftSDV [110] is a presilicon software development environment that has been used widely at Intel to enable several generations of commercial operating systems and applications on new x86-based client and server processors/chips. Tools like SoftSDV are important for commercial OSes and applications to be enabled in a pre-silicon version of the upcoming hardware system, which would future enable downstream power, performance optimizations and fine tuning once the silicon is ready.

6 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 Intel (or x86 in general) 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.

6.1 ARM System Architecture and Energy Efficiency Features

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.

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.

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.

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.

6.2 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.
- *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. Intel SoCs typically support the following system states:

(1) S0i1: Shallow idle state for the entire SOC
(2) S0i1-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.
(3) S0i1-Audio: SOC in low power state except audio block.
(4) S0i1-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, with only critical wake sequencing supported.

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.

6.3 OS and Software Techniques

Linux [75] 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 - (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. There are several different governors - performance, user-mode, power-save, on-demand, p-state, interactive, and several others.
(3) **CPU Idle**: This is a Linux kernel subsystem that manages the CPU when it is idle. 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. 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. It is mostly used to enable/disable a regulator output or control the output voltage and or current. Regulating power output saves power and prolongs battery life. 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 used 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 for handling DVFS of non-CPU devices such as GPU, memory and accelerator subsystems. Devfreq is similar to cpufreq but cpufreq does not allow multiple device registration and is not suitable for heterogeneous devices with different governors. However, the usage of devfreq across GPUs or accelerators is not common yet.

(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). 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) **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.

6.4 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 [81]. 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) **Software prefetching and DVFS**: Software prefetching primarily aims to improve performance by overlapping the computing and memory access latencies. The idea is to insert prefetch instructions into the program so that data is fetched into registers or caches well before time, and processor stall on memory access instructions is avoided. Since prefetching increases the number of instructions, it also increases the power consumption, and hence, it must be balanced with suitable performance enhancement.
(4) **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 [30], 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.

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. Power management brings a host of new types of bugs which are not in the class of traditional functional bugs. Table 4 shows the different classes of bugs and the new verification techniques required, some of which are hard to verify in pre-silicon (for example, voltage sequencing, due to lack of integrated power delivery models into SOC emulation models) or thermal runways (usually these are usually verified on form factor devices in thermal chambers that simulate different thermal conditions). At a high level, verification can be done at either the gate level, RTL/architectural level or at SOC/system level, as described in detail in [118] and [85]; here we focus on verifying system level energy efficiency.

Industrial designs rely heavily on ensuring that once the silicon arrives, power management can be validated as soon as possible, and thermal solutions can be built accurately for the specific form factors in consideration. In order to accomplish this, companies typically use FPGAs to emulate the SoC RTL, and build platform level validation/verification tools that can include the ability to boot entire operating system on such FPGA systems. In [64], the authors present a good overview of the

Table 4. Summary of Power Related Bugs

Power Related Issue	Verification techniques required
Isolation/level shifting bugs	Verify connection, placement, isolation/level shifting
Control sequencing bugs	Include power intent files like UPF
Electrical problems like memory corruption	Reach good power state coverage
Power/voltage sequencing bugs	Verify FW/SW control sequences
Power gating collapse/dysfunction, Clock domain/crossover bugs	Verification at each stage of design, not just RTL; verify netlist at each handoff, power switch/rail connectivity
Power-on/reset bugs	Wide coverage of test cases across power-on/reset flows
Thermal runways/cooling inefficiencies	Verify thermal conditions, thermal modeling for different form factors/designs
Bugs due to concurrent access from multiple IPs during end-to-end use cases	Verify end to end system level power sequences, including FW, SW, drivers to uncover race conditions

different techniques used in system level low power verification, the importance of using power intent specifications like UPF and simulation tools/methodologies that can accurately model power states/sequences. The authors in [80] describe System-C based virtual prototyping techniques to perform power intent/sequence validation, and also propose using system level low power abstractions as possible extensions to UPF. This includes abstract definition of voltage relationships and dynamic aspects such as operating conditions. In [85], the authors talk about HW-SW co-design and verifying energy efficiency features in pre-silicon, and the need for simulating end-to-end use cases in such verification methodologies. Targeted verification of each IP block, including CPU cores, GPUs, memory, and others can be done using traditional silicon verification techniques through a combination of random, targeted and functional PM tests. Since SOCs typically integrate third party IP blocks, specific PM related tests are needed for such IPs. Beyond the IPs, and going into the system level, a combination of different platforms and environments are used for different aspects of pre-silicon verification. These include Virtual Platforms (VP, where an entire OS can be booted quickly on a simulated system model), FPGAs (for specific hardware), Hybrid Virtual platforms (VP plus FPGA), System Level Emulation (SLE) platforms that is a complex FPGA that simulates parts of the chip or the entire chip. Each environment is best suited for a specific set/category of pre-silicon verification. Some of them can support production OS boot in reasonable times for SW development/co-design/debug. For thermal validation, different form factor devices are built early on and are analyzed in heat chambers. Based on the thermal hot spots, appropriate thermal control algorithms are defined and fine tuned. This is a costly, but accurate way of ensuring that thermal management on the devices are validated effectively. Usually, a multi-pronged strategy is used that could be a combination of all or some of these environments and techniques.

8 ENERGY EFFICIENCY STANDARDS, CROSS LAYER ENERGY EFFICIENCY

In this section, we will discuss important industry consortiums, standards and regulations for energy efficient and sustainable computing. Some of the recent and key initiatives across research and the industry are:

(1) The Green Grid [51] is a global consortium dedicated to advancing energy efficiency in data centers founded by many companies like AMD, Dell, HP, IBM, Intel, VMware and many others.
(2) The Green500 [1] list rates supercomputers by energy efficiency, encouraging a focus on efficiency (megaflops/watt) rather than absolute performance.
(3) The Transaction Processing Performance Council (TPC) Energy specification [15] augments existing TPC benchmarks with energy metrics. The metric is calculated as the ratio of the energy consumed by all components of the benchmark system (typically measured in watts-seconds) to the total work completed (typically measured as a number of transactions).
(4) SPECpower [14] is perhaps the first industry standard benchmark that measures power consumption in relation to performance for server-class computers. The workload exercises the CPUs, caches, memory hierarchy and the scalability of shared memory processors (SMPs) as well as the implementations of the JVM (Java Virtual Machine), JIT (Just-In-Time) compiler, garbage collection, threads and some aspects of the operating system. Other benchmarks which measure energy efficiency include SPECweb, SPECvirt, and VMmark and EEMBC's ULPMark [13].
(5) The Energy Star [102] program sets regulations around energy efficiency requirements for computer equipment, along with a tiered ranking system for approved products. It is run by the U.S. Environmental Protection Agency and U.S. Department of Energy to promote energy efficiency across all categories of computing and electronic systems using different standardized methods.

8.1 Energy Efficient HPC and the Power API

The Energy Efficient HPC (EEHPC) [52] is a group that is focused on driving implementation of energy conservation measures and energy efficient design of HPC systems. The working groups cover several aspects of EE HPC systems - infrastructure, cooling, efficient power sources, systems architecture, energy aware job scheduling, specifications (Power API) and benchmarks. The key motivation for **Power API** is that achieving practical exascale computing will require massive increases in energy efficiency across hardware and software. With every generation of new hardware, more power measurement and control capabilities are exposed, with in-chip monitoring rapidly increasing as there are more sensors to track process, voltage, and temperature across the die [40]. EEHPC's Power API is a portable API for power measurement and control; it provides multiple levels of abstractions, and allows algorithm designers to add power and energy efficiency to their optimization criteria at the system level like energy-aware scheduling.

8.2 Geo PM

The Global Extensible Open Power Manager (GEOPM) [38] is an open source runtime framework with an extensible architecture enabling new energy management strategies in HPC systems. Different plugins can be tailored to the specific performance or energy efficiency priorities of each HPC center. It can be used to dynamically coordinate hardware settings across all compute nodes used by an application in response to the application's behavior and requests from the resource manager. The dynamic coordination is implemented as a hierarchical control system for scalable communication and decentralized control. The hierarchical control system can optimize for various

objective functions including maximizing global application performance within a power bound or minimizing energy consumption.

8.3 Energy Standards: California Energy Commission (CEC)

The California Energy Commission's [25] goal is to lead the state to a 100 percent clean energy future. As the state's primary energy policy and planning agency, the Energy Commission plays a critical role in creating the energy system of the future. CEC has been driving some of the most stringent energy regulatory standards for computing systems and other electronic appliances via Energy Star and related programs that have now been adopted in different countries around the world.

8.4 IEEE P2416 Standard for Power Modeling of Electronic Systems

IEEE P2416 [8] defines a framework for the development of parameterized, accurate, efficient, and complete power models for hardware IP blocks and the entire system that can be used for power modeling and analysis. It is based on process, voltage, and temperature (PVT) independence and defines power and thermal management interfaces for hardware models and also workload and architecture parameterization. Such models are suitable for use in software development and hardware design flows, as well as for representing both pre-silicon estimates and post-silicon data. The working group recently released a version of this standard [59].

8.5 IEEE P2415 Unified HW Abstraction and Layer For Energy Proportional Systems

IEEE P2415 standard [7] intends to define the syntax and semantics for energy oriented description of hardware, software and is expected to be compatible with the IEEE 1801 (UPF) and IEEE P2416 standards to support an integrated flow across architecture, design, estimation and system software. The standard complements functional models in VHDL/Verilog/SystemVerilog/ SystemC by providing an abstraction of the design hierarchy and the design behavior with regard to power/energy usage in order to fill a key gap - current IEEE P1801 (UPF) is focused on the voltage distribution structure in design at RTL and below, has minimal abstraction for time, but depends on other hardware oriented standards to abstract events, scenarios, clock or power trees that are required for energy proportional design, verification, modeling and management of electronic systems.

9 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 3 nm will provide a few generations of performance gains and energy efficiency, but perhaps requiring new transistor architectures like nanosheets and nanowires beyond today's FinFETs.
(2) **Exascale and beyond**: Research and industry will continue the push to build exascale systems using new architectures [18] and computing paradigms like mixing von Neumann and non-von Neumann models [32].

(3) **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 [89]. 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.
(4) **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.
(5) **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 [106] is an important emerging area of machine learning under the 1mW power envelope. Similarly, software-defined hardware [41] is an important area of reconfigurable systems.
(6) **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.
(7) **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.
(8) **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) [104], which is a tightly integrated, domain-specific system for autonomous driving with a TDP of under 40W.
(9) **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* [28] 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.

Systems are thus becoming complex to design and optimize, requiring a tight interplay between energy efficient hardware and energy-aware software across various domains.

10 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 the limits of chip specialization will continue to impose constraints. 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. 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 standards 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.

REFERENCES

[1] The Green 500. 2020. https://www.top500.org/green500/.

[2] A. Agarwal, J. Hennessy, and M. Horowitz. 1989. An Analytical Cache Model. *ACM Trans. Comput. Syst.* 7, 2 (May 1989), 184âĂŞ215.

[3] Ayaz Akram and Lina Sawalha. 2016. A Comparison of x86 Computer Architecture Simulators, Computer Architecture and Systems Research Laboratory (CASRL) Technical Report. https://scholarworks.wmich.edu/casrl_reports/1.

[4] A. Akram and L. Sawalha. 2019. A Study of Performance and Power Consumption Differences Among Different ISAs. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*. 628–632.

[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] Jason H. Anderson and Farid N. Najm. 2004. Power Estimation Techniques for FPGAs. *IEEE Trans. Very Large Scale Integr. Syst.* 12, 10 (Oct. 2004), 1015âĂŞ1027.

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

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

[9] Erfan Azarkhish, Davide Rossi, Igor Loi, and Luca Benini. 2018. Neurostream: Scalable and energy efficient deep learning with smart memory cubes. *IEEE Transactions on Parallel and Distributed Systems* 29, 2 (2018), 420–434.

[10] Rajeev Balasubramonian, Jichuan Chang, Troy Manning, Jaime H Moreno, Richard Murphy, Ravi Nair, and Steven Swanson. 2014. Near-data processing: Insights from a micro-46 workshop. *IEEE Micro* 34, 4 (2014), 36–42.

[11] 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.

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

[13] The EEMBC ULPMark Energy Benchmark. 2019. https://www.eembc.org/ulpmark/.

[14] The SPEC Power Benchmark. 2019. http://www.spec.org/power_ssj2008/.

[15] The TPC Energy Benchmark. 2019. http://www.tpc.org/tpc_energy/.

[16] 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).

[17] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, and et al. 2011. The Gem5 Simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011), 1âĂŞ7.

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

[19] Hadi Brais, Rajshekar Kalayappan, and Preeti Ranjan Panda. 2020. A Survey of Cache Simulators. *ACM Comput. Surv.* 53, 1, Article Article 19 (Feb. 2020), 32 pages.

[20] Robert A. Bridges, Neena Imam, and Tiffany M. Mintz. 2016. Understanding GPU Power: A Survey of Profiling, Modeling, and Simulation Methods. *ACM Comput. Surv.* 49, 3, Article Article 41 (Sept. 2016), 27 pages.

[21] D. Brooks, P. Bose, V. Srinivasan, M. K. Gschwind, P. G. Emma, and M. G. Rosenfield. 2003. New Methodology for Early-Stage, Microarchitecture-Level Power-Performance Analysis of Microprocessors. *IBM J. Res. Dev.* 47, 5âĂŞ6 (Sept. 2003), 653âĂŞ670.

[22] David Brooks, Vivek Tiwari, and Margaret Martonosi. 2000. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA âĂŹ00)*. Association for Computing Machinery, New York, NY, USA, 83âĂŞ94.

[23] Doug Burger, Todd Austin, and Stephen Keckler. 2004. Recent extensions to the SimpleScalar tool suite. *SIGMETRICS Performance Evaluation Review* 31 (03 2004), 4–7.
[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] California Energy Commission. 2020. https://www.energy.ca.gov/.
[26] Jason Cong, Zhenman Fang, Michael Gill, and Glenn Reinman. 2015. PARADE: A Cycle-Accurate Full-System Simulation Platform for Accelerator-Rich Architectural Design and Exploration. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD âĂŹ15)*. IEEE Press, 380âĂŞ387.
[27] Compute Express Link Consortium. 2020. Compute Express Link. (2020). https://www.computeexpresslink.org/
[28] 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).
[29] Intel Corporation. 2017. Intel® Movidius™ Myriad™ VPU 2: A Class-Defining Processor. https://www.movidius.com/myriad2.
[30] 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/.
[31] Intel Corporation. 2018. Intel Power and Thermal Modeling Simulation Suite. https://www.intel.com/content/www/us/en/system-modeling-and-simulation/docea/overview.html.
[32] 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/.
[33] Intel Corporation. 2019. Lakefield: Hybrid CPU with Foveros Technology. https://newsroom.intel.com/press-kits/lakefield/.
[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] R.H. Dennard, F.H. Gaensslen, V.L. Rideout, E. Bassous, and A.R. LeBlanc. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *Solid-State Circuits, IEEE Journal of* 9, 5 (Oct. 1974), 256–268. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=1050511&isnumber=22538
[37] Ashutosh Dhodapkar, Chee How Lim, George Cai, and W. Robert Daasch. 2000. TEM2P2EST: A Thermal Enabled Multi-Model Power/Performance ESTimator. In *Proceedings of the First International Workshop on Power-Aware Computer Systems-Revised Papers (PACS âĂŹ00)*. Springer-Verlag, Berlin, Heidelberg, 112âĂŞ125.
[38] 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.
[39] Lieven Eeckhout. 2010. *Computer Architecture Performance Evaluation Methods* (1st ed.). Morgan and Claypool Publishers.
[40] Semiconductor Engineering. 2019. In Chip Monitoring Becoming Essential Below 10nm. https://semiengineering.com/in-chip-monitoring-becoming-essential-below-10nm/.
[41] Semiconductor Engineering. 2020. Software Defined Hardware gains ground again. https://semiengineering.com/software-defined-hardware-gains-ground-again/.
[42] 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).
[43] Ovtcharov et al. [n.d.]. Accelerating Deep Convolutional Neural Networks Using Specialized Hardware, Microsoft Research. https://www.microsoft.com/en-us/research/publication/accelerating-deep-convolutional-neural-networks-using-specialized-hardware/.
[44] 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.
[45] 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.
[46] Richard Phillips Feynman, Anthony J. Hey, and Robin W. Allen. 2000. *Feynman Lectures on Computation*. Perseus Books, USA.

[47] 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.

[48] 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.

[49] 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.

[50] 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 Article 104 (Dec. 2018), 33 pages.

[51] The Green Grid. 2020. https://www.thegreengrid.org/.

[52] The Energy Efficient High Performance Computing (EEHPC) Group. 2019. https://eehpcwg.llnl.gov/index.html.

[53] John Gurd, Wim Bohm, and Yong Meng Teo. 1987. Performance issues in dataflow machines. *Future Generation Computer Systems* 3, 4 (1987), 285–297.

[54] Laszlo Gyongyosi and Sandor Imre. 2019. A Survey on quantum computing technology. *Computer Science Review* 31 (02 2019), 51–71.

[55] 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. *arXiv preprint arXiv:1706.02725* (2017).

[56] 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.

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

[58] M.D. Hill and M.R. Marty. 2008. Amdahl's Law in the Multicore Era. *Computer* 41, 7 (july 2008), 33–38. http://ieeexplore.ieee.org/search/srchabstract.jsp?tp=&arnumber=4563876&queryText%3DReevaluating+Amdahl%E2%80%99s+law+in+the+multicore+era%26openedRefinements%3D*%26searchField%3DSearch+All

[59] IEEE. 2019. IEEE Approves New Power Modeling Standard. https://www.businesswire.com/news/home/20190627005205/en/IEEE-Approves-New-Power-Modeling-Standard.

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

[61] 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/.

[62] Norman P. Jouppi, Cliff Young, Nishant Patil, and David Patterson. 2018. A Domain-Specific Architecture for Deep Neural Networks. *Commun. ACM* 61, 9 (Aug. 2018), 50âĂŞ59.

[63] 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.

[64] B. Kapoor, S. Hemmady, S. Verma, K. Roy, and M. A. D'Abreu. 2009. Impact of SoC power management techniques on verification and testing. In *2009 10th International Symposium on Quality Electronic Design*. 692–695.

[65] Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, and et al. 2018. Firesim: FPGA-Accelerated Cycle-Exact Scale-out System Simulation in the Public Cloud. In *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA âĂŹ18)*. IEEE Press, 29âĂŞ42.

[66] 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 (DAC âĂŹ12)*. Association for Computing Machinery, New York, NY, USA, 1153âĂŞ1158.

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

[68] Duckhwan Kim, Jaeha Kung, Sek Chai, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. 2016. Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 380–392.

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

[70] 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.

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

[72] 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.

[73] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42)*. Association for Computing Machinery, New York, NY, USA, 469âĂŞ480.

[74] Sung Kyu Lim. 2010. 3D circuit design with through-silicon-via: Challenges and opportunities. In *IEEE Electronic Design Processes Symposium Workshop*.

[75] Linux. 2020. Linux Kernel Documentation. https://www.kernel.org/doc/html/latest/.

[76] Christianto C Liu, Ilya Ganusov, Martin Burtscher, and Sandip Tiwari. 2005. Bridging the processor-memory performance gap with 3D IC technology. *IEEE Design & Test of Computers* 22, 6 (2005), 556–564.

[77] Igor L. Markov. 2014. Limits on fundamental limits to computation. *Nature* 512, 7513 (Aug 2014), 147âĂŞ154.

[78] 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.

[79] Sally A McKee. 2004. Reflections on the memory wall. In *Proceedings of the 1st conference on Computing frontiers*. ACM, 162.

[80] F. Mischkalla and W. Mueller. 2014. Advanced SoC virtual prototyping for system-level power planning and validation. In *2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 1–8.

[81] Sparsh Mittal and Jeffrey S. Vetter. 2014. A Survey of Methods for Analyzing and Improving GPU Energy Efficiency. *ACM Comput. Surv.* 47, 2, Article Article 19 (Aug. 2014), 23 pages.

[82] Gordon E. Moore. 1965. Cramming more components onto integrated circuits. *Electronics* 38, 8 (April 1965).

[83] 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.

[84] Trevor Mudge, Nam Kim, Jeffrey Ringenberg, and Taeho Kgil. 2004. Power Analyzer for Pocket Computing (PAPC). (01 2004), 58.

[85] Rajeev Muralidhar, Nivedha Krishnakumar, Bryan Morgan, and Neil Rosenberg. 2017. Pre-Silicon Power Management Verification of Complex SOCs: Experiences with IntelÂľ Moorefield.

[86] Joseph Nayfach-Battilana and Jose Renau. 2009. SOI, Interconnect, Package, and Mainboard Thermal Characterization. In *Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED âĂŹ09)*. Association for Computing Machinery, New York, NY, USA, 327âĂŞ330.

[87] Chris Nicol. 2017. A coarse grain reconfigurable array (cgra) for statically scheduled data flow computing. *Wave Computing White Paper* (2017).

[88] 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 (ISCA '15)*. ACM, New York, NY, USA, 298–310.

[89] 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.

[90] A. Patel, F. Afram, S. Chen, and K. Ghose. 2011. MARSS: A full system simulator for multicore x86 CPUs. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1050–1055.

[91] David Patterson and Andrew Waterman. 2017. *The RISC-V Reader: An Open Architecture Atlas*. Strawberry Canyon.

[92] J Thomas Pawlowski. 2011. Hybrid memory cube (HMC). In *Hot Chips 23 Symposium (HCS), 2011 IEEE*. IEEE, 1–24.

[93] Milan Radulovic, Darko Zivanovic, Daniel Ruiz, Bronis R de Supinski, Sally A McKee, Petar Radojković, and Eduard Ayguadé. 2015. Another trip to the wall: How much will stacked dram benefit hpc?. In *Proceedings of the 2015 International Symposium on Memory Systems*. ACM, 31–36.

[94] Intel Corporation Rafael Wysocki. 2015. ACPI vs Device Tree. https://elinux.org/images/f/f8/ACPI_vs_DT.pdf.

[95] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. 2016. Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA âĂŹ16)*. IEEE Press, 267âĂŞ278.

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

[97] Daniel Sanchez and Christos Kozyrakis. 2013. ZSim: fast and accurate microarchitectural simulation of thousand-core systems. *ACM SIGARCH Computer Architecture News* 41 (07 2013), 475.

[98] 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

[99] 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.
[100] Premkishore Shivakumar and Norman Jouppi. 2001. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. (01 2001).
[101] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, Wei Huang, Sivakumar Velusamy, and David Tarjan. 2004. Temperature-Aware Microarchitecture: Modeling and Implementation. *ACM Trans. Archit. Code Optim.* 1, 1 (March 2004), 94âĂŞ125. https://doi.org/10.1145/980152.980157
[102] Energy Star. 2019. https://www.energystar.gov/.
[103] Hameedah Sultan, Anjali Chauhan, and Smruti R. Sarangi. 2019. A Survey of Chip-Level Thermal Simulators. *ACM Comput. Surv.* 52, 2, Article Article 42 (April 2019), 35 pages.
[104] 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.
[105] 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/.
[106] EE Times. 2020. AI at the very very Edge. https://www.eetimes.com/ai-at-the-very-very-edge/.
[107] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez. 2007. Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors. In *19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07)*. 62–68.
[108] UEFI. 2019. Unified Extensible Firmware Interface Forum(UEFI). https://uefi.org/.
[109] UEFI-ACPI. 2019. Advanced Configuration and Power Interface (ACPI). https://uefi.org/acpi/.
[110] Richard Uhlig and Intel Corp. 2001. SoftSDV: A Presilicon Software Development Environment for the IA-64 Architecture. (09 2001).
[111] Richard A. Uhlig and Trevor N. Mudge. 2000. *Trace-Driven Memory Simulation: A Survey.* Springer Berlin Heidelberg, Berlin, Heidelberg, 97–139.
[112] Keshavan Varadarajan, S. K. Nandy, Vishal Sharda, Amrutur Bharadwaj, Ravi Iyer, Srihari Makineni, and Donald Newell. 2006. Molecular Caches: A Caching Structure for Dynamic Creation of Application-Specific Heterogeneous Cache Regions. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 39)*. IEEE Computer Society, USA, 433âĂŞ442.
[113] S. Varshney, H. Sultan, P. Jain, and S. R. Sarangi. 2019. NanoTherm: An Analytical Fourier-Boltzmann Framework for Full Chip Thermal Simulations. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.
[114] 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.
[115] Arthur H Veen. 1986. Dataflow machine architecture. *ACM Computing Surveys (CSUR)* 18, 4 (1986), 365–396.
[116] Vasanth Venkatachalam and Michael Franz. 2005. Power Reduction Techniques for Microprocessor Systems. *ACM Comput. Surv.* 37, 3 (Sept. 2005), 195âĂŞ237.
[117] N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye. 2000. Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA âĂŹ00)*. Association for Computing Machinery, New York, NY, USA, 95âĂŞ106.
[118] Vinod Viswanath, Rajeev Muralidhar, Hari Seshadri, and Ananth Narayan. 2012. Power Management Methods: From Specification and Modeling, to Techniques and Verification. *Journal of Low Power Electronics* 8 (08 2012), 353–377.
[119] David W. Wall. 1991. Limits of Instruction-Level Parallelism. *SIGPLAN Not.* 26, 4 (April 1991), 176âĂŞ188.
[120] Wm A Wulf and Sally A McKee. 1995. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news* 23, 1 (1995), 20–24.
[121] 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.
[122] Matt Yourst. 2007. PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator. *ISPASS 2007: IEEE International Symposium on Performance Analysis of Systems and Software*, 23–34.
[123] Yiying Zhang and Steven Swanson. 2015. A study of application performance with non-volatile main memory. In *Mass Storage Systems and Technologies (MSST), 2015 31st Symposium on*. IEEE, 1–10.
[124] Amirkoushyar Ziabari. 2014. Power Blurrin: Fast Static and Transient Thermal Analysis Method for Packaged Integrated Circuits and Power Devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22 (03 2014).