International Journal of Technology Management & Humanities (IJTMH)
e-ISSN: 2454 – 566X, Volume 09, Issue 4, (December 2023), www.ijtmh.com

A Survey on Energy-Efficient Microarchitecture Design Strategies for Mobile CPUs and GPUs: Techniques and Future Directions

Prathik Kumar Jannu[1], Javed Ali Mohammad[2], Sri Harsha Panchali[3], Usha Mohani kavirayani[4], Krishna Bhardwaj Mylavarapu[5], Jenitha Pilli[6]
[1]Computer Science Engineering, JNTU Hyderabad
[2]Masters in Telecommunications Middlesex University
[3]Information Systems Engineer CrowdStrike Inc
[4]Kent State University MS in Computer Science
[5]MS in Computer Science University of Illinois Springfield
[6]MS in Computer Science, University of Louisiana at Lafayette

Email ID: prathikjannu@gmail.com

DOI: 10.21590/ijtmh.2023090408

Abstract

The increasing requirement of high-performance mobile computing has exacerbated the desire of energy-efficient microarchitectures, with modern smartphones being dependent on CPUs and GPUs in order to have the compute-intensive workload. Having battery limitations, thermal and heterogeneous task behavior, the energy consumption optimization and the ability to maintain responsiveness have become critical. In this paper, a review of energy efficient microarchitecture approaches to mobile CPUs and GPUs is conducted covering the major methods that include DVFS, clock gating, power gating, cache optimization, and workload- aware scheduling. The survey also examines new technologies including the GPU-specific technologies such as warp scheduling, task migration, and resource-sharing techniques to tackle the parallel and thermal limits of mobile graphics processors. Also, the analysis identifies new architectural trends including AI-based power management, neuromorphic computing, and reconfigurable hardware that strive to enhance scalability of energy in the next-generation devices. On the whole, the paper summarizes existing discoveries, issues, and design compromises that can assist in creating sustainable and high-efficiency mobile architectures.

Keywords: Microarchitecture, Mobile CPUs and GPUs, Energy Efficiency, Technology, Design Strategies.

I. INTRODUCTION

Mobile phones have been developed as simple communications devices but now use computing devices capable of supporting functions such as high-definition video processing, real time data analytics. The functions that desktops used to play are gradually being taken over by modern mobile devices like smartphones and tablet computers. Activities such as high-definition video display, interactive games and web browsing require high levels of computational capability and this is why microprocessor manufacturers have been active in developing better and better mobile CPUs [1]. These increasing computing needs necessitate effective data processing and data storage on-board as well as through client server systems. Consequently, the contemporary smartphones are becoming more dependent on fast CPUs and GPUs to run complex data-intensive tasks. The combined constraint of power in mobile environments and frequency

scaling meant that frequency scaling could not go as high as it needed to be within a short time, so the industry turned to multi-core processors. Practical applications have cast doubt on the effectiveness of the use of these multi-core CPUs. Even quad core processors are often overloaded in most instances and peak performance does not last long [2]. Meanwhile, mobile systems have GPUs (Graphics Processing Unit) that are highly important because they can offload and accelerate parallel tasks (particularly in graphics rendering and workloads). With the usage of ASICs (Application-Specific Integrated Circuits) and DSPs (Digital Signal Processors, GPUs serve to offload the CPU even more, which makes it even more difficult to overstate the role of heterogeneous computing architectures in mobile platforms.

Performance and power consumption optimization in smartphones are quite difficult due to the complexity and heterogeneity of their hardware and software equipment. All the components of the CPU to wireless modules possess different energy needs and different ways of usage in different devices and platforms. In developers are often concerned with functionality, energy efficiency may be neglected, which results in energy holes. Further, power modelling and optimization are not easy because of the variations between smartphone architectural designs [3]. Moreover, the energy efficiency may be optimized at the expense of the performance, which is why it is crucial to develop versatile microarchitectures that are dependent on the realistic usage patterns. Mobile CPU microarchitecture should focus on lightweight and flexible power-management strategies to help solve these problems. DVFS, effective task scheduling, and heterogeneous core design are some of the approaches that can be used to balance the workload requirements with low energy consumption. The inclusion of predictive control is also more responsive and consumes less power that is not essential. All these measures enhance energy-saving CPU performance in current smartphones.

Smartphones are becoming the new reality of life, which leads to the increased carbon footprints due to the increase in power requirements. Therefore, performance coupled with energy efficiency demands the overall understanding of the behavior of a device, its usage patterns and system-wide integration. The emergence of AI, AR/VR [4], and mobile gaming requires mobile gadgets to perform as desktop computers without affecting their battery capacity. Microarchitecture, the internal implementation pipelines, caches, execution units and control logic are critical to a performance-energy consumption tradeoff. This survey looks at energy efficient CPU and GPU microarchitecture methods such as DVFS, resource sharing, power gating, cache optimization and heterogeneous architecture. It discusses the innovations, both processor- and system-level innovations, the current strategies, innovations, and future research directions in mobile energy efficiency.

A. Structure of the Paper

The structure of this paper is as follows: Section II discusses the mobile CPU and GPU microarchitectures. Section III covers energy-efficient design techniques for mobile CPUs. Section IV presents energy-aware strategies of GPUs and microarchitecture. Section V shows the recent studies on related domain. Finally, Section VI concludes the conclusion insights and the future work of the study.

II. MICROARCHITECTURE IN MOBILE SYSTEMS

The central processing unit, or CPU, is the main part of computer systems that executes defined instructions for arithmetic logic and control (I/O) [3]. The graphics processing unit, or GPU, is a specialized part designed to swiftly manage and alter memory in order to speed up the generation of pictures for display. Higher performance may be achieved by combining the special qualities and strengths of CPU and GPU. The CPU's architecture consists of a small number of cores that communicate with caches that can handle a small number of instructions at once. On the other hand, a GPU is made up of several cores that are capable of handling numerous instructions simultaneously. The GPU in modern computers can now do a wide range of multimedia activities, including speeding up image processing, pattern matching, and transcoding (converting) video between many formats.

In terms of computing speed, GPU computation has significantly outperformed CPU computation. As a result, it is among the most intriguing study topics in the realm of contemporary industrial research and

development [5] A GPU is a graphical processing unit that allows to run high-definition graphics on computer, which is necessary for modern computing. It is a single-chip processor, just like the CPU. But as Figure 1 illustrates, the GPU has hundreds of cores, whereas the newest CPUs only have four or eight. The computation of 3D functions is the GPU's primary purpose. The GPU may help the computer become considerably more efficient because these sorts of calculations need a lot of CPU power. Although GPUs were first developed for graphics, they are now used for computation, accuracy, and performance.

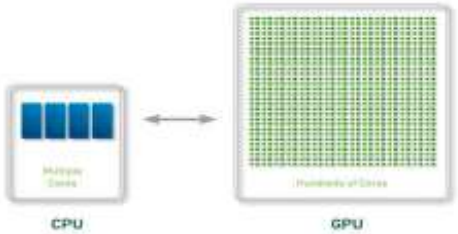

Fig. 1. CPU and GPU

Table I show a brief comparison of CPU and GPU architecture in major areas of operation. It sums up the concepts of the CPUs focus on a high level of single-thread performance, an efficient branching and a balanced workload processing whereas the GPUs are based on a high degree of parallelism, high bandwidth caches, and throughput-driven execution. On the whole, the differences between the design priorities and performance advantages of any two architectures are reflected in the table

TABLE I. COMPARISON BETWEEN CPU AND GPU

Aspects	CPU	GPU
Cache & Compute Units	CPUs use very fast, low-latency caches designed for frequent data reuse and complex operations, supporting balanced compute performance across diverse workloads.	GPUs incorporate extremely high-bandwidth caches and hundreds of math units, enabling efficient execution of large batches of uniform computations simultaneously.
Branching & Memory Access	CPUs manage fine-grained branching with sophisticated control logic, providing fast access to onboard memory for	GPUs also support branching but are optimized for predictable patterns, allowing rapid memory access mainly for

	highly dynamic, instruction-heavy tasks.	structured, parallel workloads like graphics processing.
Concurrency Model	CPUs run diverse processes and threads efficiently, prioritizing responsiveness and task switching across multiple applications in general-purpose environments.	GPUs execute thousands of lightweight threads together, applying the same program to fragments or vertices to maximize throughput on highly parallelizable tasks.
Single-Thread Performance	CPUs deliver strong single-thread performance with large caches and complex pipelines, making them ideal for sequential or non-parallel sections of workloads.	GPUs provide weaker single-thread performance because they focus on massive parallelism rather than deep optimization of individual instruction execution paths.
Parallel Processing	CPUs handle moderate levels of parallelism but remain optimized for mixed workloads requiring combinations of serial and parallel task execution.	GPUs achieve exceptionally high throughput on parallel tasks by distributing work across many cores, making them ideal for compute-intensive, data-parallel operations.

B. Power Consumption Sources in Microarchitectures

Two factors are making power dissipation in microprocessors a major concern for designers: (1) the market for mobile and embedded systems is growing quickly, and in such systems, battery life is crucial and power is at a premium; and (2) complex designs and large on-chip caches present in modern chips require thermal-management strategies to prevent the chip from overheating; this is true not only for mobile computing but also for conventional processor design. Despite more than 10 years of intensive research on low-power design [6], power modelling and optimization at the microarchitectural level have just lately attracted much attention. The literature discusses several methods for processor-level power optimization and system-level power management, with a particular emphasis on memory and cache power management.

- **Memory and Cache Optimizations:** Adds another small L0 cache to hold commonly used instruction which improves the efficiency of the cache and lowers power. Suggests methods directly intended at making caches more energy efficient.
- **Dynamic Power Management Techniques:** This is the implementation of low-power sleep modes on microprocessors using standards such as ACPI. Dynamic supply voltage variation has a great potential and has been commercialized.
- **Microarchitectural Power Modelling Tools:** This is a parameterized model of a power simulator (within 10% of actual microprocessors) that can be used to investigate power vs. performance trade-offs. Attention to Datapath-dominated architectures and concentrating on the effect of memory architecture and compiler improvements on power use.
- **Microarchitectural Power Reduction Techniques:** Leverages confidence estimation to block execution of probable mis- predicted branches, saving on unnecessary energy. Suggests profiling-guided program execution in order to attain reduced power consumption. Calculates the optimum energy-delay trade-offs in the case of the processor configuration of superscalar processor (non-adaptive). Suggests complexity-adaptive processors, but the mechanism of adaptation is not given. Introduces a crude executing adaptive scheme, and discusses the issue queue resizing dynamically. Profiles using IPC, and modulates the use of energy by various parts of the code using a fixed instruction window.

C. Metrics for Evaluating Energy Efficiency

Energy Efficiency Metrics are applied to measure performance of computing systems, particularly mobile and embedded computing and data centres and processors. The main standard measures include the following:

1) Performance per Watt

A major metric that can be used to assess performance per Watt measures the energy efficiency of mobile devices, including wearables, tablets, and smartphones, where battery life is crucial [7]. Performance per watt mathematical equation presented in Equation (1).

$$Performance\ per\ Watt = \frac{Throughput\ or\ Task\ Rate}{Power\ Consumption(W)} \qquad (1)$$

2) Energy Delay Product (EDP)

The compute requirement of a removable job is measured using the Energy Delay Product (EDP) [8]. Equation (2) defines EDP, a performance metric that takes into account power and execution time.

$$EDP = T \times E = T^2 \times P \qquad (2)$$

In this equation, T denotes the total time that the task took to execute, E denotes the energy that was used to execute the task, and P denotes average power drawn during the execution. All these variables are used to measure the duration a computation takes, and the amount of energy consumed. An effective performance indicator is the Energy Delay Product and energy behavior that can be used to measure the computational demand of power-sensitive systems by synthesizing them.

3) Thermal Design Power (TDP)

The mobile AP's TDP value is mostly determined by its size, the application set's thermal design, and its power efficiency. [9] Due to the mobile set's limited heat dissipation capability, the TDP value is significantly smaller than the IDP even if it rises with the size of the application set (see Figure 2) which shows the graph points to the fact that the power indicators depend on the size of the equipment and emphasizes the need to pay attention to several indicators of power (not only TDP) when designing mobile microarchitectures. It also demonstrates that practical use (SDP) can often be more intensive than what TDP indicates to ensure energy-efficient design is of paramount importance and of particular concern in thermally challenged devices such as smartphone and tablets. Helps design cooling systems; indirectly linked to energy efficiency since lower TDP usually means better thermal and power efficiency.

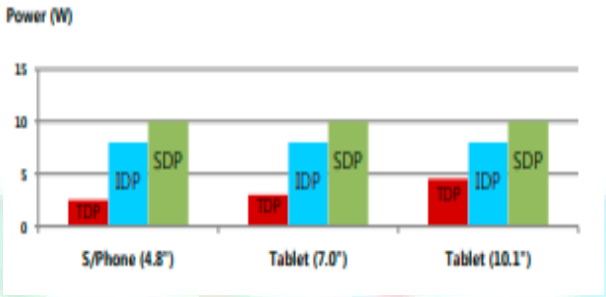

Fig. 2. Thermal Power of Mobile AP

III. Energy-Efficient Design Techniques for Mobile CPUs

This is a summary of some important methods of energy-efficiency CPU microarchitecture, and these methods are crucial, especially for embedded and mobile systems where power efficiency is crucial.

D. Dynamic Voltage and Scaling (DVS)

Devices can adapt their processor voltage and frequency during operation to meet workload requirements thanks to Dynamic Voltage Scaling (DVS). Mobile devices with a limited battery supply may be made more energy-efficient by reducing the voltage linearly, which results in quadratic energy savings. DVS has the drawback of producing a processor with inconsistent performance [10]. However, the end-user may not notice this varying performance if DVS algorithms are properly developed and executed. Accurately estimating future workload requirements and modulating the processor voltage (and consequently the frequency) to that level is the answer to providing this characteristic.

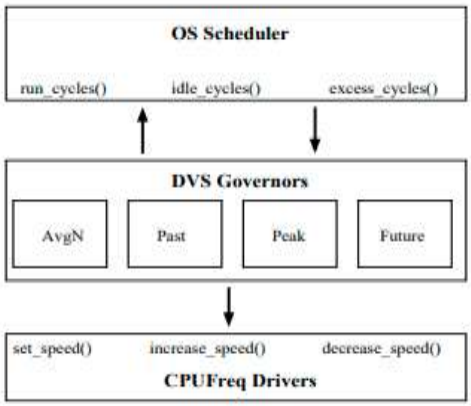

Fig. 3. DVS System Architecture

In Figure 3, the DVS architecture is shown. The interval technique is used by DVS algorithms to forecast future workloads by recording workload information from prior intervals. Prediction accuracy has been the primary focus of the majority of prior research in this field, which is simulation-based. In order to adopt a new strategy, implement and assess the performance of current DVS algorithms on actual hardware that is running an actual operating system (OS) and carrying out actual workloads.

DVS may be applied at many levels, including:

- In the processor,
- In the OS scheduler,
- In the compiler, and
- In the application

E. Clock Gating

This method is based on the idea of turning off the clock to flip-flops or memory components that aren't involved in the present processing; as a result, their needless flipping is halted, saving the dynamic power needed for their speculative toggling. Another way to reduce power is to utilize fewer gates in each stage, which lowers the operating frequency at the expense of performance deterioration. Performance deterioration may be ignored by reducing dynamic power usage by 20% to 40% [11] As a result, the cost of performance deterioration is acceptable at this power-saving level and is predicated on striking a balance between these two factors. Over 60% of the overall dynamic power consumption are connected to clocks. The most widely used and well-known clock gating (CG) technique is found in many systems. An example of clock-gating strategy is presented in Figure 4 which provides enable signals to the gated clocks to enable only those circuits required and reduce switching activities resulting in lower dynamic power consumption.

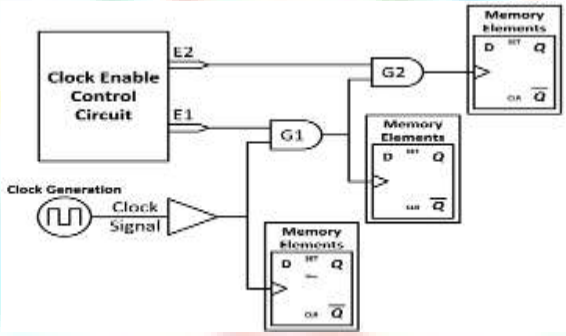

Fig. 4. Clock Gating Scheme

F. Power Gating

Dynamic power management is a technology that reduces running power consumption by shutting off unnecessary FPGA components. This is accomplished through switching off the parts or sections that do not need a power supply while reducing leakage currents. Figure 5 illustrates an example of a power-gating circuit which switches power to the logic block on when it is idle by cutting off the power to the block with sleep transistors, decreasing the leakage power.

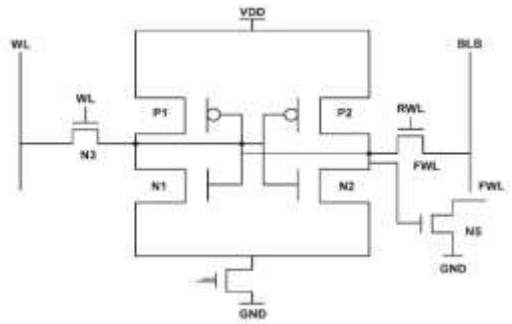

Fig. 5. Power Gating Scheme

G. Key Components

The power-gating architecture is made up of activity monitors, decision logic and sleep transistors which tell whether a block needs to be switched off. Certain blocks of FPGA can be power-gated without impacting the rest of the blocks whereas specific wakeup logic is used to make sure that when an activity is reinstated the power is brought back.

1) Power Gating Control Logic

The control logic monitors the activity of each block, analyzes shutdown conditions and activates sleep transistors in order to disconnection idle blocks. It helps in the passage of states between active and gated states to preserve functional integrity.

- **Activity Monitor:** This provides the ability to keep a constant frequency of various blocks of the FPGA.
- **Decision Logic:** This determines in advance potential states in which a block can be closed or possibly re-energized after being in a standby condition.
- **Sleep Transistors:** In circuitry, the power flow to inactive circuits is switched off using controlled high-threshold transistors.
- **Power Gated Blocks:** FPGA components that may be disconnected and which may be turned off without affecting other FPGA components.
- **Wakeup Logic:** This makes the smooth transition between the condition in which the circuit is powered and the condition in which power is switched off.

2) Process Flow

This is done by checking the activity of the blocks and then activating the sleep transistors to cut power to non-active parts. The gating of power occurs when idle and the reverse is true.

- **Control Logic Assessment:** Provides a cord and checks the activity of the different blocks in the FPGA.
- **Sleep Transistor Activation:** It now wakes up sleep transistors meant to switch off the power supply to the blocks that are inactive or not in use.
- **Power Gating:** It abruptly puts the out of use blocks to sleep to reduce Static Power Consumption.
- **Reactivation:** The logic of the wake up is employed to re-energize the blocks driven by the power gates to the rest of the circuit when they are needed.

IV. ENERGY-AWARE GPU MICROARCHITECTURE AND EMERGING TRENDS

In mobile platforms, graphics processing units (GPUs) have also become essential since they enable general-purpose computing tasks like image processing and augmented reality in addition to graphics rendering. They, however, have substantial power requirements as a consequence of their naturally parallel design and high throughput. Mobile devices need to have energy efficient design of the GPU microarchitecture that balances between performance, power consumption and thermal constraints. The key techniques used in attaining this balance are discussed in this section.

H. Resource Sharing and Task Migration

Resource sharing and task migration [12] has become one of the most important strategies of improving microarchitectural efficiency in energy-constrained mobile environments. Since current mobile SoCs (System-on-Chips) combine CPU and GPU cores, along with specialized accelerators, task scheduling and switching between them to use the most energy-efficient resources is essential to best maintain their energy use and performance. Resource sharing entails the dynamic sharing of computing resources like execution units, caches or memory bandwidth among two or more tasks or processing units. Shared microarchitectures allow more flexible and efficient utilization of resources (including avoiding underutilization and energy waste), unlike in a situation of statical partitioning. Indeed, can use mobile

GPUs as an example and have seen that each warp or kernel may share a few pipeline elements or memory access paths, minimizing redundant activity and total power consumption.

In mobile systems, real-time factors like thermal conditions, workload characteristics or user interaction pattern can be used to drive task migration. The systems can also be changed to meet the changing performance and energy budgets by intelligently delegating work to the best processing unit [13]. A number of mobile systems today use heterogeneous-aware runtime schedulers which proactively coordinate resource sharing as well as task translocation. These systems keep track of the progress of tasks, resource competition and energy profiles to find the best points of execution. As an example, the computing-intensive parallel workloads can be offloaded to the GPU at the time of low-power states of the CPU, and latency-sensitive tasks can be run on CPU cores.

1) Warp Scheduling and Throttling

Warp scheduling and throttling is of particular significance to mobile GPU microarchitectures in terms of performance-energy tradeoffs. One of the threads in a SIMT (Single Instruction, Multiple Thread) architecture, also known as the common usage of modern GPUs, is called a warp and performs the same instruction simultaneously. The management of these warps is imperative in ensuring that the energy consumption is minimized without causing serious deterioration in the performance a major challenge where power is limited as applies to mobile environments [14]. Warp scheduling is the method which is used to select the warps in order of their execution [15]. Conventional scheduling algorithms (i.e. round-robin or greedy scheduling) can maximize throughput, but result in inefficient energy consumption because of idle resources or memory stalls. By contrast, energy-aware warp schedulers are made to trade off both performance and power by dynamically enabling warps depending on dynamic workload properties, resource availability and temperature.

Conversely, warp throttling is the intentional act of increasing the quantity of active warps to reduce power consumption and thermal power production. Although this can momentarily decrease parallelism, it avoids power-hungry behavior, e.g. excess memory contention, or overheating that can be harmful to stability of the system but power consumption in mobile devices. Adaptive throttling methods dynamically modify the execution rate depending on the runtime measurements like the power budget, temperature limits and the behaviour of the application phase.

2) Emerging Trends of Mobile CPUs and GPUs

Future Energy-efficient Microarchitecture Minor emergencies include incorporating AI/ML-based power management, neuromorphic and in-memory computing, adaptive and reconfigurable hardware, and the trade-off between security and energy limitation. These innovations are targeted to improve performance, save energy and offer secure robust systems that see the rise of intelligent sustainable and high-performance mobile computing system in future.

a) AI-Driven Power Management

A new paradigm of intelligent applications made possible by artificial intelligence (AI) is completely altering the way live. Particularly for applications running on mobile devices (such as smartphones, wearable technology, and automobiles), many of these AI-enabled apps [16] have extremely strict latency requirements. In order to provide mobile AI applications with quicker and more energy-efficient computing, smaller and quantized deep neural network (DNN) models are created for mobile devices [17]. Nevertheless, little is known about how AI models use energy in mobile devices. It is necessary to thoroughly examine these models' behavior utilizing a variety of processing sources in order to predict their energy consumption as well as their numerous uses, including vision and non-vision.

b) Neuromorphic and Domain-Specific Architectures

Neuromorphic and domain-specific architectures offer power-saving solutions based on alternative CPU and GPUs particularly to mobile and embedded applications. Neuromorphic chips are based on brain-like processing, event-driven computing with low power, and high-quality parallel processing. The examples of commercial projects include Intel Loihi, NeuroMem, and IBM ZISC, which are efficient to perform pattern-recognition tasks that are appropriate when embedded AI is needed. Neuromorphic chips and FPGAs are also showing high levels of energy efficiency over GPUs as task difficulty increases, and FPGAs

have been demonstrated with configurable low-power acceleration and neuromorphic designs have been shown to save even more energy with a fundamentally different model of computing. Such architectures decrease by a significant percentage the overhead of data-movement, which is an important source of energy waste. Their special designs enable them to be more applicable in the next-generation intelligent edge devices.

c) Adaptive Architectures and Reconfigurable Logic

Another design is in reconfigurable computing frameworks e.g., FPGAs, which allow flexibility in mapping arbitrary applications but they have increased power usage and decreased performance to custom hardware. The conventional techniques such as voltage scaling are compromising between energy and performance [18] Co-designing a method for the significantly enhanced Energy-Delay Product (EDP) using circuit architecture and software. To enhance the read dominant memory access, the architecture proposes an asymmetric memory cell to increase the read access and lessen power especially in processes where some logic values are to be stored.

V. LITERATURE REVIEW

According to this literature review, existing advances in CPU/GPU energy optimization focus on methods of enhancing efficiency, performance, and workload management in the modern computing systems.

Yang et al. (2019) suggested the Energy-aware CPU Frequency Scaling (EFS) method, which chooses the CPU frequency that can balance CPU energy and data transmission energy savings. also suggest a way to decide when and how much data to download because the current video streaming apps' downloading schedules are not energy-efficient. It show that the EFS algorithm can cut energy consumption by 30% using real measurements and trace-driven simulations [19].

Chen, Cheng and Hsiu (2019) suggested a CPU-GPU governance structure that is online and user-centric. Categorize rendered game frames into redundant/changing frames to meet user demand, classify an application into GPU-sensitive/insensitive phases to comprehend the application's demand, and ascertain the frequency scaling intents of the CPU and GPU to capture processor demand in order to fill in the identified information gaps. It use a frequency-scaling intent communicator, a unified policy selector, and a necessary workload estimate in the framework to conserve energy in response to the measured demand [20].

Bermejo, Juiz and Guerrero (2018) suggested a new index called CiS2 (Consolidation index for Server CPU in Saturation) to measure the trade-off between energy and performance in consolidation, or the relationship between energy efficiency and performance degradation of virtual machine consolidation under parallel workload execution. Therefore, CiS2 offers a method to assist performance engineers in determining if several merged virtual machines are appropriate. CiS2 might be expanded to include any further server non-functionalities [21].

Valery, Liu and Wu (2017) suggested TransferCL, a deep learning architecture that facilitates mobile transfer learning. To speed up deep learning computation on mobile devices, method depends on the cooperation of the integrated GPU and the multicore CPU. And take into account three key concerns: memory management, power efficiency, and performance/portability trade-off. Then, suggest solutions and carry out tests to assess them [22].

Cheng et al. (2016) suggested a combined CPU-GPU power management strategy for mobile games that is behavior-aware. also put power-saving policy into practice on the actual platform, where the assessment results demonstrate that behavior-aware policy can greatly lower power consumption while enhancing game performance. On average, the policy increases electricity efficiency by 18% and 5% when compared to the state-of-the-art policy and the present policy utilized in platform, respectively [23].

Lee, Jang and Kim (2016) proposed a technique that improved the processing time by modifying CPU–GPU task parallelism, sliding window parallelism, scale image parallelism, dynamic thread allocation, and

local memory optimization. According to the experimental findings, the suggested approach outperformed the well-optimized OpenCV CPU implementation in terms of processing time by 3.3~6.29. The suggested approach can be modified for use with mobile CPUs and GPUs in additional applications [24].

Table II gives a brief summary of significant CPU-GPU optimization works, detailing ways of enhancing power efficiency and performance. As much as these methods demonstrate evident improvements, there are still issues to handle the various workloads and assure a stable level of effectiveness. The future research hopes to create more flexible and hardware-conscious optimization methods.

TABLE II. RECENT STUDIES REVIEW OF CPU–GPU ENERGY EFFICIENCY AND OPTIMIZATION TECHNIQUES

Reference	Study On	Approach	Key Findings	Challenges	Future Direction
Yang et al. (2019)	Energy-conscious CPU frequency scaling for streaming videos on mobile devices	Proposed EFS algorithm to select optimal CPU frequency and suggested improved download scheduling	Achieved up to 30% energy reduction through balanced CPU and data-transfer energy use	Need for real-time adaptation across diverse apps and network conditions	Extend to broader streaming platforms and integrate predictive workload models
Chen, Cheng & Hsiu (2019)	User-centric CPU–GPU governance for mobile gaming	Classified frames and phases; built workload estimator, policy selector, and frequency-scaling communicator	Improved energy savings while meeting user demand and system responsiveness	Managing dynamic game behavior and heterogeneous workloads	Enhance generalization across game engines and integrate machine learning for prediction
Bermejo, Juiz & Guerrero, (2018)	VM consolidation performance–energy tradeoff	Introduced CiS2 index to quantify CPU saturation impact on performance and energy	CiS2 helps engineers decide VM consolidation levels effectively	Requires tuning for various server architectures and workloads	Extend CiS2 to broader non-functional metrics and cloud environments
Valery, Liu & Wu (2017)	Transfer learning acceleration on mobile devices	Proposed TransferCL, combining CPU–GPU collaboration with optimized models for mobile deep learning	Enhanced performance, portability, energy efficiency, and memory usage in on-device learning	Balancing model complexity with mobile hardware limits	Apply to real-time applications and optimize for emerging mobile AI accelerators
Cheng et al. (2016)	Mobile gaming with behavior-aware CPU-	Implemented real-time power-saving policy using behavior-aware application profiling	Achieved 18% and 5% higher power efficiency than platform	Maintaining performance while reducing power	Broaden policy applicability to diverse game genres

	GPU power management		default and state-of-the-art policies	remains complex	and mobile platforms
Lee, Jang & Kim (2016)	Parallelism techniques to accelerate mobile CPU–GPU computation	Used memory optimization, sliding window parallelism, dynamic thread allocation, and task parallelism	Achieved 3.3–6.29× faster computation vs. optimized CPU OpenCV	Managing memory constraints and parallelism overhead	Extend techniques to new mobile GPU architectures and other computational applications

VI. Conclusion and Future Work

The growing use of mobile devices in computationally intensive applications has put the development of energy-efficient CPU and graphics card microarchitecture design in the criterion of importance regarding research priorities. Intense performance requirements require the emergence of new scaling methodologies owing to the strong power, thermal and battery limitations. DVFS, power gating, and clock gating, and cache optimization and heterogeneous scheduling methods are some of the most important energy-saving methods that were discussed in this paper to explain the current processors as a means of handling workload diversity with consumption minimization. The strategies with GPUs in the paper included warp scheduling, throttling, and task migration which are used to balance throughput with real-time energy budgets. Moreover, new paradigms like AI-based power control, neuromorphic processors and adaptable reconfigurable architectures suggest good directions which the future mobile platforms can take. Taken together, the presented insights support the notion that to attain sustainable performance, workload-conscious, architecture-conscious and integrated design solutions must be employed.

Future studies need to investigate AI-based runtime controllers that can automatically color the CPU-GPU parameters according to the real-time behavior prediction. Additional solutions to this include integration of neuromorphic accelerators and in-memory computing that will minimize data-movement overheads, which are a significant contribution to the energy waste. Also, more comprehensive tests with actual mobile load and thermal models should be conducted to confirm the suggested approaches. The next generation energy efficient mobile systems will require cross-layer co-design of circuits, architecture and software.

References

1. K. Patrick, W. G. Griswold, F. Raab, and S. S. Intille, “Health and the Mobile Phone,” *Am. J. Prev. Med.*, vol. 35, no. 2, pp. 177–181, 2008, doi: 10.1016/j.amepre.2008.05.001.
2. C. Gao, A. Gutierrez, M. Rajan, R. G. Dreslinski, T. Mudge, and C. J. Wu, “A study of mobile device utilization,” *ISPASS 2015 - IEEE Int. Symp. Perform. Anal. Syst. Softw.*, no. September, pp. 225–234, 2015, doi: 10.1109/ISPASS.2015.7095808.
3. P. K. D. Pramanik *et al.*, “Power Consumption Analysis, Measurement, Management, and Issues: A State-of-the-Art Review of Smartphone Battery and Energy Usage,” *IEEE Access*, vol. 7, pp. 182113–182172, 2019, doi: 10.1109/ACCESS.2019.2958684.
4. R. Karne, “Virtual reality driving simulator for analysis of user response time,” 2019.
5. *Ghorpade, “GPGPU Processing in CUDA Architecture,” Adv. Comput. An Int. J., vol. 3, no. 1, pp. 105–120, 2012, doi: 10.5121/acij.2012.3109.*
6. Iyer and D. Marculescu, “Microarchitecture-level power management,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 10, no. 3, pp. 230–239, 2002, doi: 10.1109/TVLSI.2002.1043326.

7. Pathania, A. E. Irimiea, A. Prakash, and T. Mitra, "Power-performance modelling of mobile gaming workloads on heterogeneous MPSoCs," *Proc. - Des. Autom. Conf.*, vol. 2015-July, 2015, doi: 10.1145/2744769.2744894.
8. H. Qian and D. Andresen, "An energy-saving task scheduler for mobile devices," *2015 IEEE/ACIS 14th Int. Conf. Comput. Inf. Sci. ICIS 2015 - Proc.*, no. July, pp. 423–430, 2015, doi: 10.1109/ICIS.2015.7166631.
9. H. K. Kwon, J. Hwang, H. Chung, M. Kang, H. D. Cho, and Y. M. Kim, "Thermal power of mobile application processor," *Proc. - 2012 45th Int. Symp. Microelectron. IMAPS 2012*, pp. 866–872, 2012, doi: 10.4071/isom-2012-wp34.
10. D. Tam, W. Tsang, and C. Drula, "Dynamic Voltage Scaling in Mobile Devices," *Work*, no. 1, pp. 1–6, 2003.
11. Shahid, S. Arif, M. Y. Qadri, and S. Munawar, "Power optimization using clock gating and power gating: A review," *Innov. Res. Appl. Next-Generation High Perform. Comput.*, no. July, pp. 1–20, 2016, doi: 10.4018/978-1-5225-0287-6.ch001.
12. V. M. L. G. Nerella, "Automated cross-platform database migration and high availability implementation," *Turkish J. Comput. Math. Educ.*, vol. 9, no. 2, pp. 823–835, 2018.
13. Pattnaik *et al.*, "Opportunistic computing in GPU architectures," in *Proceedings of the 46th International Symposium on Computer Architecture*, in ISCA '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 210–223. doi: 10.1145/3307650.3322212.
14. S. Gupta and A. Mathur, "Enhanced Flooding Scheme for AODV Routing Protocol in Mobile Ad Hoc Networks," in *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, IEEE, Jan. 2014, pp. 316–321. doi: 10.1109/ICESC.2014.60.
15. S. Song, M. Lee, J. Kim, W. Seo, Y. Cho, and S. Ryu, "Energy-efficient scheduling for memory-intensive GPGPU workloads," *Proc. -Design, Autom. Test Eur. DATE*, pp. 4–9, 2014, doi: 10.7873/DATE2014.032.
16. W. Wang and K. Siau, "Artificial Intelligence, Machine Learning, Automation, Robotics, Future of Work and Future of Humanity:," *J. Database Manag.*, vol. 30, pp. 61–79, 2019, doi: 10.4018/JDM.2019010104.
17. S. Garg, "AI/ML Driven Proactive Performance Monitoring, Resource Allocation and Effective Cost Management in SaaS Operations," *Int. J. Core Eng. Manag.*, vol. 6, no. 6, pp. 263–273, 2019.
18. S. Paul, S. Chatterjee, S. Mukhopadhyay, and S. Bhunia, "Energy-efficient reconfigurable computing using a circuit-architecture- software co-design approach," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 1, no. 3, pp. 369–380, 2011, doi: 10.1109/JETCAS.2011.2165232.
19. W. Hu and G. Cao, "Energy-Aware CPU Frequency Scaling for Mobile Video Streaming," *IEEE Trans. Mob. Comput.*, vol. 18, no. 11, pp. 2536–2548, Nov. 2019, doi: 10.1109/TMC.2018.2878842.
20. W.-M. Chen, S.-W. Cheng, and P.-C. Hsiu, "A User-Centric CPU-GPU Governing Framework for 3-D Mobile Games," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 38, no. 5, pp. 961–974, 2019, doi: 10.1109/TCAD.2018.2834404.
21. Bermejo, C. Juiz, and C. Guerrero, "On the Linearity of Performance and Energy at Virtual Machine Consolidation: The CiS2 Index for CPU Workload in Server Saturation," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, Jun. 2018, pp. 928–933. doi: 10.1109/HPCC/SmartCity/DSS.2018.00154.
22. O. Valery, P. Liu, and J.-J. Wu, "CPU/GPU Collaboration Techniques for Transfer Learning on Mobile Devices," in *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, 2017, pp. 477–484. doi: 10.1109/ICPADS.2017.00069.
23. Z. Cheng, X. Li, B. Sun, J. Song, C. Wang, and X. Zhou, "Behavior-Aware Integrated CPU-GPU Power Management for Mobile Games," in *2016 IEEE 24th International Symposium on*

Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, Sep. 2016, pp. 439–444. doi: 10.1109/MASCOTS.2016.18.

24. Y. Lee, C. Jang, and H. Kim, "Accelerating a Computer Vision Algorithm on a Mobile SoC Using CPU-GPU Co-processing - A Case Study on Face Detection," in *2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2016, pp. 70–76.
25. Routhu, K. K. (2022). From Case Management to Conversational HR: Redefining Help Desks with Oracle's AI and NLP Framework. *International Journal of Science, Engineering and Technology*, *10*(6).
26. Vattikonda, N., Gupta, A. K., Polu, A. R., Narra, B., Buddula, D. V. K. R., & Patchipulusu, H. H. S. (2022). Blockchain Technology in Supply Chain and Logistics: A Comprehensive Review of Applications, Challenges, and Innovations. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(3), 72-80.
27. Attipalli, A., BITKURI, V., Mamidala, J. V., Kendyala, R., & KURMA, J. (2022). Empowering Cloud Security with Artificial Intelligence: Detecting Threats Using Advanced Machine learning Technologies. *Available at SSRN 5741263*.
28. Padur, S. K. R. (2022). Intelligent resource management: AI methods for predictive workload forecasting in cloud data centers. *J. Artif. Intell. Mach. Learn. & Data Sci*, *1*(1), 2936-2941.
29. Routhu, K. K. (2022). From RFID to Geofencing: IoT-Enabled Smart Time Tracking in Oracle HCM Cloud. *International Journal of Science, Engineering and Technology*, *10*(4).
30. Polam, R. M., Kamarthapu, B., Kakani, A. B., Nandiraju, S. K. K., Chundru, S. K., & Vangala, S. R. (2022). Data Security in Cloud Computing: Encryption, Zero Trust, and Homomorphic Encryption. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 31-41.
31. Padur, S. K. R. (2022). AI augmented platform engineering, transforming developer experience through intelligent automation and self optimizing internal platforms. *International Journal of Science, Engineering and Technology*, *10*(5), 10-5281.
32. Polu, A. R., Buddula, D. V. K. R., Narra, B., Gupta, A., Vattikonda, N., & Patchipulusu, H. (2021). Evolution of AI in Software Development and Cybersecurity: Unifying Automation, Innovation, and Protection in the Digital Age. *Available at SSRN 5266517*.
33. Padur, S. K. R. (2020). From centralized control to democratized insights: Migrating enterprise reporting from IBM Cognos to Microsoft Power BI. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol*, *6*(1), 218-225.
34. Bitkuri, V., Kendyala, R., Kurma, J., Mamidala, V., Enokkaren, S. J., & Attipalli, A. (2021). Systematic Review of Artificial Intelligence Techniques for Enhancing Financial Reporting and Regulatory Compliance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(4), 73-80.
35. Padur, S. K. R. (2019). Machine learning for predictive capacity planning: Evolution from analytical modeling to autonomous infrastructure. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *5*(5), 285-293.
36. Attipalli, A., Enokkaren, S., BITKURI, V., Kendyala, R., KURMA, J., & Mamidala, J. V. (2021). Enhancing Cloud Infrastructure Security Through AI-Powered Big Data Anomaly Detection. *Available at SSRN 5741305*.
37. Singh, A. A. S., Tamilmani, V., Maniar, V., Kothamaram, R. R., Rajendran, D., & Namburi, V. D. (2021). Predictive Modeling for Classification of SMS Spam Using NLP and ML Techniques. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(4), 60-69.
38. Padur, S. K. R. (2020). AI augmented disaster recovery simulations: From chaos engineering to autonomous resilience orchestration. *International Journal of Scientific Research in Science, Engineering and Technology*, *7*(6), 367-378.

39. Reddy Padur, S. K. (2021). From Scripts to Platforms-as-Code: The Role of Terraform and Ansible in Declarative Infrastructure Rollouts. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 621-628.
40. Kothamaram, R. R., Rajendran, D., Namburi, V. D., Singh, A. A. S., Tamilmani, V., & Maniar, V. (2021). A Survey of Adoption Challenges and Barriers in Implementing Digital Payroll Management Systems in Across Organizations. *International Journal of Emerging Research in Engineering and Technology*, *2*(2), 64-72.
41. Padur, S. K. R. (2018). Autonomous cloud economics: AI driven right sizing and cost optimization in hybrid infrastructures. *International Journal of Scientific Research in Science and Technology*, *4*(5), 2090-2097.
42. Rajendran, D., Namburi, V. D., Singh, A. A. S., Tamilmani, V., Maniar, V., & Kothamaram, R. R. (2021). Anomaly Identification in IoT-Networks Using Artificial Intelligence-Based Data-Driven Techniques in Cloud Environmen. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(2), 83-91.
43. Padur, S. K. R. (2021). Bridging Human, System, and Cloud Integration through RESTful Automation and Governance. *the International Journal of Science, Engineering and Technology*, *9*(6).
44. Attipalli, A., BITKURI, V., KURMA, J., Enokkaren, S., Kendyala, R., & Mamidala, J. V. (2021). A Survey of Artificial Intelligence Methods in Liquidity Risk Management: Challenges and Future Directions. *Available at SSRN 5741342*.
45. Padur, S. K. R. (2021). From Control to Code: Governance Models for Multi-Cloud ERP Modernization. *International Journal of Scientific Research & Engineering Trends*, *7*(3).
46. Routhu, K. K. (2021). Harnessing AI Dashboards in Oracle Cloud HCM: Advancing Predictive Workforce Intelligence and Managerial Agility. *International Journal of Scientific Research & Engineering Trends*, *7*(6).
47. Padur, S. K. R. (2021). Deep learning and process mining for ERP anomaly detection: Toward predictive and self-monitoring enterprise platforms. *Available at SSRN 5605531*.