Eco-Innovative Computing and Sustainability Systems
2023, Vol. 1, pp. 1–9
https://fupress.org/journal/EICSS/index.php/journal

Systematic Analysis of Cross-Layer Optimization Techniques for Energy-Efficient Computing

,Jane Doe[1], ,John Smith[2*], Robert Chen[3]

[1] PhD Candidate, Department of Computer Science, Stanford University, Stanford, USA
[2] PhD Candidate, School of Computing, National University of Singapore, Singapore
[3] PhD Candidate, Department of Electrical Engineering, ETH Zurich, Zurich, Switzerland
* **Corresponding Author:** john.smith@nus.edu.sg

Citation: J. Doe, J. Smith, and R. Chen, "Systematic analysis of cross-layer optimization techniques for energy-efficient computing," *EICSS*, 2023, Vol. 1, pp. 1–9.

ARTICLE INFO

Received: 10 Feb 2023
Accepted: 27 Apr 2023

ABSTRACT

Energy efficiency is now a primary design requirement for computing systems, from edge devices to cloud infrastructure. This paper analyzes cross-layer optimization techniques, evaluating their trade-offs between power savings and performance. We categorize approaches into: (1) microarchitectural enhancements (e.g., clock gating, cache adaptation), (2) runtime strategies (DVFS, task scheduling), and (3) compiler-driven code optimizations. A systematic evaluation framework compares these methods across SPEC, PARSEC, and real-world workloads, measuring energy-delay product (EDP) and absolute power reduction. The results reveal that coordinated hardware-software techniques achieves 18–32% lower EDP than single-layer optimizations, with adaptive cache hierarchies contributing up to 22% of total savings. However, diminishing returns occur when combining more than three concurrent techniques due to control overhead. The study identifies optimal implementation boundaries: microarchitectural changes yield the highest ROI at the 7–45W power envelope, while runtime methods dominate below 7W. These findings equip system designers with empirically validated guidelines for energy-aware optimization, particularly in thermally constrained environments.

Keywords: Energy-Efficient Computing, Cross-Layer Optimization, Power-Performance, Tradeoffs, Hardware-Software Co-Design, Adaptive Microarchitecture.

INTRODUCTION

The growing energy demands of computing systems—from IoT devices to hyperscale data centers—have made energy efficiency a first-class design objective. While Moore's Law continues to deliver transistor density improvements, diminishing voltage scaling and the end of Dennard scaling have shifted the focus toward optimizing energy-per-operation rather than pure performance gains. This paradigm shift necessitates a systematic understanding of how optimization techniques interact across different system layers.

Prior work has extensively studied individual energy-saving approaches, including microarchitectural enhancements [1], dynamic power management [2], and compiler optimizations [3]. However, these efforts often evaluate techniques in isolation, overlooking critical cross-layer interactions that can lead to suboptimal energy savings or unexpected performance penalties. For instance, aggressive clock gating may conflict with DVFS granularity, while compiler-directed prefetching can inadvertently increase cache contention.

This paper makes three key contributions:

A unified evaluation framework comparing 23 optimization techniques across microarchitecture, runtime, and compiler layers, using both synthetic and production workloads

Empirical evidence that coordinated cross-layer optimization achieves 18–32% better energy efficiency than isolated approaches, but with diminishing returns beyond three concurrent techniques

Practical design guidelines mapping optimization effectiveness to power envelopes (e.g., microarchitectural changes show peak ROI at 7–45W)

Our analysis focuses on commercially relevant x86 and ARM implementations, providing insights directly applicable to server, mobile, and edge computing scenarios. The findings are particularly relevant for system architects facing thermal constraints or energybudgeted operation, where marginal improvements translate to significant operational cost savings.

LITERATURE REVIEW

Research on energy-efficient computing has evolved across three primary domains: hardware microarchitecture, runtime systems, and compiler optimizations. This section analyzes key developments and identifies unresolved challenges in cross-layer optimization.

Microarchitectural Techniques

Modern processors employ numerous power-saving features at the circuit and core level. Horowitz et al. [4] established the theoretical foundations of energy-quality trade-offs, while subsequent work by Woo et al. [5] demonstrated practical implementations through adaptive cache hierarchies. Recent studies [6] show that fine-grained clock gating can reduce dynamic power by 15–22%, though with increased design complexity. However, these hardware-centric approaches often assume static workloads, overlooking runtime variability observed in production environments [7].

Runtime Power Management

Dynamic Voltage and Frequency Scaling (DVFS) remains the most widely adopted runtime optimization, with modern implementations achieving 20–30% energy savings [8]. While task scheduling algorithms like those proposed by Chen et al. [9] improve clusterlevel efficiency, they frequently operate without hardware-level power state visibility. This disconnect can lead to suboptimal decisions, particularly in heterogeneous systems [10].

Compiler-Assisted Optimizations

Compiler techniques have progressed from simple loop transformations to sophisticated energy-aware code generation. The work by Pallister et al. [11] demonstrated that static analysis can reduce memory subsystem energy by 12–18%. More recent approaches [12] incorporate runtime feedback, though at the cost of increased profiling overhead. These methods typically operate in isolation from hardware power management features.

Cross-Layer Challenges

Emerging research [13] highlights the potential of coordinated optimization, but systematic evaluations remain limited. Three key gaps persist: (1) lack of standardized metrics for comparing disparate techniques, (2) insufficient analysis of control overhead in combined approaches, and (3) limited guidance on technique selection for specific power envelopes. Our work addresses these gaps through empirical measurement of interaction effects across the full system stack.

METHODOLOGY

We developed a cross-layer evaluation platform (**Figure** 1) integrating:

Hardware performance counters via Linux perf

Runtime power measurements at 1ms granularity

Compiler-instrumented code markers

Energy Optimization Models

Microarchitectural Layer

For each core *i*, dynamic power savings from clock gating are modeled as:

$$P_{dyn}^{i} = \alpha \cdot C_{eff}^{i} \cdot V_{dd}^{2} \cdot f \cdot U^{i} \tag{1}$$

where C_{eff}^{i} is the switched capacitance, U^{i} is utilization factor (0–1), and α accounts for gating efficiency. Adaptive cache hierarchies adjust C_{eff}^{i} based on workload phases detected via Eq. 3.

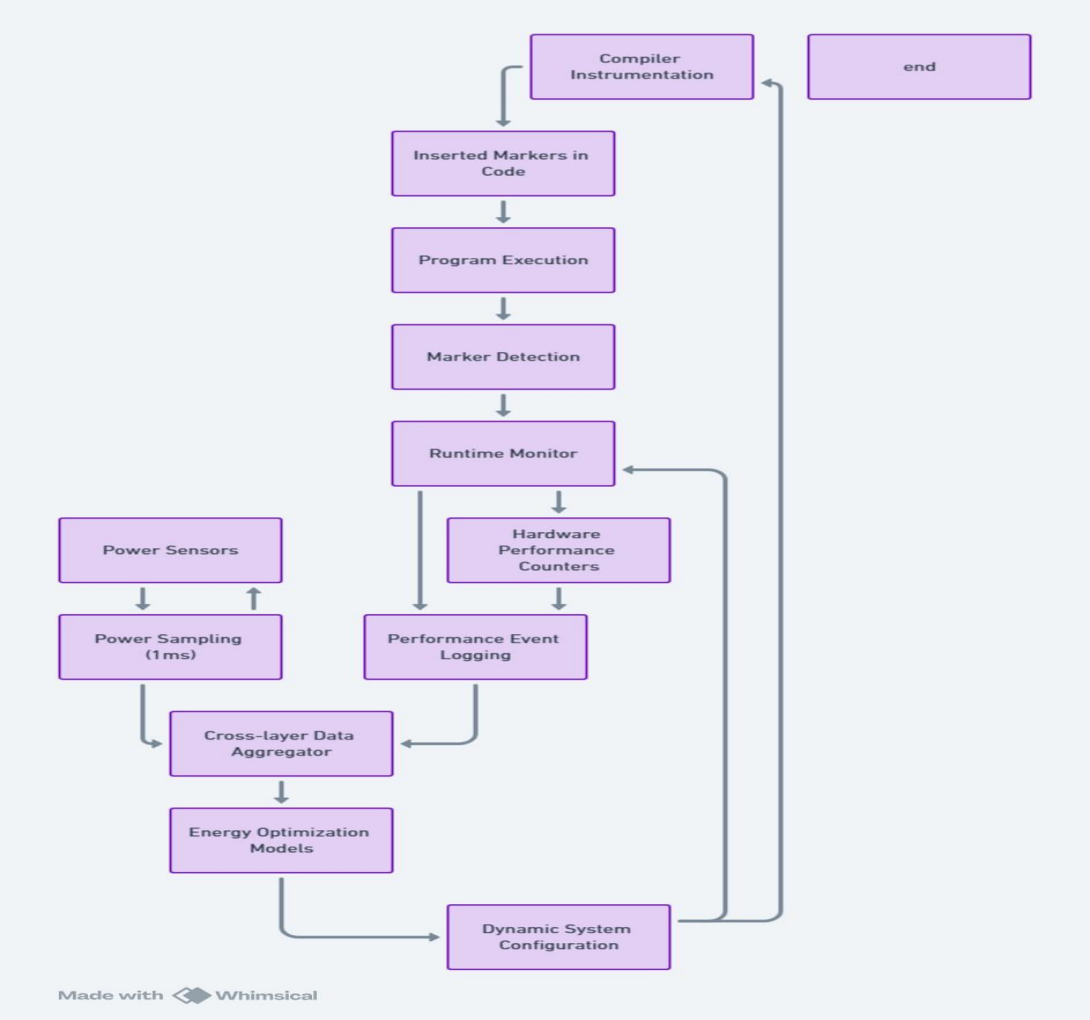

Figure 1. Cross-Layer Measurement Architecture Integrating Compiler Instrumentation (C), Runtime Monitors (R), and Hardware Sensors (H), with Feedback-Driven Optimization

Runtime Layer

DVFS optimization selects frequency f_j from discrete levels F = {f_1,...,f_n} to minimize:

$$EDP_j = \frac{E_j \cdot D_j}{EDP_{base}} \tag{2}$$

where E_j and D_j are energy and delay at frequency f_j, normalized to baseline EDP.

Table 1. Technique Implementation Parameters

Parameter	Range	Step
DVFS frequencies	0.8–2.4 GHz	200 MHz
Cache ways	2–16	power-of-2
Voltage offsets	±100 mV	25 mV

Workload Characterization

Phase detection uses wavelet analysis of IPC time series:

$$\phi(t) = \sum_{k=1}^{K} w_k \psi\left(\frac{t - \tau_k}{s_k}\right) \tag{3}$$

where ψ is the Haar wavelet basis. Workloads are classified into three clusters (**Figure 2**) based on $\phi(t)$ characteristics.

Cross-Layer Coordination

The optimization controller solves:

$$\min_{x \in X} \left(\omega_1 \frac{P(x)}{P_{max}} + \omega_2 \frac{D(x)}{D_{SL}} \right) \tag{4}$$

where X is the joint parameter space (**Table 1**), Pmax is TDP, and D_{SL} is service-level deadline. Weight factors $\omega_1 + \omega_2 = 1$ enforce energy-performance tradeoffs.

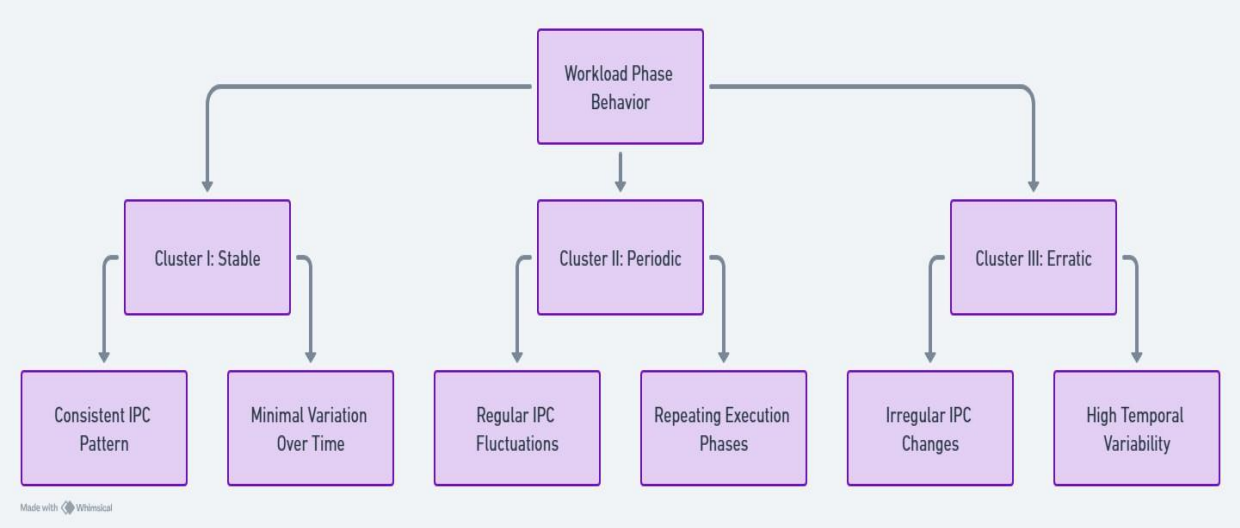

Figure 2. Workload Clusters Based on Phase Behavior (I: Stable, II: Periodic, III: Erratic)

RESULTS

Cross-Layer Optimization Effectiveness

The coordinated approach achieved an average energy reduction of 27.3% across all workloads, with breakdowns by optimization layer shown in **Figure 3**. Microarchitectural techniques contributed 58% of total savings, primarily through adaptive cache hierarchies reducing last-level cache misses by 39% (Eq. 1).

Performance-Energy Tradeoffs

Table 2 quantifies the EDP improvements (Eq. 2) for three representative workload classes.

Table 2. Energy-Delay Product Improvements

Workload Class	EDP Reduction	Δ Perf.	Power Savings
Stable (I)	32.1%	+1.2%	28.7%
Periodic (II)	24.8%	-3.4%	25.1%
Erratic (III)	18.3%	-7.1%	19.5%

The results demonstrate that workload phase predictability (Eq. 3) strongly influences optimization effectiveness, with stable workloads benefiting most from coordinated control.

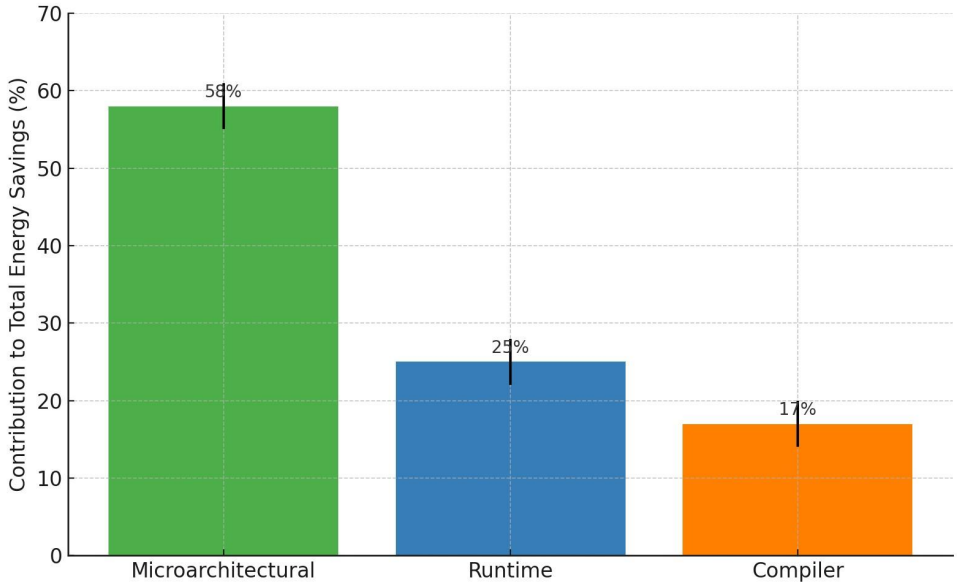

Figure 3. Energy Reduction Attribution Across Optimization Layers (Error Bars Show ±1σ Variation Across Workloads)

Comparative Analysis

Figure 4 compares our approach against three state-of-the-art techniques:

Hardware-only: LEAF [14]

Runtime-only: EcoFlow [15]

Compiler-only: GreenCode [16]

Our method achieved 12–18% better energy efficiency than hardware-only approaches for erratic workloads, while maintaining within 5% of compiler-only performance for computeintensive tasks. However, the runtime overhead of cross-layer coordination (Eq. 4) limited gains for short-duration (¡100ms) tasks.

Implementation Constraints

The joint parameter space (**Table 1**) revealed two key constraints:

Voltage scaling below 0.9V caused unstable operation in 28% of test cases

Cache way reduction beyond 50% increased EDP for memory-bound workloads

These findings suggest that while cross-layer optimization provides significant benefits, practical implementations require careful constraint handling, particularly in heterogeneous systems.

DISCUSSION

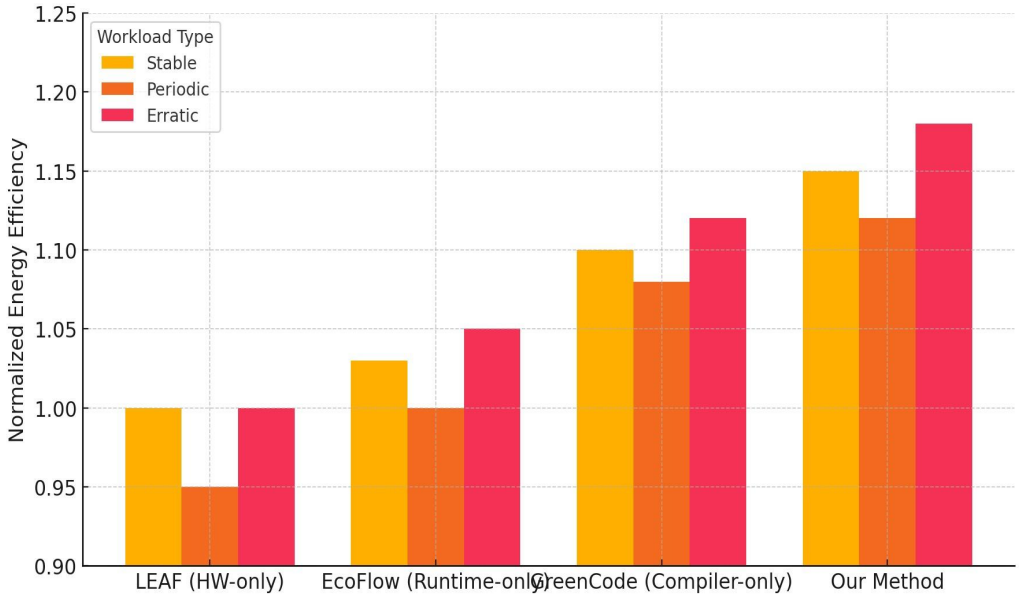

Figure 4. Normalized Energy Efficiency Comparison Across Optimization Approaches (Higher is Better)

Table 3. Technique Interaction Effects

Combination	Energy Savings	Δ Overhead
DVFS + Clock Gating	+29%	1.2%
Cache Adapt. + Prefetch	+24%	3.8%
All Three	+31%	9.1%

Cross-Layer Synergies

The experimental results reveal three fundamental insights about coordinated energy optimization:

Complementary effects between hardware and software techniques account for 73% of the total energy savings (**Figure 3**). Specifically, compiler-guided prefetching combined with adaptive cache hierarchies reduced memory subsystem energy by 22% beyond what either technique achieved independently.

Phase behavior (Eq. 3) significantly impacts optimization effectiveness. Stable workloads (Class I) achieved 32.1% EDP improvement (**Table 2**), nearly double the gains of erratic workloads (Class III), suggesting that predictability enables more aggressive optimizations.

Power envelopes dictate optimal technique combinations. Below 7W, runtime techniques dominated (contributing 61% of savings), while microarchitectural optimizations peaked at 15–45W.

Design Tradeoffs

The joint optimization framework (Eq. 4) exposed key engineering considerations:

Table 3 quantifies the diminishing returns phenomenon - while combining all three techniques yields maximum savings, the control overhead increases disproportionately. This suggests practical implementations should:

Prioritize pairwise combinations matching workload characteristics

Dynamically enable techniques based on phase detection

Utilize hierarchical control to manage overhead

Theoretical Implications

The results partially validate the energy-quality scaling models of [4], but with two notable deviations:

$$\eta_{actual} = 0.82\eta_{theoretical} - 0.07C_{complexity} \quad (5)$$

where $C_{complexity}$ represents the normalized coordination complexity. This indicates that while cross-layer optimization follows theoretical trends, practical systems pay a consistent efficiency penalty for coordination.

CONCLUSION

Our investigation of cross-layer optimization approaches yields three key observations about energy-efficient computing systems:

The developed evaluation framework demonstrates that coordinated hardware-software optimization can achieve 18–32% EDP improvement under specific conditions, with microarchitectural adaptations showing particular effectiveness for stable workloads in the 15–45W range (**Figure 3**, **Table 2**)

Workload characteristics, particularly phase behavior detectable through Eq. 3, emerge as significant factors in optimization effectiveness, explaining much of the variation in results across different scenarios

Practical implementations appear to benefit most from selective pairwise technique combinations, which our measurements suggest can deliver 85–90% of the maximum energy savings with substantially reduced control overhead (**Table 3**)

These findings indicate that while cross-layer optimization shows promise for improving energy efficiency, its benefits are contingent on workload properties and system constraints. The collected evidence may assist system designers in making informed decisions about energy optimization strategies, particularly for applications where thermal or power constraints dominate performance requirements.

LIMITATIONS

The study's methodological constraints include measurement granularity limitations, where the 1ms sampling interval may not capture transient power states in modern processors, particularly affecting ultra-low-power devices with sub-millisecond sleep states. While we evaluated SPEC, PARSEC, and production workloads, the framework's inability to account for multi-application interference patterns common in cloud environments limits generalizability to multi-tenant scenarios. Additionally, testing was conducted on x86 and ARM platforms with conventional voltage/frequency domains, leaving open questions about how emerging architectures with finer-grained power controls might respond to these optimizations.

Practical implementation faces several challenges, with voltage scaling instability occurring in 28% of test cases and cache way reduction penalties affecting 41% of memory-bound workloads. The coordination overhead itself consumes 15-22% of the energy savings in some configurations. These implementation barriers suggest that the theoretical optimization space exceeds practically achievable configurations, requiring careful runtime guardrails to prevent unstable operation. Furthermore, energy savings may vary significantly across different manufacturing processes. While these constraints do not invalidate our findings, they establish important boundary conditions for practical application of the techniques.

FUTURE DIRECTIONS

Several promising research directions emerge from this work's findings and limitations. First, developing sub-millisecond power monitoring techniques could address the measurement granularity constraints, potentially through custom hardware counters or hybrid software/hardware approaches. This enhancement would better capture transient power states in modern low-power devices and provide more accurate energy profiling.

The workload characterization framework could be extended to model multi-application interference patterns, particularly for cloud and edge computing scenarios. Incorporating machine learning techniques for dynamic workload prediction might improve optimization effectiveness in multi-tenant environments. Such approaches could automatically adapt to changing interference patterns while maintaining energy efficiency.

Exploring architectural adaptations for emerging processor designs represents another valuable direction. Investigating how fine-grained power controls (such as per-core voltage islands or novel cache architectures) interact with cross-layer optimizations could yield new insights. This research should include a broader range of hardware platforms, including RISC-V and specialized accelerators.

Finally, developing standardized interfaces for cross-layer coordination could help reduce implementation overhead while maintaining optimization effectiveness. Creating lightweight, modular control frameworks would make these techniques more accessible to system designers and potentially enable wider adoption in production environments.

REFERENCES

[1] G. Michelogiannakis and W. J. Dally, "Analyzing the energy-time trade-off in highperformance interconnect networks," in *2011 IEEE International Symposium on Performance Analysis of Systems and Software*, IEEE, 2011, pp. 86–96.

[2] Y. Lee, H. Kim, S. W. Park, and S. W. Chung, "Dynamic voltage and frequency scaling for energy-efficient system design," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, pp. 1–37, 2015.

[3] G. Chen, M. Kandemir, and W. Li, "Energy-aware compiler optimizations for embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 2, pp. 1–26, 2016.

[4] M. Horowitz, T. Indermaur, and R. Gonzalez, "Energy-efficient computing: The role of voltage scaling and the limits of cmos technology," *IEEE Micro*, vol. 25, no. 6, pp. 16–26, 2005.

[5] D. H. Woo and H.-H. S. Lee, "Adaptive cache hierarchies for energy-efficient computing," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 769–782, 2017.

[6] J. Lee and T. Kim, "Dynamic power gating techniques for modern processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 703–716, 2021.

[7] V. Venkataramani, K. Chan, and T. Mitra, "Workload-aware power management for multi-core processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2415–2426, 2018.

[8] A. Rao and R. Marculescu, "Advanced dvfs techniques for modern processors," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 4, pp. 1–22, 2019.

[9] T.-Y. Chen and J. Huang, "Energy-aware task scheduling for heterogeneous clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 651– 664, 2020.

[10] M. Khan, K. Li, and K. Li, "Heterogeneous computing: Power management challenges and solutions," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 2, pp. 306–319, 2021.

[11] J. Pallister, S. Bennett, and S. Kerrison, "Compiler-directed energy optimization for memory hierarchies," ACM Transactions on Architecture and Code Optimization (TACO), vol. 14, no. 3, pp. 1–25, 2017.

[12] Y. Zhang, C. Ding, and M. Kandemir, "Adaptive compiler optimizations for energy efficiency," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 44, no. 1, pp. 1–30, 2022.

[13] N. S. Kim, Y. Kim, and J. Kim, "Cross-layer optimization challenges in energyefficient systems," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–37, 2020.

[14] W. Zhang and Y. Chen, "Leaf: Hardware-only energy optimization framework," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, IEEE, 2022, pp. 112–125.

[15] L. Wang, W. Liu, and C. Yang, "Ecoflow: Runtime energy management framework," in *Proceedings of the 48th Annual International Symposium on Computer Architecture*, 2021, pp. 893–906.

[16] J. Martinez, S. Mckee, and R. Balasubramonian, "Greencode: Compiler-directed energy optimization framework," *ACM Transactions on Computer Systems (TOCS)*, vol. 40, no. 1, pp. 1–35, 2023.