F-LEMMA: Fast Learning-based Energy Management for Multi-/Many-core Processors

An Zou*[1], *Member, IEEE,* Yehan Ma*[1], *Member, IEEE,* Karthik Garimella*[2], *Member, IEEE,*
Benjamin Lee[3], *Member, IEEE,* Christopher D. Gill[4], *Senior Member, IEEE,* and Xuan Zhang[4], *Member, IEEE.*
[1]Shanghai Jiao Tong University, [2]New York University,
[3]University of Pennsylvania, [4]Washington University in St. Louis

***Abstract*—Over the last two decades, as microprocessors have evolved to achieve higher computational performance, their power density has also increased at an accelerated rate. Improving energy efficiency and reducing power consumption are therefore critically important to modern computing systems. One effective technique for improving energy efficiency is dynamic voltage and frequency scaling (DVFS). With the emergence of integrated voltage regulators, the speed of DVFS can reach microsecond (µs) timescales. However, a practical and effective strategy to guide fast DVFS remains a challenge. In this paper, we propose F-LEMMA: a fast, learning-based, hierarchical DVFS framework consisting of a global power allocator in the kernel space, a reinforcement learning-based power management scheme at the architecture level, and a swift controller at the digital circuit level. This hierarchical approach leverages computation at the system and architecture levels with the short response time of the swift controller to achieve effective and rapid µs-level power management supported by the integrated voltage regulator. Our experimental results demonstrate that F-LEMMA can achieve significant energy-savings (35.2%) across a broad range of workloads. Conservatively compared with existing state-of-the-art DVFS-based power management schemes that can only operate at millisecond timescales, F-LEMMA can provide notable (up to 11%) Energy-Delay Product (EDP) improvements across benchmarks. Compared with state-of-the-art non-learning-based power management, our method has a universally positive effect on evaluated benchmarks, proving its adaptability.**

I. Introduction

Multi-/many-core processors have become the mainstream computing workhorses for both general-purpose and embedded systems. With the demise of Dennard scaling [1], [2] and the increasing level of integration of digital logic on a single die, high power density has become a key design constraint and performance-limiting bottleneck for future generations of computing systems. Dynamic power management (DPM) techniques, such as dynamic voltage and frequency scaling (DVFS) and power gating, are widely used in state-of-the-art processor systems to save power and improve energy efficiency. For example, Intel's Enhanced Intel SpeedStep Technology (EIST) [3], AMD's PowerNow! [4], ARM's Intelligent Energy Controller (IEC) [5] and NVIDIA's Power Management Mode [6] provide utilities to allow the voltage and frequency (clock speed) of the processor to be dynamically changed to different power states by software. This capability allows the processor to meet the instantaneous performance demands from diverse computational workloads while minimizing power consumption and heat generation. In a typical setting, voltage and frequency are decreased as the processor enters an idle stage and increased as it enters an active stage.

Seeking a more effective power management strategy, many adaptive solutions have been explored recently by leveraging control theory and machine learning approaches. In these adaptive power management schemes, the control/learning agent can monitor the workload status at run-time and adjust the voltage and frequency settings according to its online estimation model [7], [8], [9], [10]. In conventional power delivery systems for multi-core and many-core processors, a cluster of cores (or even all the cores) may reside in one voltage domain and share one voltage rail from an off-chip voltage regulator. Due to the long physical distance and associated parasitic loading effect, the voltage transition time of an off-chip voltage regulator generally exceeds a millisecond, which fundamentally limits how quickly the power management settings can be adjusted in response to transient workload events that can happen in several microseconds. Although integrated voltage regulation provides much finer spatial (per-core) and temporal (tens to hundreds of nanoseconds) granularity in supply voltage allocation and delivery [11], [12], a practical and effective method to realize adaptive power management at microsecond timescales and take advantage of such fast integrated voltage and frequency scaling ability, is still needed. Meanwhile, as the computational complexity of the control and machine learning algorithms and their execution costs in software increase, the latency and response time of many adaptive power management schemes cannot be readily scaled to meet the demands of microsecond-level DVFS.

In this paper, we propose F-LEMMA, a fast learning-based voltage and frequency scaling approach for energy-efficient multi-core and many-core processors. To reap the previously unattainable benefits of microsecond timescale power management, we propose a hierarchical learning-based approach. This hierarchical power management approach has three layers: a global controller works as the kernel space interface to a userspace energy and power management methodology; an intermediate learning-based controller takes in the architectural information and utilizes a reinforcement learning agent to update the configuration of a lower-level swift controller; finally, the swift controller uses a fast linear classifier to generate voltage and frequency pairs for each core at the microsecond timescale. We validate the proposed F-LEMMA

*Authors contributed equally to this research.

approach, under different configurations and using several benchmark applications, and compare it with previous related work. Our experimental results show that F-LEMMA achieves a 35.2% energy savings on average across a wide range of benchmarks. Compared with state-of-the-art power management at millisecond timescales, the microsecond-level fast power management in F-LEMMA saves significant amounts of energy with only minimal performance loss.

This paper makes the following contributions to the state-of-the-art in power and energy management:

- An illustration of the potential benefits of microsecond timescale per-core DVFS and a comparison study of integrated voltage regulators and power delivery systems supporting this fast DVFS.
- A hierarchical power management strategy, including a global controller as the interface to the operating system, a learning controller at the architecture layer, and a swift controller at the circuit layer. This architecture provides adaptive, microsecond timescale, per-core, fast DVFS.
- A quantitative study methodology proposed and applied to F-LEMMA with OpenMP synthetic benchmarks. F-LEMMA power management achieves over 90% of the ideal DVFS and the learning-based program phase prediction is critical to the power management.
- An evaluation of the run-time adaptive hierarchical power management approach, and an implementation of its learning controller with High-Level Synthesis (HLS).
- A comprehensive experimental study of the proposed F-LEMMA approach, which demonstrates extra energy savings from fast power management. The evaluation includes comparisons to previous related work, ablation studies of different layers, and assessments of performance with different system configurations and scales.

II. Background

A. Dynamic Voltage Frequency Scaling (DVFS)

Dynamic voltage and frequency scaling (DVFS) is a technique to manage processor power consumption. Run-time dynamic power has a squared and linear relationship with frequency and voltage ($P_{\text{dynamic}} \sim CV^2f$), respectively, whereas static power has a relationship with voltage ($P_{\text{static}} \sim VN_{tr}I_{\text{static}}$) where N_{tr} is the number of transistors and $I_{\text{static}}(V)$ is the normalized static current for each transistor, which also depends on supply voltage.

Effective DVFS for multi-core processors requires multiple voltage domains. The circuitry within one voltage domain shares a common voltage rail, hence opportunities to reduce the domain's voltage are limited by the unit that needs the highest supply voltage. Voltage levels are scaled in fixed, discrete steps and are typically selected using tables that map frequency to voltage. Voltage and frequency scaling is based on the application's performance requirements. For example, when one core is waiting for synchronization, its voltage and frequency are reduced to save power and energy.

B. Adpative Power Management

In recent years, as the workloads in multi-core and many-core systems have become more diverse and variable, adaptive power management has replaced previous fixed models. To achieve effective power management, workloads are predicted at run-time using adaptive models. There are two general strategies. On one hand, control theoretic mechanisms, such as Kalman filters [13] and model predictive control [14], use dynamically updated models to scale voltage and frequency under power or performance constraints. On the other hand, learning mechanisms predict application phases and control decisions without knowing an accurate workload model in advance [15], [16]. With reinforcement learning, an agent learns to act optimally in an environment by evaluating and selecting actions that optimize for desired rewards. Reinforcement learning can be adapted for power management by training a per-core DVFS agent that selects the appropriate voltage and frequency levels by observing system conditions [7]. Because both the adaptive control and learning algorithms are relatively complex with considerable execution time, such adaptive power management can operate only at low frequencies. This problem can be mitigated by introducing a hierarchical design in which adaptive power management techniques are deployed at the software level and supply information to fast controllers.

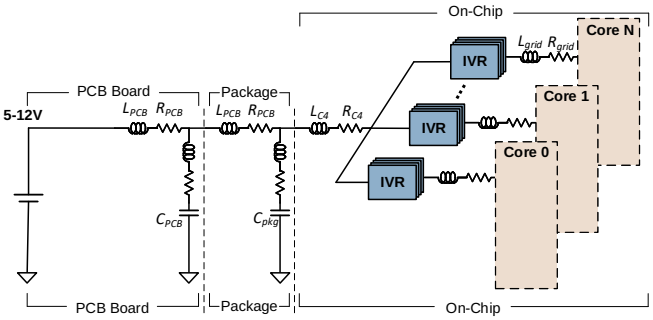


Fig. 1: The integrated voltage regulator based power delivery system.

C. Integrated Voltage Regulators

In a conventional power delivery system for multi-core or even many-core processors, cores share a common voltage rail and a centralized voltage regulator is located off-chip to step down the supply voltage from the PCB board level (5-12V) to the core level (0.8-2V). Because the off-chip voltage regulator uses large inductors and capacitors, together with the board-level decoupling capacitors and prominent parasitic inductance, there is an unavoidable long transition time (rise time and fall time) before the voltage reaches a desired level. It limits the dynamic voltage and frequency scaling in processors with off-chip VRM based power delivery systems to millisecond timescales.

Emerging power delivery systems use integrated voltage regulators, moving the step-down voltage regulator on-chip, as shown in Fig. 1. Integrated regulator design strives to reduce the size of inductors and capacitors to a small on-die area. One prominent side effect of this design strategy is pushing the switching frequency from tens to hundreds of MHz. Such a higher switching frequency incurs significant switching losses and degrades conversion efficiency.

The integrated voltage regulator naturally has a much shorter transition time than conventional off-chip voltage regulators. This advantage comes from smaller inductors and capacitors, faster switching, and reduced parasitic inductance thanks to its closer location to the core. Measured results from prototype silicon chips [17], [18], [19], [20] suggest that power delivery with integrated regulators can easily switch between voltage levels at tens to hundreds of nanosecond timescales. As the integrated on-chip regulators can have a distributed configuration with rare overhead, it naturally support multiple, flexible voltage domains, which would incur expensive design overhead when using off-chip regulators. Therefore, integrated voltage regulators permit fast, per-core power management which was previously unattainable.

III. Methodology

In this section, we first reveal the potential benefits of microsecond timescale per-core DVFS and compare the integrated voltage regulator and power delivery system designs to study the possible speeds of the fast DVFS. Then we introduce the proposed fast hierarchical learning-based power management strategy with the global controller as the interface to users, the learning controller at the architecture level, and the swift controller at the level of digital circuits.

Fig. 2 illustrates the potential benefits of microsecond-level power management. Here, the power consumption of a core is shown in black lines and its throughput (instruction per cycle IPC) in blue lines during microsecond intervals for representative workloads on a simulated 16-core Intel Nehalem CPU processor. An interval with fewer instructions per cycle (IPC) transitions could be a candidate DVFS interval, in which the core can reduce the frequency and voltage to save power with only rare instances of performance degradation. In addition to intervals that have transitions at millisecond timescales, we find there exist many more transitions at microsecond timescales, exhibiting distinctive traits. First, such transitions often appear irregularly within the workloads. For example, the transitions indicated in Fig. 2 (a) are occasional power and activity peaks and valleys in the power-light "FFT" benchmark, providing opportunities to apply DVFS to lower the voltage and frequency during the low-activity period without incurring performance loss. Secondly, transitions arise from interactions among threads. In running the power-hungry "Radix" benchmark, core 1 and core 4 (in Fig. 2 (b) and (d)) are in synchronization stalls, waiting for core 2 (in Fig. 2 (c)). Thirdly, transitions occur from periodic power and activity within a workload, and between the completion of one workload and the start of another. Fig. 2 (e) shows the periodic power and activity in the "Water" benchmark. In addition to computation in user space, the majority of request service times in kernel space require less than 250 microseconds, even with millisecond tail latencies [21]. Based on observations from benchmark executions on architecture simulators, and program execution patterns (under fast DVFS scenarios) discussed in related work [11], [22], typical DVFS opportunities generally fall into two classes. One originates from periodic or occasional execution

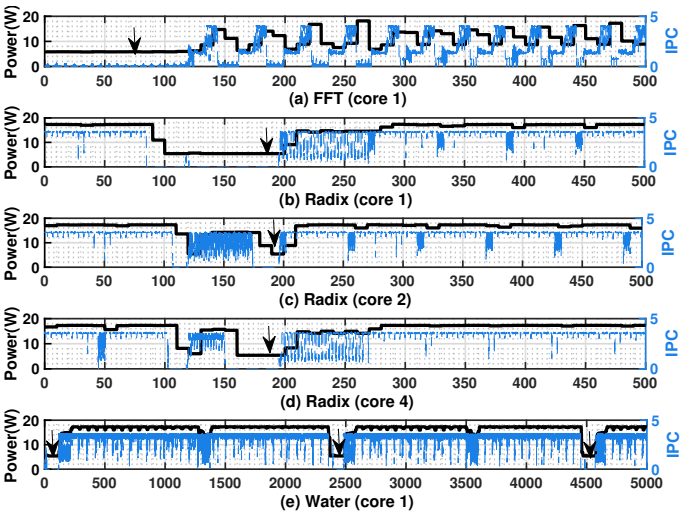


Fig. 2: Power and throughput traces in many-core processors.

intervals with low computation and memory intensity, and the other can be attributed to stalls from synchronization, thread scheduling, periodic activities and so on. Since conventional power delivery systems with off-chip voltage regulators can only support millisecond voltage scaling, many energy-saving opportunities are lost. In contrast, integrated voltage regulators can adjust voltages within microseconds and offer flexible per-core implementation, thus opening the door for fast and adaptive power management at the system level.

A. Power Delivery System for Fast DVFS

As the first step in building the foundation for our hierarchical power management approach with online learning, we explore state-of-the-art power delivery systems designed to enable fast, per-core DVFS. In conventional power delivery systems that use off-chip voltage regulators, a buck converter is deployed for its high efficiency across a wide input and output range. However, it requires more than 10 microseconds to scale voltage, due to passive components like inductors and capacitors in off-chip voltage regulators, parasitic inductance along the power delivery networks, and bloated decoupling capacitance at the PCB board and package levels.

Recent technology advances make it possible for switching regulators to operate at much higher frequencies. At higher switching frequencies, the passive components can be much smaller and integrated on the same die as processors. Given these advantages, integrated voltage regulators have been adopted in both academic prototypes and industrial and commercial processors. Although IVRs have a slightly lower voltage conversion efficiency than off-chip voltage regulators, they enjoy lower supply voltage noise, which compensates for the voltage conversion loss. Most importantly, the IVR naturally has a much shorter transition time because of smaller passive components, reduced parasitics, and avoidance of PCB and package-decoupling capacitance.

As the starting point for exploring hierarchical fast in-integrated voltage and frequency scaling for energy-efficient multi-/many-core processors, we begin with the power delivery systems that determine the possible DVFS speeds. To maximize the versatility of the proposed hierarchical learning-based

TABLE I: Summary of design space explorations of 16-phase buck IVRs.

DVFS Speed	**1μs**	**2μs**	**4μs**	**8μs**	**16μs**
Efficiency (%)	79.1	80.6	82.8	82.8	82.8
Switch Freq. (MHz)	146	119	60	60	60
L per-phase (nH)	0.188	0.188	0.75	0.75	0.75
C per-phase (μF)	0.281	0.422	0.422	0.422	0.422
Area (mm^2)	92	137	142	142	142

power management approach, we choose mainstream two-stage heterogeneous power delivery systems with both off-chip and on-chip integrated buck voltage regulators. A buck-based two-stage heterogeneous power delivery system will represent mainstream power delivery systems with integrated voltage regulators because it offers high power delivery efficiency and flexible, fast voltage scaling [23], [24].

Alternatives for the on-chip regulator suffer from several limitations. A switched-capacitor voltage regulator has a fixed conversion ratio and is hard to support fine-grained voltage scaling with multiple voltage levels. A low drop out (LDO) voltage regulator offers fast voltage scaling, but its power conversion efficiency is determined by the ratio of output to input voltages. As voltage and frequency scale down, the conversion losses in an LDO more than offset any power and energy savings in the processor. Customized reconfigurations of IVR-based power delivery systems are studied in [25], but they lack the needed versatility.

Having decided to use heterogeneous power delivery systems with both off- and on-chip integrated buck voltage regulators, we proceed to determine the proper DVFS speeds. As we discussed before, passive components (like inductors and capacitors) in integrated voltage regulators and power delivery networks limit the voltage transition speed. For a heterogeneous power delivery system, we use the *Ivory* open-source integrated voltage regulator modeling tool [26] and power delivery networks for manycore systems [27] to explore the design spaces of IVRs (maximum area budget for IVR of each core is 150 mm^2 where the lowest power density is 0.1 W/mm^2) in heterogeneous power delivery systems that can support different fast DVFS. The loads are the processor cores described in Section VI-A. Here, we set the voltage scaling rise time to within 0.5% of the DVFS interval durations [28], [29], [23], [30] and the voltage overshoot to less than 5%. The key design parameters for IVRs that support different DVFS speeds are summarized in Table I. When the DVFS speeds are faster than 4 μs, the DVFS speed is one of the constraints of IVR design. When supporting faster DVFS, IVR designs keep reducing the size of on-die inductors and capacitors to achieve a faster voltage transition, and one prominent side effect is pushing the switching frequency from tens to hundreds of MHz. Higher frequency switching comes at a cost of degrading the conversion efficiency of the IVRs as the switching loss becomes more significant. When the DVFS speeds are slower than 4 μs, the DVFS speed is not a constraint on IVR design, which means the optimal IVR that targets high efficiency can routinely support DVFS speeds no faster than 4 μs.

Unlike conventional millisecond timescale power management, at microsecond timescales, the DVFS controller has limited computational ability, and only simple arithmetic operations can be applied to control the DVFS. Therefore, in this DVFS strategy we draw lessons from using run-time throughput (or workload) to adjust frequency and voltage [31]. Fig. 3 shows the normalized energy consumption of throughput-guided DVFS (measured in IPC, instructions per cycle) at different microsecond timescales of the system described in Section VI. If the run-time IPC at the DVFS interval is larger than 0.8 times the average run-time IPC, the voltage and frequency will increase by a level. If the run-time IPC at the DVFS interval is smaller than 0.6 times the average run-time IPC, the voltage and frequency will decrease by a level. From the experimental results, we can see that microsecond level fast DVFS based on throughput IPC can save more energy when the DVFS runs faster. However, limited by the computational ability at the microsecond timescale, the fast DVFS cannot be effective on all the benchmarks, e.g., benchmark radix, lu.cont, cholesky, blackscholes, and cg.

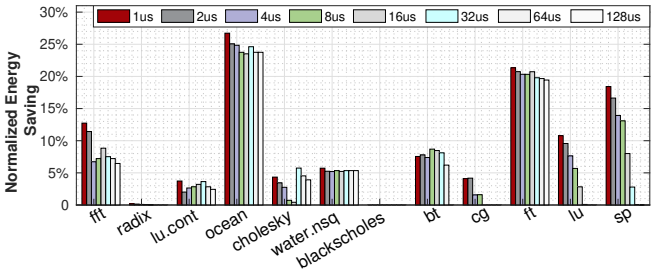


Fig. 3: Normalized energy consumption of throughput (IPC) guided DVFS at different microsecond timescales.

B. Hierarchical Power Management Framework

Conventional DVFS control algorithms can be implemented in the processor microarchitecture, in the scheduler, or through compiler algorithms [32], [33]. Most prior research in DVFS control has been implemented in the operating system with coarse temporal granularity, a sensible approach when off-chip regulators have slow response times and voltages change on the order of several milliseconds. Integrated voltage regulators enable more responsive DVFS, saving power and energy at microsecond granularity, but effective mechanisms are required to guide such fine-grained DVFS. Directly increasing the execution frequency of previous conventional DVFS control algorithms is not applicable, not only because it is hard to finish the computation within microseconds but also because the start up overhead (such as scheduling the thread to run the DVFS algorithm) already takes more than microseconds.

To guide microsecond timescale fast DVFS effectively within computational constraints, we propose a hierarchical DVFS management that implements three control layers, as shown in Fig. 4. First, a global controller in kernel space specifies a power budget and energy performance weights. This global controller also works as the interface with computer users, who can use their own power budget and energy performance weights, based on the applications they are running. Next, a per-core learning controller in the architectural layer is implemented with reinforcement learning to pass the refined run-time architectural information to a swift controller. Finally, the swift controller then makes decisions based on

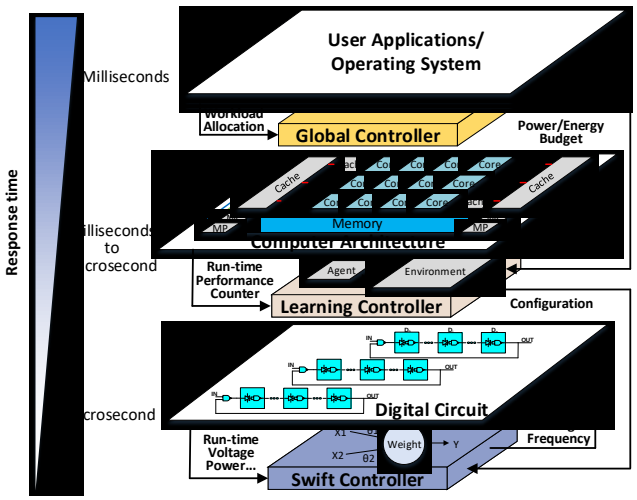


Fig. 4: Microsecond-Level hierarchical fast power management (DVFS) for multi-core and many-core processors.

the refined architectural information and run-time power and performance. This hierarchical layered approach not only acts rapidly at the swift controller frequency, but also adapts at the learning controller frequency as the application progresses.

C. Global Controller

The top controller in this hierarchical framework is the global controller. The global controller runs at the kernel level and provides a programmable interface for the users of a multi-/many-core system to adjust the power management features. The global controller can accept energy and performance weights and power budgets from user inputs that guide the learning controller as it navigates varied modes that favor battery life, favor performance, or balance the two. The global controller updates the reward function of the learning controller with the user inputs, as shown in Eq. 1.

$$R = -W_E \times \frac{\text{energy}}{\text{peak energy}} + W_I \times \frac{\text{IPC}}{\text{peak IPC}} - W_B \times \frac{|\text{power} - \text{budget}|}{\text{peak power}} \tag{1}$$

where W_E is the weight for energy, W_I is the weight for performance, and W_B is the weight for power budget. The energy is calculated at the learning controller frequency. By adjusting the weights in the reward functions, the corresponding power management features are selected.

D. Learning Controller

The learning controller, at the architectural layer, leverages reinforcement learning (RL) to help DVFS adapt to applications. RL is a subset of machine learning built upon a Markov Decision Process, which describes interactions between an agent and its environment (represented by states) over time. At each time step, the agent selects an action that changes the environment and thus the state, and receives a reward associated with that transition to the new state.

Table II lists the components of the RL model. Note that actions or elements of the state space can be either continuous or discrete.

The value functions describe the expected return for being in some state (or for taking an action in a state) when following policy π:

$$V(s) = \mathbb{E}_\pi \left[G | S_t = s \right] \tag{2}$$

TABLE II: RL terminology. (RL's goal is to find an optimal policy $\pi(a|s)^*$)

Terminology	Symbol
Action Space	$a \in A$
State Space	$s \in S$
Reward Function	$R \in R^1$
Return	$G = \sum_{t=0}^{k} R$
Policy	$\pi(a\|s)^*$
State Value Function	$V(s)$
State-Action Value Function	$Q(s,a)$

$$Q(s,a) = \mathbb{E}_\pi \left[G | S_t = s, A_t = a \right] \tag{3}$$

The agent's goal is to learn policy $\pi(a|s)^*$, which maps each state to an action that maximizes the expected return G over k time-steps, and future time-steps are discounted by factor γ.

Policy gradient methods, such as Actor-Critic, directly optimize the policy by approximating the policy π and value function (Q or V) using approximators such as neural networks [34]. The *actor* consumes the state and produces a probability distribution over the action space. The *critic* learns a real-valued number that approximates the value function, $V(s)$. The actor-critic network combines the benefits of both *value based reinforcement learning methods* which are more sample efficient and steady, and *policy-based methods* which are better for continuous and stochastic environments.

If the action space is discrete with size $|A|$, the final layer in the approximator network is a flattened vector of size $|A|$ that is passed through a softmax layer to produce a discrete probability distribution. If the action space is continuous (*e.g.*, $a \in [0,1]$), the final layer approximates a probability distribution by predicting its parameters. For example, a layer that approximates a normal distribution must predict the mean μ and variance σ of a, increasing the number of outputs required for the approximation function [35].

The per-core actor and critic's heads share a common neural network backbone. For each benchmark, a trajectory is built by saving rewards, actions, and states over 25 timesteps. Both the actor and critic are jointly optimized for each collected trajectory using loss functions and Adam optimizer. The actor optimizes for actions that produce a higher reward value (Eq. 3), and the critic learns to predict the expected return given the normalized input state. Thus, the critic aids in the training process by guiding the actor to optimize for a policy with an overall higher return over the sampled trajectories.

For power management, the environment is the processor core's activity, and the state space is defined by 19 normalized performance counters, including instruction throughput, branch prediction misses, cache misses, and reads as well as voltage levels. The core power is collected with on-chip power sensing circuits [36] designed for fine-grain power management at microseconds. To collect samples of the state space, each benchmark is run with a random DVFS policy that performs DVFS every 500 μs. The performance counter values are stored at each of the VF transitions. This process is repeated until roughly $1,000$ samples for each benchmark are collected. When a particular benchmark is studied, the corresponding stored samples are then used to normalize

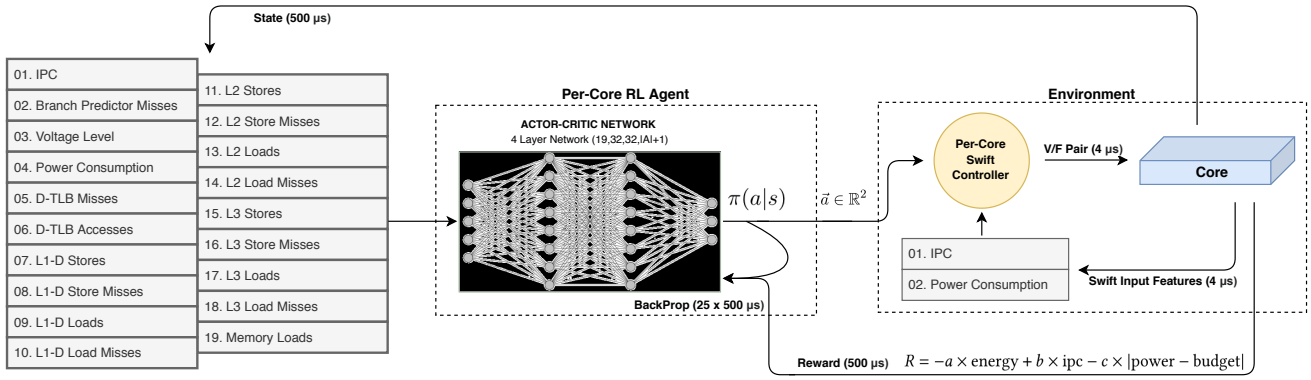


Fig. 5: Reinforcement learning and swift controllers.

1 Learning Controller (with Swift) ($\sim 500\mu s$)

Input: $N_{\text{cores}}, f(s;\theta_1), \cdots, f(s;\theta_{N-1}), \hat{s}_{\text{mean}}, \hat{s}_{\text{std}}$
$i \leftarrow 0$
while $(i < N_{\text{cores}})$ **do**
  $s \leftarrow$ get core state(i)
  $s \leftarrow (s - \hat{s}_{\text{mean}}) / \hat{s}_{\text{std}}$
  Forward propagation $\mu_{\text{policy}}, \sigma_{\text{policy}}, V(s) \leftarrow f(s;\theta_i)$
  Construct $\pi(a|s) \leftarrow \mathcal{N}(\mu_{\text{policy}}, \sigma_{\text{policy}})$
  Sample weights $\vec{w}_i \sim \pi(a|s)$
  Update swift controller $(i, \vec{w}_i)$
  $R \leftarrow$ observe reward(i)
  Store $\mu_{\text{policy}}, \sigma_{\text{policy}}, R, V(s)$
  $i \leftarrow i + 1$
end while

TABLE III: Action space of the actor neural network.

Experiment Type	Action Space
Without Swift Controller.	$a \in \{VF_1, \cdots, VF_4\}$
With Swift Controller	$\vec{a} \in [0,1]^2$

inputs to the learning controller. See Fig. 5 for details. The reward function is a linear combination of instruction throughput, energy, and the power budget determined by the global controller [37].

The learning controller can manage the DVFS settings either independently or in coordination with the swift controller at a lower level. During independent management, it directly maps the core's state to a voltage-frequency pair. During coordinated management, it sends an intermediate weight vector to the swift controller as described in Algorithm 1. Table III summarizes the action spaces of these two operations.

E. Swift Controller

The swift controller for each core is implemented at the digital circuit layer, managing its power and energy consumption by adjusting its voltage and frequency on microsecond timescales, which is supported by the integrated voltage regulator. First, the swift controller monitors current drawn by its core during each fine-grained monitoring interval (e.g., 100 ns in our study) to calculate power consumption. Second, it accesses hardware performance counters. These measurements together guide voltage and frequency settings at microseconds.

The swift controller uses a linear classifier as described in Eq. 4, where X is the input feature vector, W is the weight vector for the input feature, and b is the bias. When $f(X, W, b)$ is greater than threshold R_i, the swift controller sets voltage and frequency to V_i and F_i.

$$f(X, W, b) = WX + b \tag{4}$$

Operating at microsecond timescales, the linear classifier must be computationally simple yet effective. In this work, the classifier takes only two run-time parameters, power consumption and instruction throughput IPC, to define input $X = [P(t), IPC(t)]$. Depending on the workload phase, power and IPC have different roles in estimating system behavior. For example, suppose the fixed-point unit dissipates less power and the floating-point unit dissipates more power. As a workload performs a varying mix of fixed and floating-point operations, simply using power or instruction throughput alone cannot accurately classify the system behavior. Beyond the power and instruction throughput, we also consider and test other performance counters, such as cache hits and misses. At conventional millisecond timescales, these counters help improve the model's accuracy when estimating system dynamics. However, at the microsecond timescales we consider, these counters exhibit rapid and large fluctuations that can cause the system to oscillate and fail to converge. Therefore, only two most directly related parameters, power consumption and instruction throughput IPC, are used in the swift controller to adjust the voltage and frequency within microseconds. Weight vector W is updated by the global and learning controllers according to user inputs and system run-time status. The updated weights help the swift controller capture diverse workload phases and variations adaptively.

To summarize, in this proposed hierarchical management strategy, the global and learning controllers perceive and predict the system status and the swift controller adjusts the voltage and frequency within microseconds based on predicted system status. All three controllers work together to perform effective power management at microsecond timescales.

IV. Study of DVFS Response Time with Synthetic Benchmarks

In this section, we study the response time for the hierarchical learning-based fast power management. We use

synthetic benchmarks with manually generated "ideal" DVFS opportunities, because in the "ideal" opportunities for all the reasonable reward functions, the voltage and frequency should be immediately reduced to the lowest levels, with no performance loss. By comparing the behaviors of the DVFS controller with the ideal oracle strategy where the voltage and frequency should be set to the lowest level without performance loss, we can quantitatively describe the distance between the proposed F-LEMMA DVFS controller and the ideal DVFS controllers at microsecond timescales and find out what contributes to and dominates any "less than ideal" mismatches. For these "non-ideal" DVFS opportunities, the degrees of voltage and frequency adjustments are evaluated with real benchmarks in Section V and Section VI.

To cover the two categories of fast DVFS opportunities (computation/memory intensity variations and long stalls in threads' activities), we generate benchmarks for computation, memory, and both in combination, based on OpenMP for multi-core and many-core systems. We manually create ideal opportunities for microsecond timescale DVFS by inserting microsecond timescale sleep (usleep) intervals between the operations. In benchmarks involving computation and memory, we use usleep to create DVFS opportunities by adjusting the computation and memory intensity. In the combination benchmark shown in Algorithm 2, we not only use the inner loop usleep to create DVFS opportunities by adjusting the computation and memory intensity, but we also use the outer loop usleep to emulate long stalls, such as for thread synchronization and scheduling. Meanwhile, switching between the computation and memory parts represents program phase changes, which usually involve long stalls.

2 Combination Benchmark Example

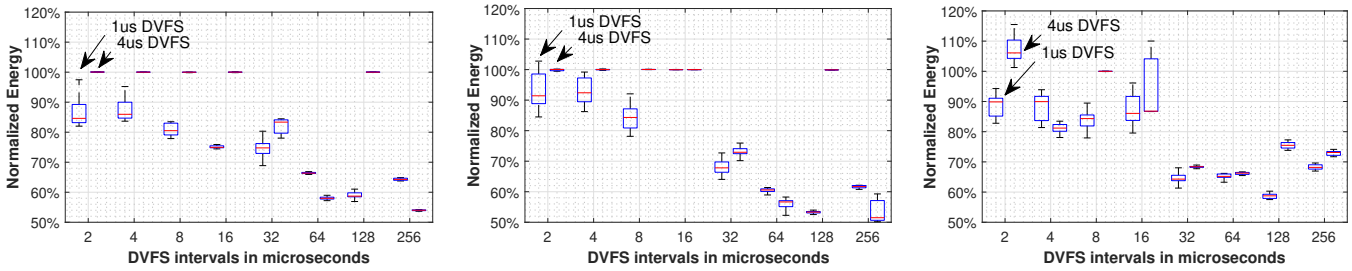
```
void main (int argc, char* argv[]){
int threads;
double x[length], y[length];
//OpenMP parallel execution
#pragma omp parallel
for(int i=0; i < threads_1; i++) {
  // The computation part:
  for(int j=0; j < computation_length; j++) {
    x[j] = (i+j)*0.5/(threads+0.1);
  }
  usleep(low_computation_interval);
  // The memory part:
  for(int j=0; j < memory_length; j++) {
    x[j] = y[j];
  }
  usleep(low_memory_interval);
  // The next computation/memory part .....
}
usleep(iteration_interval);
for(int i=0; i < threads_2; i++) {
  // .....
}
return;
```

In these synthetic benchmarks, the voltage and frequency can be reduced during the sleep intervals without performance loss. With these synthetic benchmarks, any power and performance patterns with a microsecond timescale resolution can be generated easily and their DVFS theoretical boundaries can be obtained. After applying F-LEMMA on these three synthetic benchmarks, we measure the energy saving and performance loss of F-LEMMA against the theoretically ideal DVFS strategy for each benchmark. We apply F-LEMMA with the swift controller at different speeds ($1\mu s$ and $4\mu s$) on benchmarks with different DVFS interval lengths. Fig. 6 shows the normalized energy consumption of F-LEMMA applied on the three synthetic benchmarks. Fig. 7 shows the normalized performance of F-LEMMA applied on the three synthetic benchmarks, within the ideal boundaries at 100%. The X axis shows the DVFS intervals, and the boxplots at each interval indicate the performance of F-LEMMA with swift controllers of different speeds.

Accurate detection and fast action, the most important aspects of the response time, are the results of cooperation between the learning controller and the swift controller. Accurate detection is mainly determined by how precisely the learning controller can tune the weights to capture the status of the program and processor for the swift controllers. Fast action is determined by how quickly the swift controller can detect DVFS intervals with the linear classifier and adjust voltage and frequency.

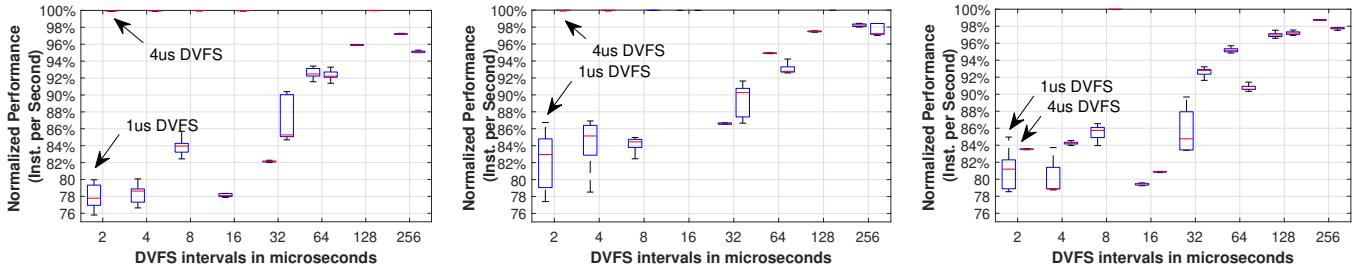
We follow a bottom-up approach and start with fast action given an accurate prediction. We choose synthetic benchmarks with either computation or memory operations, no program phase changes, and an accurate prediction that can be obtained after a long enough training process. We use the energy saving and performance loss under different swift controller speeds inside of each synthetic benchmark to evaluate the impacts of fast action on DVFS. In the separate computation and memory benchmarks shown in sub-figures (a) and (b) respectively in Fig. 6 and Fig. 7, when the swift controller operates at 1 μs, the voltage and frequency will be adjusted to save energy since the DVFS interval is over 2 μs. The swift controller operating at 4 μs can only save energy when the DVFS interval reaches 32 μs: if the swift controller operating at 4 μs is used to adjust voltage and frequency to save energy for intervals smaller than 32 μs, significant performance loss will be introduced. For both the swift controllers operating at 1μs and 4μs the wider distribution is more obvious when the DVFS interval is short and immediate voltage and frequency changes are not always possible. When the start or end of a DVFS interval is detected just before the swift controller's action, then the swift controller can act quickly and more energy will be saved. However, when the start or end of a DVFS interval is detected just after the swift controller's action, the voltage and frequency cannot be adjusted until the swift controller's next action. On one hand, delayed voltage and frequency reduction will cause less energy saving. On the other hand, if the swift controller is not able to immediately increase the voltage and frequency because of action time, performance loss may be introduced.

Next, we consider accurate prediction. We use the combina-



(a) Computation benchmark (b) Memory benchmark (c) Combination benchmark

Fig. 6: Study of the DVFS response time from energy saving.



(a) Computation benchmark (b) Memory benchmark (c) Combination benchmark

Fig. 7: Study of the DVFS response time from performance loss.

tion synthetic benchmark to test the prediction, shown in sub-figure (c) in Fig. 6 and Fig. 7. We choose the combination benchmark because it contains two representative program phases (computation intensive and memory intensive) and these two phases keep switching as the benchmark executes. In the separate computation and memory benchmarks shown in sub-figures (a) and (b) in Fig. 6 and Fig. 7, there is only one type of operation and no program phase changes. Therefore, the learning controller only needs to give out an accurate prediction after training. However, in the combination benchmarks, 100% accurate prediction is not seen, especially when the program phases change within the period of the learning controller. This is because the learning controller cannot update the prediction when the phase changes are faster than the prediction of the learning controller. The best prediction from the learning controller is a compromise considering all the phases. For example in those synthetic benchmarks, the learning controller needs to give a compromise prediction considering both computation and memory intensity. Previously in the separate computation and memory benchmarks, accurate prediction lets the 1μs and 4μs swift controllers adjust the voltage and frequency when the DVFS interval is longer than 2μs or 32μs respectively, which successfully saves energy. However, in the compromise prediction of the combination benchmark, the 4μs swift controller changes voltage and frequency even when the DVFS interval is only 2 μs which is even faster than the swift controller's speed. This means the changed voltage and frequency cannot catch up to the DVFS and the voltage and frequency should not be changed at all. Therefore, not only is performance loss introduced but also more energy is consumed with this compromise prediction.

To summarize, both fast action and accurate prediction are critical to hierarchical learning-based fast power management. Under an accurate prediction, the action speed determines the speeds of DVFS intervals for which the controller can keep up. However, because of limitations of the learning controller (learning rate given the computation size), it has to give a compromise prediction. This compromise prediction impacts DVFS effects, and on occasion may even cause extra energy consumption.

V. Online Learning and System Implementation

In this section, we first demonstrate the necessity of online learning control and then validate the functionality of learning controllers and test them with different reward functions. Then we use high-level synthesis (HLS) to implement the online learning controller and estimate its latency and power cost.

To demonstrate the necessity of online learning control, we examine the impacts of the input features (19 performance counters) on the output weights for IPC and power. We measure the average Pearson correlation coefficients between input features and output weights, where the reward function has IPC, energy, and power budget terms with the same weight during 100 epochs. Table IV shows the average Pearson correlation coefficients of the most power-light and power-hungry benchmarks, fft and radix, in the splash-2 benchmark set. From the results of the fft benchmark, the weight of the IPC performance indicator has a higher correlation coefficient with the input features than the one for power. Conversely, for the radix benchmark, the weight for power has a higher correlation coefficient than the weight for IPC. In the power-light fft application, the performance indicator is more critical in guiding DVFS. Not surprisingly, misses like TLB misses and cache misses have a negative correlation with the weight for IPC. Meanwhile, the correlation coefficients for the same input feature vary greatly across different benchmarks. Using an offline trained learning controller, it is hard to balance the variations across benchmarks and adapt to different benchmarks with a meaningful energy saving.

TABLE IV: Pearson correlation coefficients between input features and output weights

Benchmark	**fft**		**radix**	
Weight for	**IPC**	**power**	**IPC**	**power**
IPC	0.054	0.0161	0.0574	0.3794
Branch Pred. Misses	0.085	0.0035	0.0055	0.2174
Voltage Levels	-0.0251	0.0098	0.0017	-0.2725
Power Consumption	-0.0146	0.0123	-0.0232	-0.1653
D-TLB Misses	-0.0219	0.0190	-0.0249	0.3244
D-TLB Accesses	0.0026	0.0143	-0.0544	0.4350
Memory Loads	-0.0310	-0.0290	0.0185	0.2091
L1 D $ Stores	0.0032	0.0166	-0.0541	0.4381
L1 D $ Store Misses	-0.0223	0.0099	-0.0195	0.2272
L1 D $ Loads	0.0023	0.0126	-0.0550	0.4326
L1 D $ Load Misses	-0.0042	-0.0260	-0.0510	0.4955
L2 Stores	-0.0223	0.0099	-0.0195	0.2272
L2 Store Misses	-0.0223	0.0099	-0.0184	0.2335
L2 $ Loads	-0.0032	-0.0256	-0.0491	0.4999
L2 $ Load Misses	-0.0059	-0.0218	-0.0497	0.5032
L3 Stores	-0.0261	0.0098	-0.0189	0.2350
L3 Store Misses	-0.0243	0.0159	0.0051	0.2175
L3 $ Loads	-0.0073	-0.0217	-0.0497	0.5045
L3 $ Load Misses	-0.0097	-0.0234	0.0101	0.2259

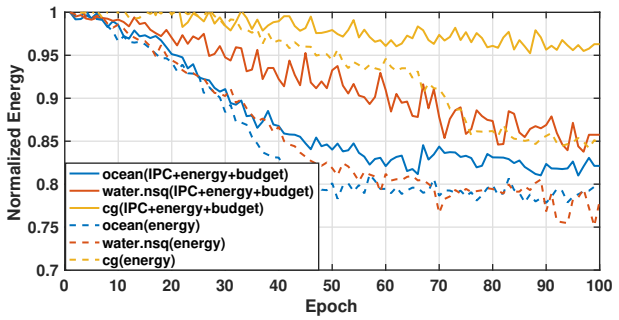


Fig. 8: Learning process under different reward functions.

Online-learning based power management, however, can adapt to these variations. Fig. 8 shows the convergence (learning) process for three representative benchmarks, where the reward function has IPC, energy, and power budget terms with the same weight. For comparison, the convergence process of the reward function with only the energy term is also shown, (where we set the weights for IPC and power budget to 0). In these experiments, each learning controller for each benchmark is started from randomly generated weights of the learning controller, meaning the learning controller starts from a random position. It shows that energy consumption in both scenarios is reduced as the benchmark progresses, which means the online learning controller can keep adjusting to adapt to the current benchmark and save more energy. Also as expected, the reward function with only the energy term makes the system energy consumption converge faster (fall further and even faster) over epochs. By adjusting the weights in the reward learning controller, the user can customize F-LEMMA to favor either performance or energy.

As shown in Table V, in our design the global controller runs in a kernel module, the learning controller executes on the neural network accelerator – an application-specific integrated circuit (ASIC), and the swift controller is implemented by digital logic gates. The learning controller executed on the

TABLE V: Implementation of hierarchical learning-based power management strategy

Controllers	**Implementation**	**Period**
Global controller	operating system kernel	10ms
Learning controller	neural network accelerator	500μs
Swift controller	digital logic gates	4μs

neural network accelerator is located close to the core. Compared with execution at the software level on general-purpose CPUs, execution in an ASIC increases both performance and energy efficiency. We experimentally compared the learning controllers using both software and ASIC designs. In software, the learning controller took over 40 microseconds (about $40\mu s/500\mu s$ = 8% performance overhead) to execute on a 2.3 GHz Dual-Core Intel Core i5 processor. To estimate the performance of the ASIC design, we implemented the learning controller using Vivado Vitis and the Design Compiler with TSMC 65nm Technology. The pipelined design is applied to optimize the performance. The learning controller takes 943 cycles to execute. It has a static power consumption of 0.12 mW and dynamic power consumption of 1.1 mW. The area consumption is 0.35741 mm^2. The swift controller has a static power consumption of 0.0039 mW and dynamic power consumption of 0.036 mW. The area consumption is 0.0013 mm^2. The overhead is accounted in the final evaluation. The swift controllers operate at microsecond timescales, and each swift controller operation has 2 fixed point multiplications, 1 addition, and up to 3 comparisons. The overheads from the swift controllers are negligible compared with those from the learning controller.

VI. Evaluation Results

In the section, we test F-LEMMA (the proposed hierarchical learning-based fast DVFS) via architecture-level performance and power simulators. We compare F-LEMMA with state-of-the-art power and management solutions and evaluate F-LEMMA with an ablation study and under different system configurations across real benchmarks.

A. Experimental Setup

We evaluate the proposed hierarchical learning-based power management scheme with experiments on an Intel Nehalem x86 processor architecture, which is detailed in Table VI. We use Sniper v7.3 [38] (with Mcpat [39]) to simulate the system performance and power (dynamic power and leakage power) for this multi-/many-core processor, generating runtime statistics with a fine granularity of 100 ns. We integrate both the `Numpy` and `PyTorch` packages with Sniper to implement the hierarchical learning design. Sniper performs timing simulations for multithreaded, shared-memory applications with tens to hundreds of cores, and has been validated for Intel Core2 and Nehalem systems. From the *parsec*, *splash2*, and *NPB* benchmark suites, we select representative power-light, power-moderate, and power-hungry benchmarks that cover a wide range of scientific and computational domains. The global controller operates in kernel space and is triggered by userspace power management. The learning controllers operate

TABLE VI: Architecture parameters and hyperparameters for the hierarchical controller.

Configurations	Value
Number of cores	2-128
Core architecture	Intel Nehalem (x86)
V/F Levels (V/GHz)	1.20/2.0,1.08/1.8,0.96/1.6,0.84/1.4
Sta. Power at V/F Levels	19.2, 11.7, 6.9, 3.8
Dyn. Power at V/F Levels	23.6, 17.8, 13.5, 8.8
DVFS transition overhead	40 cycles
L1-I/D cache	32KB, 4-way, LRU
L2 cache	512KB, 8-way, LRU
L3 cache	8MB, 16-way, LRU
Global/learning/swift ctrl.	10 ms, 500μs, 4μs
Learning rate	1×10^{-3}
Discount reward factor	$\gamma = 0.95$
Trajectory size for backprop	25
Optimizer	Adam ($\beta_{1,2} = 0.9, 0.999$)

every 500 microseconds, a rate limited by the computational complexity of the learning algorithm. To accurately estimate the DVFS transition overhead, each voltage and frequency switch is set to 40 cycles by the Sniper simulator. For a conservative consideration, the swift controllers work at 4 microseconds scales, which can be supported by the integrated voltage regulators without extra efficiency drop as described in Table I. Later we also examine the swift controllers working at different DVFS speeds supported by integrated voltage regulators, as studied in Section III-A.

B. Hierarchical Fast Learning Approach

We compare **F-LEMMA** to the two state-of-the-art DVFS techniques on multi-/many-core processors: **Profit**, Priority and Power/Performance Optimization for Many-Core Systems [7], and **Grape**, Minimizing Energy for GPU Applications with Performance Requirements [13]. Our methods are normalized to the default race-to-idle execution mode. F-LEMMA and previous techniques are implemented in the same Sniper and Mcpat simulation platforms as described above. For fairness, F-LEMMA (the learning controller), Profit, and Grape all operate at a fixed timescale of 500 microseconds. Profit and Grape are implemented according to the best information found in the papers that describe them. The names of the approaches used in our experiments are given below.

- **Default Race-to-Idle.** Runs each benchmark as fast as possible. All other methodologies are normalized to this.
- **F-LEMMA:** The proposed learning-based fast power and energy management in a hierarchical layered approach.
- **Profit:** State-of-the-art *reinforcement learning-based* power, and energy management for multi-core and many-core systems [7].
- **Grape:** State-of-the-art *feedback control based* power and energy management for multi-core and many-core systems with performance constraints [13].

We compare F-LEMMA with Profit and Grape from three perspectives: energy, performance, and energy-delay product. We evaluate energy consumption, not power dissipation, for a standard comparison against workloads and configurations. The energy consumption metric evaluates net benefits and accounts for potential losses due to extended execution times when lowering frequency. We normalize energy and performance results to the energy consumed in Race-to-Idle case.

TABLE VII: Comparison of F-LEMMA, Profit, and Grape.

Approach	F-LEMMA	Profit	Grape
Normalized Energy Saving	35.2%	28.6%	23.7%
Performance Penalty	11.8%	8.3%	9.2%
Energy-Delay Product	0.73	0.78	0.84

The average ratios of DVFS decisions on the four voltage/frequency pairs in the proposed F-LEMMA are 50.1%, 15.8%, 10.7%, 23.4%. The average normalized energy, performance, and energy-delay product of these three approaches are summarized in Table VII. Compared to Profit; F-LEMMA achieves 6.6% extra energy savings with 3.5% performance penalties; compared to Grape, F-LEMMA achieves 11.5% extra energy savings with 2.6% performance penalties. Fig. 9 shows the normalized energy consumption, and Fig. 10 shows the normalized performance loss (instructions per second) across benchmarks. The best case is the *fft* benchmark, which saves 30.4% energy with only a 1.0% performance loss. The worst case is the *radix* benchmark, which saves 30.4% energy with a 25.3% performance loss. For the *fft*, *lu.cont*, *cholesky*, *water.nsq*, *blackscholes* and *ft* benchmarks, DVFS saves significant amounts of energy with a minimal performance penalty across all three power management approaches. In the Grape results, although using feedback control improves the effectiveness compared with the throughput-based DVFS in Section III on benchmarks *radix*, *bt*, and *cg*, it is still hard to achieve consistent effects across all the benchmarks.

We also compare the Energy-Delay Product of these approaches, which evaluates the benefits of energy saving with performance loss. Fig. 11 shows the *Energy-Delay Product* across benchmarks. On average, F-LEMMA, Profit and Grape have normalized energy- delay products of 0.73, 0.78, and 0.84, respectively. It indicates that F-LEMMA has the highest energy efficiency and smallest energy-delay product, after accounting for potential performance losses. F-LEMMA achieves a better (smaller value) of Energy-Delay Product than Profit (which uses a similar learning approach) across all the benchmarks, which indicates that F-LEMMA exploits DVFS

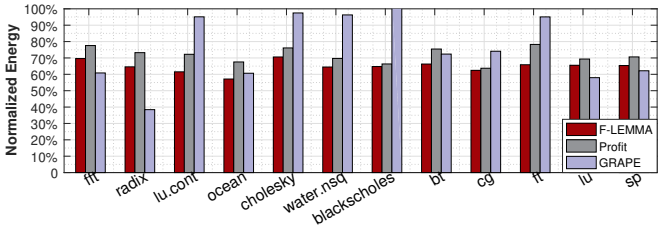


Fig. 9: Normalized energy consumption of F-LEMMA.

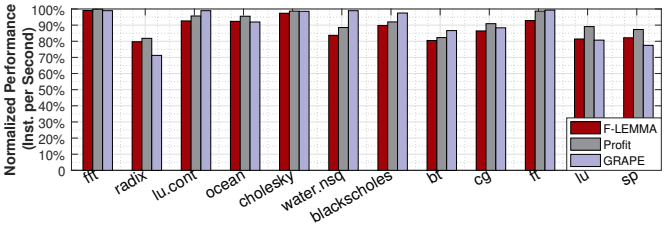


Fig. 10: Normalized performance of F-LEMMA.

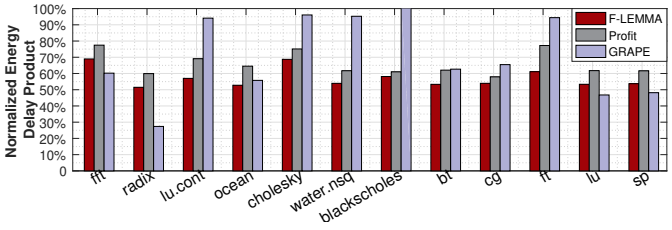


Fig. 11: Energy-delay product of F-LEMMA.

opportunities at microsecond time scales to achieve extra energy saving with small performance loss. Compared to Profit, F-LEMMA achieves better (smaller valued) Energy-Delay Products on 8 of 12 benchmarks. Moreover, the learning-based approach (i.e. F-LEMMA and Profit) is generally effective across all benchmarks, while the model-based approach (i.e. Grape) can achieve better results on specific benchmarks but worse results on remaining benchmarks.

C. Hierarchical Layered Approach with Ablation Study

We used an ablation study, shown in Fig. 12–13, to compare the energy savings and performance penalties from F-LEMMA and alternatives that use only a subset of the layered global, learning, and swift controllers. The configuration with only learning controllers works like online learning. In the swift controller only configuration, the weights for the controller inputs are from an off-line trained learning controller. F-LEMMA outperforms a framework with only global and learning controllers (i.e., the second bar), achieving significant energy savings with only a tiny performance loss. For example, on the *lu.cont*, *ocean*, and *ft* benchmarks, F-LEMMA achieves 9%, 8%, and 6% energy saving respectively, while reducing performance by less than 1%. F-LEMMA also outperforms a framework with only the swift controller (i.e., the third bar).

We also compare full hierarchical management with different configurations at each layer. Suppose the learning controller only pursues energy savings because the global controller specifies weights (1,0,0) for its reward function (i.e., the fourth bar). The system achieves more energy saving but with slightly greater performance penalties. Finally, suppose the swift controller uses only power as the input feature and neglects instruction throughput (i.e., the fifth bar). Compared

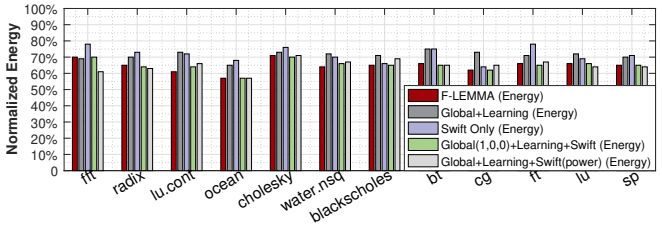


Fig. 12: Normalized energy consumption of F-LEMMA.

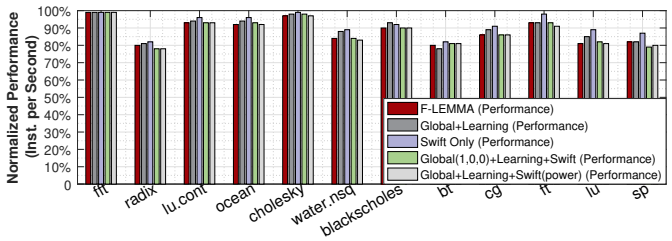


Fig. 13: Normalized performance of F-LEMMA.

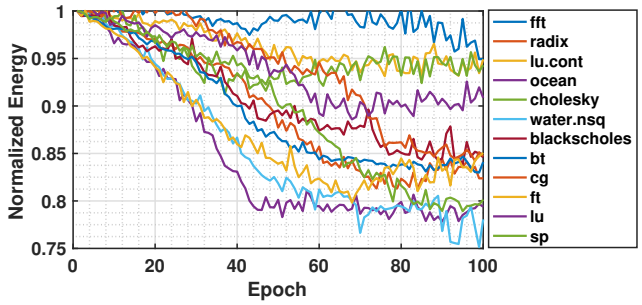


Fig. 14: Learning process within the same applications.

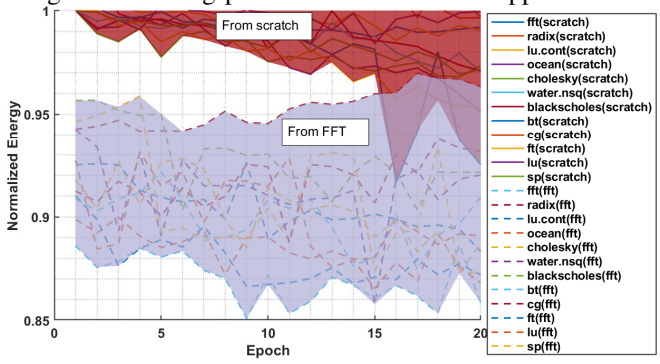


Fig. 15: Learning process across different applications.

to F-LEMMA, this configuration induces larger performance penalties for the same energy savings. With only power measurements, the swift controller predicts the effects of DVFS less accurately. These effects were discussed in Section III.3.

D. Workload Transition and Scalability

Finally, we examine the features needed or performed by F-LEMMA. The learning controller must be effective or converge quickly. First, the convergence process happens when the processor executes one application or keeps executing the same application. Second, the convergence process happens when the processor executes different applications such as finishing one application and then beginning to execute another application. Fig. 14 compares energy consumption when control starts from randomly generated weights (i.e., no

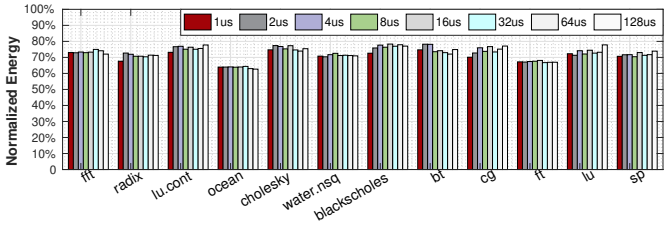


Fig. 16: Normalized energy consumption of F-LEMMA DVFS with the swift controller at different microsecond timescales.

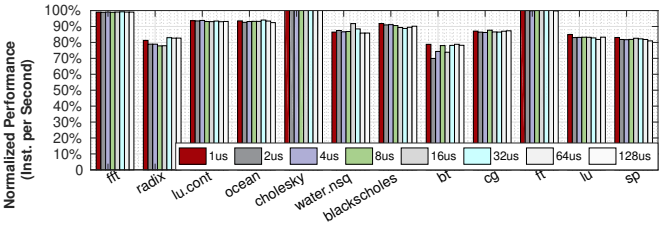


Fig. 17: Normalized performance of F-LEMMA DVFS with the swift controller at different microsecond timescales.

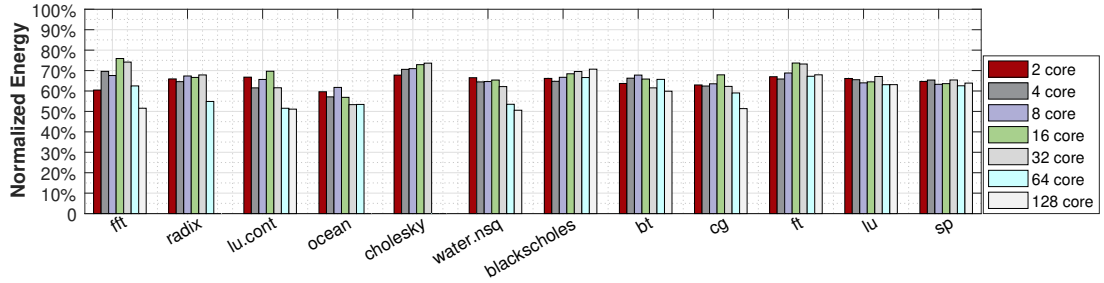


Fig. 18: Normalized energy of F-LEMMA on multi-core and many-core processors.

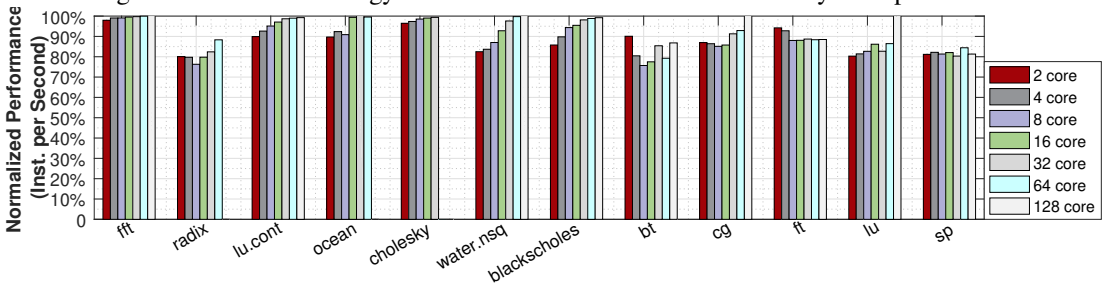


Fig. 19: Normalized performance of F-LEMMA on multi-core and many-core processors.

prior epoch). We can see that the training processes reach the convergent states in 40 epochs. Then, Fig. 15 compares energy consumption when control starts from randomly generated weights (i.e., no prior epoch) and starts from weights learned for the *fft* benchmark in the first two epochs before the workload transition. The energy is normalized to the first execution of each benchmark. Compared with starting from scratch, the benchmarks switching from *fft* inherit a learning controller configuration that was trained under *fft*, and present a significant energy saving at the beginning. Benchmarks other than *radix* save more energy as the execution proceeds. The same phenomena are also observed in transitions for other benchmarks.

We also changed the swift controller frequency to control the DVFS at different speeds as shown in Table I. Fig. 16 and Fig. 17 show the normalized energy consumption and performance across benchmarks. Generally, F-LEMMA is effective with different swift controller speeds. Meanwhile, a faster DVFS could achieve more energy savings with a small performance loss. By speeding up the swift control from 128us to 1us, the most significant energy saving is 7% and it is achieved on benchmark **cg** with 0.5% extra performance loss. Last but not least Fig. 18 and Fig. 19 show the energy and performance when scaling from 2 cores to 128 cores; some bars are blank because the benchmark does not support that configuration. Overall, as the number of cores scales from 2 to 128, F-LEMMA achieves from 35.2% to 41.1% energy saving at a cost from 12.1% to 5.4% performance loss overhead on average for 2, 4, 8, 16, 32, 64, and 128-core systems. As the number of cores increases, the performance penalty decreases as more DVFS opportunities are created by more thread synchronizations.

To summarize, F-LEMMA achieves effective power management at microsecond timescales with significant energy saving and only moderate performance loss. The global and learning controllers help the swift controller make better decisions, saving more energy across benchmarks. F-LEMMA is also effective during workload transitions and supports user space inputs to balance energy and performance according to specified weights. Furthermore, F-LEMMA can be applied from small multi-core systems up to many-core systems.

VII. Related Work

As more transistors are integrated on die and the limits of Dennard scaling are being reached, power and energy have become major constraints on processors' development. Many power management techniques have been proposed. In terms of power management architecture, hierarchical power management [40] has been widely adopted from mobile devices [41] to cloud computers [42]. Sartori et al. [43] studied peak power management in a distributed hierarchical configuration, given a power budget. Muthukaruppan et al. [44] and Ren et al. [10] used hierarchical frameworks for adaptive power management.

The power management algorithms usually fall in three categories: control model-based algorithm, optimization formulation-based algorithm, and recent learning-based algorithms. In terms of control model-based power and performance management algorithms, Haghbayan et al. [1], Rahmani et al. [2] and Shafique et al. [8] used a PID controller, a multi-objective controller, and an adaptive controller-based dynamic power management method to improve system power efficiency. For multiple objective optimizations, Rahmani et al. [45], [45], Ebi et al. [46], Lai et al. [47] and Kanduri et al. [48] explored reliability/variability, thermal, latency or accuracy aware solutions. Winter et al. [49] presented a thread scheduling and global power management co-design for a heterogeneous many-core processor. Meanwhile, Jung et al. [9], Shen et al. [15], [50], Chen et al. [51], [7], Rapp et al. [52] and Yu et al. [53] used a learning-based predictor and controller to find optimal power and performance. In above each subdirection, Ababei et al.[54] conclude the control model-based prediction and classification for processor power management; Khattar [55] et al. survey optimization techniques for processor power management on cloud applications; Pagani et

al. [56] summarize and compare the related machine learning techniques for power, energy, and thermal management on multicore processors. Performance and power management for special multi-core or many-core processors are studied in [57], [58], [59]. Limited by the supply voltage transition time and the complexity of effective power management algorithms, such management operates at millisecond timescales.

With the development of integrated voltage regulators (IVRs), per-core microsecond level fast DVFS has become practical. Kim et al. [17], Toprak-Deniz et al. [18], Meinerzhagen et al. [60], Kim et al. [28], and Keller et al. [20] designed IVRs that can support sub-microsecond level dynamic voltage scaling. Kim et al. [11] and Eyerman et al. [22] studied the potential system level energy benefits from microsecond level dynamic voltage scaling supported by on-chip IVRs. Höppner et al. [61] and Tseng et al. [19] studied fast DVFS on MPSoCs and SRAMs respectively. Kasture et al. [62] proposed a fine-grain DVFS scheme for latency-critical workloads. Bai et al. [37] proposed a voltage regulator efficiency aware power management strategy, which leverages voltage regulator runtime features and reinforcement learning to configure voltage and frequency. This work is extended from the previous workshop paper on MLCAD 2020 [63]. The previous paper demonstrates the prototype of learning-based microsecond power management. We extended the previous method and presented a complete microsecond DVFS solution.

VIII. Conclusion

In this paper, we proposed F-LEMMA, a hierarchical fast integrated voltage and frequency scaling approach for multi-core and many-core processors. With integrated voltage regulators, DVFS power management can reach microsecond timescales. A learning-based hierarchical approach, including a global controller in userspace, a learning controller at the architecture level, and swift controllers at the digital circuit level, is presented to guide microsecond level power management. Experimental results show that on average F-LEMMA can save 35.2% of energy with an 11.8% performance decrease. Compared with two classic millisecond timescale DVFS techniques using control theory and reinforcement learning, F-LEMMA achieves 5% and 11% Energy-Delay Product (EDP) improvements, respectively. Furthermore, F-LEMMA is readily applied to multi-core systems and can scale up to many-core systems.

IX. Acknowledgements

The research was partly supported by Semiconductor Research Corporation (SRC) Task 2810 and Task 2821; NSF CNS-1739643, CNS-1822085, CCF-1527610, and CCF-1408784; Samsung Research; Lenovo Research; and NSFC 62103268. We are grateful to the reviewers for their feedback.

References

[1] M.-H. Haghbayan, A.-M. Rahmani, A. Y. Weldezion, P. Liljeberg, J. Plosila, A. Jantsch, and H. Tenhunen, "Dark silicon aware power management for manycore systems under dynamic workloads," in *32nd International Conference on Computer Design (ICCD)*. IEEE, 2014.

[2] A.-M. Rahmani, M.-H. Haghbayan, A. Kanduri, A. Y. Weldezion, P. Liljeberg, J. Plosila, A. Jantsch, and H. Tenhunen, "Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2015.

[3] Wikipedia, "Speedstep," [EB/OL], https://en.wikipedia.org/wiki/SpeedStep/.

[4] A. Staff, "Amd powernow! technology brief," *Advanced Micro Devices, Inc*, 2002.

[5] T. Neagoe, E. Karjala, and L. Banica, "Why arm processors are the best choice for embedded low-power applications?" in *IEEE 16th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 2010.

[6] T. F. Review, "Nvidia power," [EB/OL], https://www.thefpsreview.com/2019/12/04/nvidia-geforce-driver-power-mode-settings-compared/.

[7] Z. Chen, D. Stamoulis, and D. Marculescu, "Profit: priority and power/performance optimization for many-core systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 10, pp. 2064–2075, 2017.

[8] M. Shafique, B. Vogel, and J. Henkel, "Self-adaptive hybrid dynamic power management for many-core systems," in *2013 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2013, pp. 51–56.

[9] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1395–1408, 2010.

[10] Z. Ren, B. H. Krogh, and R. Marculescu, "Hierarchical adaptive dynamic power management," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 409–420, 2005.

[11] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *14th International Symposium on High Performance Computer Architecture*. IEEE, 2008.

[12] B. Zimmer, Y. Lee, A. Puggelli, J. Kwak, R. Jevtić, B. Keller, S. Bailey, M. Blagojević, P.-F. Chiu, H.-P. Le *et al.*, "A risc-v vector processor with simultaneous-switching switched-capacitor dc–dc converters in 28 nm fdsoi," *IEEE Journal of Solid-State Circuits*, 2016.

[13] M. H. Santriaji and H. Hoffmann, "Grape: Minimizing energy for gpu applications with performance requirements," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture*, 2016.

[14] A. Majumdar, L. Piga, I. Paul, J. L. Greathouse, W. Huang, and D. H. Albonesi, "Dynamic gpgpu power management using adaptive model predictive control," in *2017 IEEE International Symposium on High Performance Computer Architecture*. IEEE, 2017, pp. 613–624.

[15] H. Shen, J. Lu, and Q. Qiu, "Learning based dvfs for simultaneous temperature, performance and energy management," in *Thirteenth International Symposium on Quality Electronic Design*. IEEE, 2012.

[16] Y. Wang, Q. Xie, A. Ammari, and M. Pedram, "Deriving a near-optimal power management policy using model-free reinforcement learning and bayesian classification," in *Proceedings of the 48th Design Automation Conference*, 2011, pp. 41–46.

[17] W. Kim, D. M. Brooks, and G.-Y. Wei, "A fully-integrated 3-level dc/dc converter for nanosecond-scale dvs with fast shunt regulation," in *IEEE International Solid-State Circuits Conference*. IEEE, 2011.

[18] Z. Toprak-Deniz, M. Sperling, J. Bulzacchelli, G. Still, R. Kruse, S. Kim, D. Boerstler, T. Gloekler, R. Robertazzi, K. Stawiasz *et al.*, "5.2 distributed system of digitally controlled microregulators enabling per-core dvfs for the power8 tm microprocessor," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2014.

[19] C.-Y. Tseng, L.-W. Wang, and P.-C. Huang, "An integrated linear regulator with fast output voltage transition for dual-supply srams in dvfs systems," *IEEE journal of solid-state circuits*, vol. 45, no. 11, pp. 2239–2249, 2010.

[20] B. Keller, M. Cochet, B. Zimmer, Y. Lee, M. Blagojevic, J. Kwak, A. Puggelli, S. Bailey, P.-F. Chiu, P. Dabbelt *et al.*, "Sub-microsecond adaptive voltage scaling in a 28nm fd-soi processor soc," in *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, 2016.

[21] C.-H. Chou, L. N. Bhuyan, and D. Wong, "μdpm: Dynamic power management for the microsecond era," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2019, pp. 120–132.

[22] S. Eyerman and L. Eeckhout, "Fine-grained dvfs using on-chip regulators," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 1, pp. 1–24, 2011.

[23] X. Wang, J. Xu, Z. Wang, K. J. Chen, X. Wu, Z. Wang, P. Yang, and L. H. Duong, "An analytical study of power delivery systems for many-core processors using on-chip and off-chip voltage regulators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 9, pp. 1401–1414, 2015.

[24] A. Roth, C. Zhou, M. Wong, E. Soenen, T.-C. Huang, P. Ranucci, Y.-C. Hsu, H.-C. Lin, C. Kuo, M.-J. Wang *et al.*, "Heterogeneous power delivery for 7nm high-performance chiplet-based processors using

integrated passive device and in-package voltage regulator," in *2020 IEEE Symposium on VLSI Technology*. IEEE, 2020, pp. 1–2.

[25] W. Godycki, C. Torng, I. Bukreyev, A. Apsel, and C. Batten, "Enabling realistic fine-grain voltage scaling with reconfigurable power distribution networks," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2014, pp. 381–393.

[26] A. Zou, J. Leng, Y. Zu, T. Tong, V. J. Reddi, D. Brooks, G.-Y. Wei, and X. Zhang, "Ivory: Early-stage design space exploration tool for integrated voltage regulators," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[27] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *2007 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2007, pp. 1–6.

[28] S. T. Kim, Y.-C. Shih, K. Mazumdar, R. Jain, J. F. Ryan, C. Tokunaga, C. Augustine, J. P. Kulkarni, K. Ravichandran, J. W. Tschanz *et al.*, "Enabling wide autonomous dvfs in a 22 nm graphics execution core using a digitally controlled fully integrated voltage regulator," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 18–30, 2015.

[29] X. Zhan, J. Chen, E. Sánchez-Sinencio, and P. Li, "Power management for multicore processors via heterogeneous voltage regulation and machine learning enabled adaptation," *IEEE Transactions on Very Large Scale Integration Systems*, 2019.

[30] B. Keller, M. Cochet, B. Zimmer, J. Kwak, A. Puggelli, Y. Lee, M. Blagojević, S. Bailey, P.-F. Chiu, P. Dabbelt *et al.*, "A risc-v processor soc with integrated power management at submicrosecond timescales in 28 nm fd-soi," *IEEE Journal of Solid-State Circuits*.

[31] S. Ghiasi, J. Casmira, and D. Grunwald, "Using ipc variation in workloads with externally specified rates to reduce power consumption," in *In Workshop on Complexity Effective Design*. Citeseer, 2000.

[32] G. Von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in dvfs-enabled clusters," in *International Conference on Cluster Computing and Workshop*. IEEE, 2009.

[33] Q. Wu, M. Martonosi, D. W. Clark, V. J. Reddi, D. Connors, Y. Wu, J. Lee, and D. Brooks, "Dynamic-compiler-driven control for microprocessor energy and performance," *IEEE Micro*, pp. 119–129, 2006.

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[35] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[36] M. Cochet, A. Puggelli, B. Keller, B. Zimmer, M. Blagojevic, S. Clerc, P. Roche, J.-L. Autran, and B. Nikolić, "On-chip supply power measurement and waveform reconstruction in a 28nm fd-soi processor soc," in *2016 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 2016, pp. 125–128.

[37] Y. Bai, V. W. Lee, and E. Ipek, "Voltage regulator efficiency aware power management," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 825–838.

[38] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–12.

[39] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *The 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009.

[40] P. Rong and M. Pedram, "Hierarchical power management with application to scheduling," in *ISLPED'05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.* IEEE, 2005, pp. 269–274.

[41] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins, "Turducken: hierarchical power management for mobile devices," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, 2005.

[42] N. Ioannou, M. Kauschke, M. Gries, and M. Cintra, "Phase-based application-driven hierarchical power management on the single-chip cloud computer," in *2011 International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 2011, pp. 131–142.

[43] J. Sartori and R. Kumar, "Distributed peak power management for many-core architectures," in *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2009, pp. 1556–1559.

[44] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2013, pp. 1–9.

[45] A. M. Rahmani, M.-H. Haghbayan, A. Miele, P. Liljeberg, A. Jantsch, and H. Tenhunen, "Reliability-aware runtime power management for many-core systems in the dark silicon era," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 25, no. 2, pp. 427–440, 2016.

[46] T. Ebi, M. A. Al Faruque, and J. Henkel, "Tape: Thermal-aware agent-based power econom multi/many-core architectures," in *2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers*. IEEE, 2009, pp. 302–309.

[47] Z. Lai, K. T. Lam, C.-L. Wang, and J. Su, "Latency-aware dvfs for efficient power state transitions on many-core architectures," *The Journal of Supercomputing*, vol. 71, no. 7, pp. 2720–2747, 2015.

[48] A. Kanduri, M.-H. Haghbayan, A. M. Rahmani, P. Liljeberg, A. Jantsch, H. Tenhunen, and N. Dutt, "Accuracy-aware power management for many-core systems running error-resilient applications," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 25, no. 10, 2017.

[49] J. A. Winter, D. H. Albonesi, and C. A. Shoemaker, "Scalable thread scheduling and global power management for heterogeneous many-core architectures," in *2010 19th International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 2010, pp. 29–39.

[50] H. Shen, Y. Tan, J. Lu, Q. Wu, and Q. Qiu, "Achieving autonomous power management using reinforcement learning," *ACM Transactions on Design Automation of Electronic Systems*, pp. 1–32, 2013.

[51] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2015.

[52] M. Rapp, A. Pathania, T. Mitra, and J. Henkel, "Prediction-based task migration on s-nuca many-cores," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1579–1582.

[53] Z. Yu, P. Machado, A. Zahid, A. M. Abdulghani, K. Dashtipour, H. Heidari, M. A. Imran, and Q. H. Abbasi, "Energy and performance trade-off optimization in heterogeneous computing via reinforcement learning," *Electronics*, vol. 9, no. 11, p. 1812, 2020.

[54] C. Ababei and M. G. Moghaddam, "A survey of prediction and classification techniques in multicore processor systems," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1184–1200, 2018.

[55] N. Khattar, J. Sidhu, and J. Singh, "Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 4750–4810, 2019.

[56] S. Pagani, P. S. Manoj, A. Jantsch, and J. Henkel, "Machine learning for power, energy, and thermal management on multicore processors: A survey," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 1, pp. 101–116, 2018.

[57] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture*, 2006.

[58] M. Ghasemazar, E. Pakbaznia, and M. Pedram, "Minimizing the power consumption of a chip multiprocessor under an average throughput constraint," in *2010 11th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2010, pp. 362–371.

[59] E. Shifer and S. Weiss, "Low-latency adaptive mode transitions and hierarchical power management in asymmetric clustered cores," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 10, no. 3, pp. 1–25, 2013.

[60] P. Meinerzhagen, C. Tokunaga, A. Malavasi, V. Vaidya, A. Mendon, D. Mathaikutty, J. Kulkarni, C. Augustine, M. Cho, S. Kim *et al.*, "An energy-efficient graphics processor featuring fine-grain dvfs with integrated voltage regulators, execution-unit turbo, and retentive sleep in 14nm tri-gate cmos," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 38–40.

[61] S. Höppner, C. Shao, H. Eisenreich, G. Ellguth, M. Ander, and R. Schüffny, "A power management architecture for fast per-core dvfs in heterogeneous mpsocs," in *2012 IEEE International Symposium on Circuits and Systems*. IEEE, 2012.

[62] H. Kasture, D. B. Bartolini, N. Beckmann, and D. Sanchez, "Rubik: Fast analytical power management for latency-critical systems," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2015, pp. 598–610.

[63] A. Zou, K. Garimella, B. Lee, C. Gill, and X. Zhang, "F-lemma: Fast learning-based energy management for multi-/many-core processors," in *2020 ACM/IEEE 2nd Workshop on Machine Learning for CAD (MLCAD)*. IEEE, 2020, pp. 43–48.