Power/Performance Modeling and Optimization: Using and Characterizing Machine Learning Applications

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Ermao Cai

B.S., Applied Physics, Shanghai Jiao Tong University

Carnegie Mellon University
Pittsburgh, PA

May 2018

Acknowledgements

The pursuit of PhD during the past five years has been a fruitful adventure for me, not only in my academic, but also in my personal life. I could not have achieved so much without the support from these resourceful, sincere, and brilliant people.

First and foremost, I would like to thank my advisor, Prof. Diana Marculescu. It has been an honor to have such a caring mentor, dedicated researcher, and a knowledgeable instructor to lead me over the years. When Diana first brought me to the research world, I had little experience and almost no engineering background. She trusted me, helped me, and created an unconstrained environment for me to explore and develop as a researcher. Her guidance and support made me more precise, more profound, and more creative. I sincerely hope when Diana looks back, she will think of taking me in as a right decision.

I would like to thank my committee member and collaborator, Dr. Da-Cheng Juan. Da-Cheng has set an example of excellence for a Ph.D. student. For the past five years, he not only has offered me helpful advice on tackling research problems and exploring ideas, but also has shown me how to enjoy research and life at the same time.

I would like to thank Prof. Radu Marculescu and Prof. Shawn Blanton for being my thesis committee members and providing invaluable guidance and feedback through this process.

I would like to thank my fellow researchers and collaborators: Mr. Dimitrios Stamoulis, Prof. Aarti Singh, Dr. Jinpyo Park, and Prof. Siddharth Garg for their great work, thoughts and ideas.

I would also like to thank my internship mentor: Dr. Florentin Dartu, for instructing me to solve real industrial challenges with my research skills.

My days at CMU were made enjoyable in large part due to the many friends and groups that became a part of my life. They are:

Mr. Guangshuo Liu, Mr. Zhuo Chen, Mr. Ruizhou Ding, Mr. Ting-Wu Chin, and Mr. Ahmet Fatih Inci, from EnyAC;

Mr. Bolun Li, Mr. Minghao Ruan, Dr. Sanxi Yao, Dr. Huang-Kai Peng, Dr. Renzhi Liu, and Mr. Xi He.

I would like to acknowledge the funding support received from Samsung Electronics, National Science Foundation (including Grants CCF-1314876, CCF-1514206, CNS-1128624, CNS-1331804, and CNS-1564022), and Carnegie Mellon University that has made pursuing my research possible.

Lastly, I would like to thank my family for all their love and support: my parents and my brother who have supported me in all my pursuits. And most of all, my loving and encouraging fiancée - I could not have done this without her.

Abstract

Energy and power are the main design constraints for modern high-performance computing systems. Indeed, energy efficiency plays a critical role in performance improvement or energy saving for either state-of-the-art general purpose hardware platforms, such as FinFET-based multi-core systems, or widely-adopted applications such as deep learning applications and in particular, convolutional neural networks. To achieve higher energy efficiency, power and performance models are key in enabling various predictive management algorithms or optimization techniques. To have accurate models, one needs to consider not only technology-related effects, including process variation, temperature effect inversion, and aging, but also application-related effects, such as the interaction between applications with software and hardware layers.

In this thesis, we study these effects and propose to combine machine learning techniques and domain knowledge to learn the performance, power, and energy models for high-performance computing systems. For technology-aware multi-core system design, we learn accurate performance and power models for FinFET-based multi-core systems considering various technology effects. By applying these models, we propose efficient power-/performance-related management algorithms for multi-core systems to 1) increase performance under iso-power constraints; 2) reduce power while keeping the same performance; and 3) decrease aging effects with negligible power overhead for the same performance. For application-aware computing system design, we propose a hierarchical framework based on sparse polynomial regression to predict the serving power, runtime, and energy consumption of deep learning applications, including convolutional neural networks. Extensive experimental results confirm the effectiveness of our proposed models, algorithms, and framework.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Power and energy issues have become the major design constraints in developing high-performance computing systems. Several hardware- and software-based techniques exist to resolve them. For multicore processors, Dynamic Voltage and Frequency Scaling (DVFS) is widely used in both academia and industry, mainly to reduce the power consumption while maintaining the same performance. Upscaling of voltage/frequency (V/F) for multicore processors has also been explored by processor manufacturers, *e.g.*, in Intel's Turbo Boost [15]. Similar techniques are also discussed in other work [88][45]. Nevertheless, as CMOS technology aggressively scales down to deca-nanometer technology nodes, these issues have also emerged as important reliability threats throughout the system lifetime. Aging concerns have therefore gathered a significant momentum and they have already triggered extensive research on aging modeling and mitigation techniques for older planar MOSFET-based system design.

From a technology standpoint, CMOS technology evolution has recently provided fundamental advancements in addressing power, thermal and aging problems. FinFET has been widely chosen as the next generation CMOS technology, especially for sub-20 nm technology nodes. Because of the different structures between planar CMOS and FinFET, different performance characteristics have been observed between them [54]. In terms of aging considerations, we have recently experienced a "paradigm shift" from older MOSFET-based

models to more accurate and detailed aging models [26]. Moreover, in terms of thermal considerations, the patterns of gate delay for FinFET with respect to temperature are quite different when compared with planar CMOS. The effect is called temperature effect inversion (TEI) [57]. TEI refers to the phenomenon in which the gate delay of FinFET decreases as temperature increases in superthreshold voltage region.

From the application perspective, deep learning, especially convolutional neural networks (CNNs) have been widely used in several important areas, such as text processing and computer vision, in both academic and industrial setups. That is because CNNs have claimed dominance over other existing methods in these domains [31] [36]. However, the high energy consumption of CNNs, which can be attributed to both (a) high power consumption and (b) long runtime, has limited the types of platforms that CNNs can be deployed on. Therefore, it is necessary to introduce energy-efficient CNNs to platforms with limited resources. To achieve energy-efficient CNN designs, it is critical to have accurate power, runtime and energy models.

1.1 Challenges

To resolve the power and energy issues for high-performance computing systems, there are two major challenges. The first is to have accurate power and performance models. Due to the fact that the complexity of the computing systems is increasing, it is always a challenge to have accurate power and performance models. As long as the models are found, one needs to face the second challenge: efficient system optimization. The optimization techniques usually vary greatly from one system to another. Therefore, we need to tackle the challenges case by case. Sometimes, domain knowledge can help to develop new techniques. We list several challenges in details in the following.

The first challenge is to address process variations (PVs). Near-Threshold Computing (NTC) has emerged as a promising solution to greatly increase the energy efficiency of next-generation multi-core systems. However, PVs expose an important effect on the performance

of NTC. Therefore, it is significant to understand and model the PVs, especially when NTC is adopted in emerging computing systems.

Second, it is critical to utilize TEI. As CMOS technology continues scaling, FinFET has recently become the common choice for multi-core systems. In contrast with planar CMOS, FinFET is characterized by lower delay under higher temperatures in super-threshold voltage region, *i.e.*, TEI. To fully utilize the potential of FinFET, it is critical to understand and consider TEI before optimizing the computing systems.

The third challenge comes from the thermal effects, especially aging. Power and thermal issues are the main constraints for high-performance multi-core systems. As the current technology of choice, FinFET is observed to have TEI. While it has been shown that system performance can be improved under power constraints, as technology aggressively scales down to sub-20nm nodes, thermal issues emerge as important reliability concerns throughout the system lifetime.

Finally, it is always hard to analyze the energy efficiency of deep learning applications, such as convolutional neural networks (CNNs). With the increased popularity of CNNs deployed on the wide-spectrum of platforms (from mobile devices to workstations), the related power, runtime, and energy consumption have drawn significant attention. From lengthening battery life of mobile devices to reducing the energy bill of datacenters, it is important to understand the efficiency of CNNs during serving for making an inference, before actually training the model. However, the complexity introduced from both the software side and hardware side makes it difficult to understand CNN efficiencies.

1.2 Thesis contributions

In this thesis, we combine machine learning techniques and domain knowledge to learn the performance, power models for high-performance computing systems. Experimental results show that our learning-based performance and power models on these systems have a high accuracy.

For technology-aware FinFET-based multi-core systems, we propose power-/performance-related management algorithms, including 1) increasing performance at iso-power; 2) reducing power while keeping the same performance; 3) decreasing aging effects with negligible power overhead at iso-performance. For application-aware computing systems which run various deep learning applications, especially CNNs, we apply our learning based model to predict the power, runtime, and energy for various neural network configurations and hardware platforms. The detailed contributions are described in the following.

Learning-based power/performance models: We evaluate PVs and learn the power and performance models for multi-core systems operating under extended range: including near-threshold, nominal, and Turbo modes. We propose to model the problem of the energy minimization or throughput maximization under given requirements as constrained convex optimization problems that can be solved efficiently. Considering PVs on FinFET-based chip-multiprocessors (CMPs), experimental results show that at 30% PV levels, our proposed method (1) reduces energy consumption by 31.09% at iso-performance and (2) increases throughput by 11.46% at iso-power when compared with variation-agnostic nominal case [7].

TEI-aware model and learning: We explore TEI-aware performance improvement and energy savings for multi-core systems. Our experimental results show that on average 15.70% throughput improvement or 31.26% energy savings can be achieved in steady state by a TEI-aware DVFS policy over a TEI-agnostic one. We observe multiple sweet spots resulting from TEI effects and introduce fast algorithms which provide iso-power maximum performance or iso-performance minimum energy consumption. Experimental results confirm the effectiveness of the proposed approach by exhibiting a 45.9-55.3x speedup when compared to state-of-the-art algorithms while losing only 0.22% or 0.68% in achieved performance or energy, respectively [9] [10].

Aging reduction: We are the first to provide a comprehensive evaluation of both TEI and aging effects on the performance and power of FinFET-based multi-core systems with multiple voltage/frequency levels. Our experimental results show that aging effects can be

reduced by up to 53.59% by exploiting the TEI effect. Based on a combined multivariate objective for power and aging, this work proposes an aging-aware algorithm, dubbed *AgingMin*, to select the optimal TEI-aware voltage/frequency operation points for decreasing the aging effects. Experimental results show that *AgingMin* improves the classic 10-year system lifetime by an average of 1.61 years while introducing less than 1% power overhead when compared to existing state-of-the-art techniques [11].

Modeling for CNN: we propose *NeuralPower*: a layer-wise predictive framework based on sparse polynomial regression, for predicting the serving power, runtime, and energy consumption of a CNN deployed on any GPU platform. Given the architecture of a CNN, *NeuralPower* provides an accurate prediction and breakdown for power and runtime across all layers in the whole network, helping machine learners quickly identify the power, runtime, or energy bottlenecks. The experimental results show *NeuralPower* predicts the runtime, power, and energy of state-of-the-art CNN architectures, with an average accuracy of 88.24%, 88.34%, and 97.21%, respectively. We comprehensively corroborate the effectiveness of *NeuralPower* as a powerful framework for machine learners by testing it on different GPU platforms [8].

1.3 Thesis organization

The rest of this thesis is organized as follows. Chapter 2 introduces the background knowledge related with PV, TEI, aging effects, and CNNs. Chapter 3 details the power and performance modeling and optimization for process-variation aware multi-core systems. Chapter 4 provides TEI-aware power-performance optimization for FinFET-based multi-core systems. Chapter 5 details the interaction of TEI and aging effects, and the algorithm to reduce aging process by considering TEI effect. Chapter 6 discusses the modeling process and results for power and runtime of convolutional neural networks. Chapter 7 provides the related work. Chapter 8 concludes this thesis and discusses the possible future research directions.

Chapter 2

Background

2.1 Technology-aware computing system design

FinFET, a type of non-planar double-gate device, has become the next generation CMOS technology of choice, especially for sub-20 nm technologies. In this section, we discuss the effects associated with this technology, including process variations (PVs), temperature effect inversion (TEI), and aging effects.

We first introduce PVs. As a compounding factor, when transistors scale down and supply voltage reaches near-threshold levels, PVs complicate chip design. Therefore, a thorough understanding of how PVs affect power-performance of CMPs is necessary. Due to the differences in channel and gate formations between bulk CMOS and FinFET, their major sources of process variations are not the same. FinFET has three parameters: gate length (L_G), work function (Φ_G) and fin thickness (T_{SI}), which dominate the PVs' effects [16] [55]. Compared to bulk CMOS, we can see there are new dominant factors for PVs' effects, like fin thickness, which is unique for FinFET with 3D fins. Generally a process parameter Θ affected by variations is described as: $\Theta = \Theta_{nom} + \delta_{total}$, where Θ_{nom} denotes the nominal value and δ_{total} is total variation from Θ_{nom}. In addition, It is widely known that variations are composed of wafer-to-wafer (W2W), die-to-die (D2D) and within-die (WID) variation.

Considering the spatial correlations induced by the manufacturing process, these random variables are difficult to characterize.

Second, we move to explain how TEI occurs. It is well-known that the operating frequency of a digital system is strongly affected by gate delays. The delay of a logic gate is greatly controlled by the drive current. When the drive current decreases, the corresponding gate delay increases, which finally determines a lower operating frequency of the system to meet the timing.

In the case of traditional planar CMOS, it is well known that as temperature increases, the corresponding drive current decreases in the superthreshold voltage region. Therefore, under high operating temperature, processors composed of billions of CMOS devices have to decrease their operating frequency to meet the timing and therefore, performance is decreasing.

For FinFET devices, the trends are quite distinct. One of the biggest differences between FinFET and planar CMOS devices is the structure of the channel. In contrast to planar CMOS comprising of a flat channel with one gate on top of it, FinFET devices have a 3D channel, which is called a "fin". As technology node decreases, the fin becomes smaller and thinner, thus becoming more vulnerable to structural effects. When temperature increases, the tensile effect from the insulator layer below the fin becomes larger, which induces the bandgap narrowing in the channel. The bandgap narrowing directly results in the drop in threshold voltage. In addition, the tensile stress causes a slight increase in the carrier mobility. Therefore, the increase in temperature results in a decrease in FinFET gate delay, which is TEI.

One direct consequence of TEI is that the worst case timing takes place at lowest operating temperatures for FinFET-based circuits. As temperature increases, gate delay decreases, which allows circuits to potentially run at a higher frequency. Digital systems are usually designed to work in a temperature range from -25 °C to 125 °C [57]. Therefore, the worst case timing in FinFET-based digital systems is at -25 °C, rather than 125 °C. To illustrate this effect, we run SPICE simulations for a fan-out-of-four (FO4) ring oscillator (RO) from

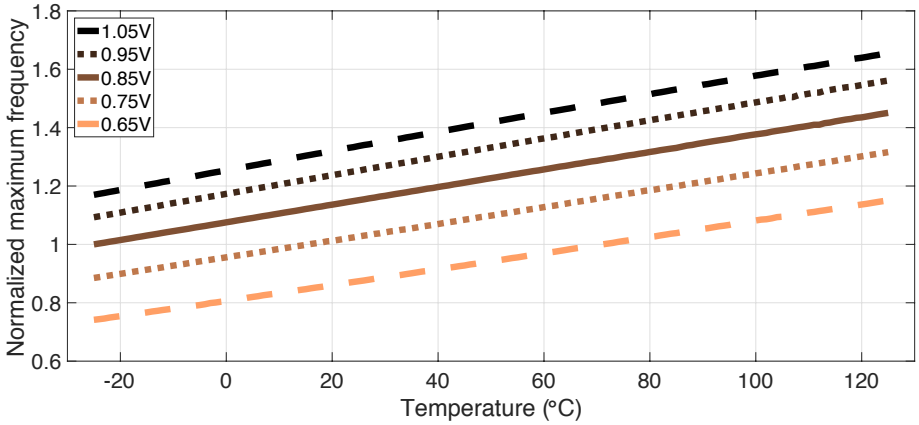

Figure 2.1: Maximum operating frequency under a wide range of temperature values for FinFET.

PTM [12] library in a 16 nm FinFET technology node, which is validated against ITRS roadmap [77]. Figure 2.1 shows the normalized maximum operating frequencies for different supply voltage levels in high-performance mode. It is clear that the operating frequency of FO4 RO increases as temperature gets higher. The same effect is also observed in low-power mode.

Finally, we introduce aging effects for FinFET. As described before, TEI allows for devices to run faster at higher operating temperatures. In Figure 2.2, the solid lines correspond to the normalized maximum operating ring oscillator (RO) frequency under the TEI effect. Nonetheless, the manifestation of aging has an exact opposite, counteracting effect: BTI/RTN mechanisms generate traps to capture minority carriers that eventually leads to ΔV_{th} fluctuations from the nominal threshold voltage value [85]. Consequently, existing TEI-aware analyses and system-level models inherently fail to account for aging-induced performance degradation, thus resulting in overly-optimistic results. To illustrate this key observation, the dotted lines in Figure 2.2 show the normalized maximum operating RO frequency under the TEI effect and under aging for three and ten years. To achieve a representative comparison, we repeat the HSPICE simulation for the same supply voltage levels as in the aging-unaware cases.

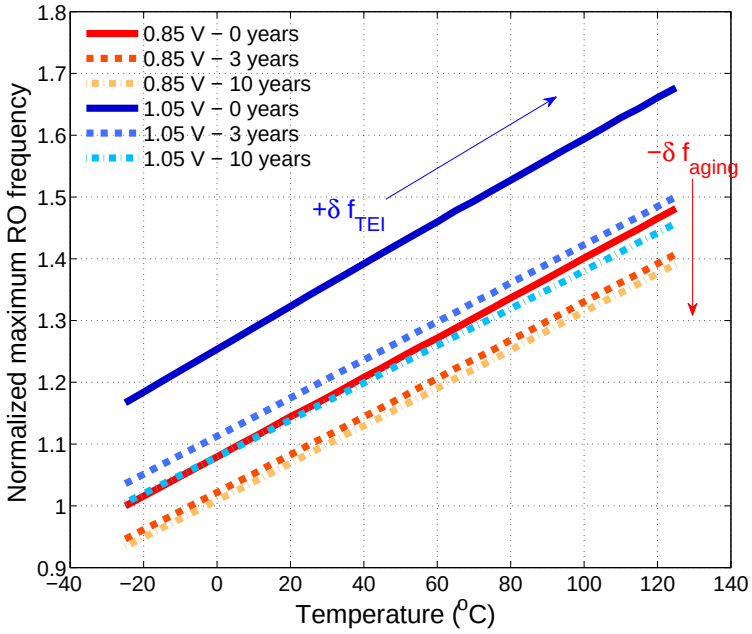

Figure 2.2: Maximum operating RO frequency under the impact of TEI and aging.

We can easily observe that overall performance throughout the RO lifetime is affected by both the TEI effect and the aging mechanisms. Thus, a model that inherently accounts for both these effects is essential for developing an effective yet representative aging reduction scheme for FinFET-based many-core systems.

2.2 Application-driven computing system design

In recent years, CNNs have been widely applied in several important areas, such as text processing and computer vision, in both academia and industry. However, the high energy consumption of CNNs has limited the types of platforms that CNNs can be deployed on, which can be attributed to both (a) high power consumption and (b) long runtime. GPUs have been adopted for performing CNN-related services in various computation environments ranging from data centers, desktops, to mobile devices. In this context, resource constraints in GPU platforms need to be considered carefully before running CNN-related applications.

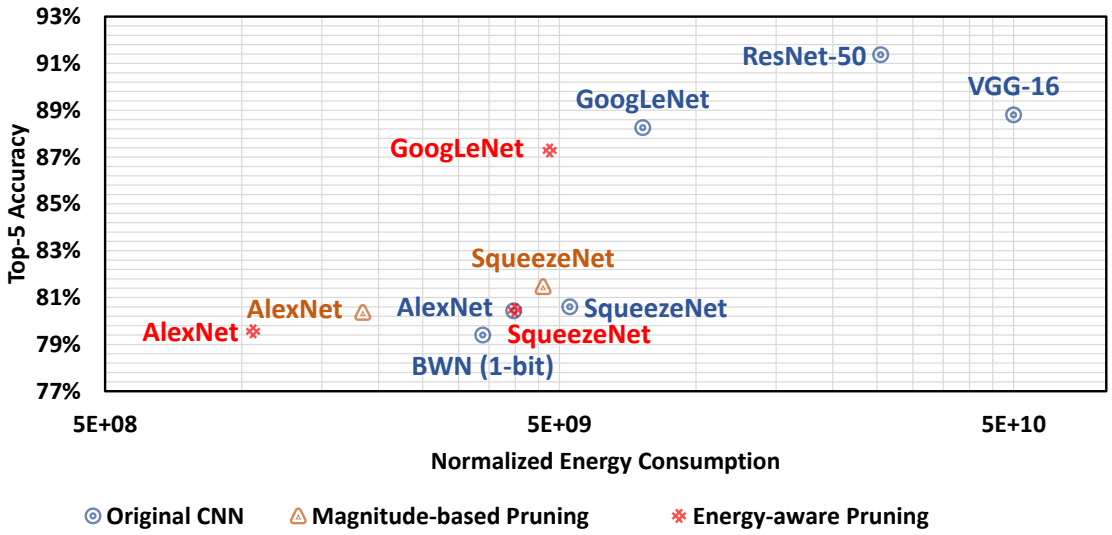

Figure 2.3: Prediction accuracy versus energy consumption of popular CNN models. This figure is adopted from [91]. Some models are pruned with either magnitude-based pruning [29] or energy-aware pruning [91]. For the architectures that achieve similar accuracy levels during test phase, the energy consumption can vary by close to 10×.

In this thesis, we focus on the *testing* or *service* phase since, CNNs are typically deployed to provide services (*e.g.*, image recognition) that can potentially be invoked billions of times on millions of devices using the same architecture. Therefore, testing runtime and energy are critical to both users and cloud service providers. In contrast, training a CNN is usually done once. Orthogonal to many methods utilizing hardware characteristics to reduce energy consumptions, CNN architecture optimization in the design phase is significant. In fact, given the same performance level (*e.g.*, the prediction accuracy in image recognition task), there are usually many CNNs with different energy consumptions. Figure 2.3 shows the relationship between model testing accuracy and energy consumption for a variety of CNN architectures [91]. We observe that several architectures can achieve a similar accuracy level. However, the energy consumption drastically differs among these architectures, with the difference close to 10× in several cases. Therefore, seeking for energy-efficient CNN architecture without compromising performance seems intriguing, especially for large-scale deployment.

Chapter 3

Learning for many-core system optimization

3.1 Chapter overview

Nowadays, power consumption plays a more and more important role in designing modern processors. Near-threshold computing (NTC) is a promising technique for power consumption reduction by lowering supply voltage to a value near the threshold voltage of transistors [37]. Operating frequency also scales down with supply voltage to give extra reduction in dynamic power consumption. However, as the frequency scales, the degradation in performance increases. In the context of multi-core systems, under performance constraint, it is not always possible to down-scale voltage/frequency due to the increased performance penalty. On the other hand, Turbo Boost (TB), where the operating voltage and frequency are upgraded higher than nominal values, has been widely implemented to improve the performance [15].

Although DVFS has been well studied for dynamic power management, extended range DVFS (ExDVFS)−DVFS over a wide operating range of voltage and frequency from NTC all the way to TB−remains to be further explored. In the turbo mode, one can upscale voltage/frequency points to achieve better performance, while in NTC mode, supply voltage decreases further to reduce power consumption. By allowing for an extended excursion for

the voltage/frequency pairs, one can more efficiently trade-off power or performance slack across cores running imbalanced threads so as to achieve better power efficiency or higher performance increase.

In addition to architecture-level advancement, FinFET, a type of non-planar double-gate device, has become main stream below the 22nm technology node. In the NTC region, since supply voltage is close to threshold voltage, the overdrive voltage is greatly affected by the change of threshold voltage induced by process variations (PVs). Although, PVs in FinFET have been extensively studied at the circuit level [90] [16], their impact in architecture level is yet to be explored. Therefore, there are more challenges considering FinFET-based computer systems with PVs.

As transistor technology evolves, it naturally motivates us to explore how power and performance optimization works on FinFET-based CMPs. Previous work addressed power and performance optimization on bulk-CMOS based CMPs [84] [44] [24] [63]. However, none of them has studied similar topics on FinFET-based CMPs, especially considering PVs. Our work in this chapter not only constructs a power and performance optimization framework, but also characterizes the impact of PVs and core count in FinFET-based CMPs. The detailed results will be discussed in Section 3.6.5.

3.1.1 Chapter contributions

To the best of our knowledge, the following novel contributions are described and supported in this chapter:

- We perform the quantitative characterization of PV effects in FinFET and propose an accurate multivariate polynomial model to learn power-PV parameter relationship. We then integrate PV information for each individual core in CMPs to get the contraint-posynomial frequency-power model.

- By leveraging the convexity of learned model, we convert the problem of the energy minimization under performance requirements into a constrained convex optimization problem that can be solved efficiently.

- As opposed to energy minimization, we also determine the maximum possible performance under a given power budget over a wide-operating range.

- In addition, further experiments **uniquely** show the benefits from our methods significantly increase with increasing core count and variation level, either in energy savings or throughput improvements. Our work also shows that iso-performance energy savings or iso-power performance improvements in FinFET technology are similar to those obtained for bulk CMOS, but the effect of PVs magnifies these benefits in the case of FinFET implementations.

3.2 Power model and performance metrics

Now we introduce the model used to characterize the dynamic and static power. For a logic module, the power can be divided into dynamic power (P_{dyn}) and static power (P_{sta}), which can be written as:

$$P_{tot} = P_{dyn} + P_{sta} = \alpha \cdot C_{tot} \cdot V_{dd}^2 \cdot \mathfrak{F} + k_{sta} \cdot I_{leak} \cdot V_{dd} \tag{3.1}$$

where α is the switching activity rate, C_{tot} is load capacitance for the switching gates, and k_{sta} is a constant proportional to the transistor count of that module. In addition, $\mathfrak{F}$ is the clock frequency, I_{leak} is the gate leakage current, and V_{dd} is the supply voltage.

However, the dynamic power and static power for a core in a CMP are more complicated than the expression in Equation 3.1. To achieve a good abstraction of core level power model, we adopt the convex power similar to the one proposed by Juan *et al.* [44]. The total power consumption of a core is formulated as:

$$P(\mathfrak{F}) = P_{dyn}^{peak} \cdot f_d(\mathfrak{F}) \cdot u(\mathfrak{F}) + P_{sta}^{peak} \cdot f_s(\mathfrak{F}) \tag{3.2}$$

where P_{dyn}^{peak} and P_{sta}^{peak} are the peak dynamic power and static power respectively, which are determined for the case in which all process parameters, operating voltage/frequency are at nominal values and the utilization is full. The $f_d(\mathfrak{F})$ and $f_s(\mathfrak{F})$ are two functions which characterize the dynamic power and static power in terms of the operating frequency and implicitly supply voltage. Finally, $u(\mathfrak{F})$ expresses the change in dynamic power due to the change of workload, *i.e.*, the change in the utilization. As the utilization is implicitly a function of frequency, we use $u(\mathfrak{F})$ to denote this.

One of the main drawbacks of the power model stated in Equation 3.2 is that it doesn't take PVs into consideration. To address this problem, we propose a hierarchical two-level regression model in Section 3.3.1.

After obtaining the model for f_d and f_s in Equation 3.2, we still need an accurate model for u. As shown in previous work, for CMPs, instructions committed per cycle (IPC) is a good approximate for the activity rate of one core. Based on prior work [66] [4], we approximate the dynamic power as a linear function of IPC. As the other components in Equation 3.2 are utilization independent, it follows that u is a linear function of IPC:

$$u = c_1 \cdot IPC + c_2 \tag{3.3}$$

To measure the performance of a CMP, we use the total throughput (TP) as the metric. Then, performance can be calculated as:

$$TP = \sum_{i=1}^{n} IPC_i \cdot \mathfrak{F}_i \tag{3.4}$$

where IPC_i and $\mathfrak{F}_i$ are the instructions per cycle (IPC) and the operating frequency for core i, while n is the total number of cores.

3.3 Model learning process

The proposed power model has two components: (1) the frequency function f, and (2) the utilization function u. To learn $\hat{f}$ and $\hat{u}$ [1], we adapt constrained-posynomial functions

[1] As a notation convention in machine learning and statistics, any symbol with a $\hat{hat}$ represents an estimate, instead of actual values or functions

to map both operating frequency and utilization to the corresponding power consumption by using leave-one-out cross validation (LOOCV). Compared to CMOS-based CMPs, we integrate process variations in the learning process for FinFET-based CMPs. The accuracy and overall validation of the learned model are also provided.

3.3.1 Learning the frequency-power relationship

Extended range operation, especially near-threshold voltage, is more severely affected by process variations than nominal operation and thus, PV effects must be incorporated for a robust model. To address this issue, we propose a variation-aware regression model customized for each core based on PV maps.

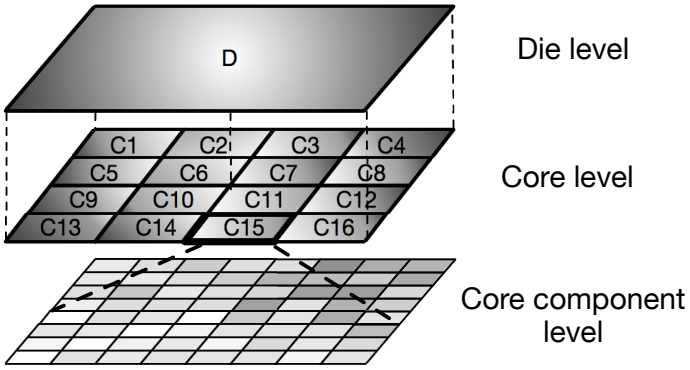

Figure 3.1: Hierarchical grid-based modeling of process variations for a CMP die with 16 cores. Different grey levels represent different PV values. The PV pattern used here is only for illustration.

First, we implement a hierarchical grid-based PV model considering spatial correlation as shown in Figure 3.1, a similar one used by Agarwal *et al.* [1]. Specifically, for component j in core i, the total variation can be expressed as:

$$\delta_{total_{i,j}} = \delta_D + \delta_{C_i} + \delta_{R_j} \tag{3.5}$$

where δ_D is die-level variation, δ_{C_i} core-level variation and δ_{R_j} component-level variation. The three components in Equation 3.5 are independent identical Gaussian random variables (RVs) [1] [14] with mean zero. In addition, we assume their standard deviation is σ. For

L_G and T_{SI}, we consider two cases, 3σ as 10% and 30% of the corresponding nominal value. For Φ_G, we also consider two cases, 3σ as 10% and 30% of $|\Phi_{G,N} - \Phi_S|$, where $\Phi_{G,N}$ is the corresponding nominal value and Φ_S is the work function of the intrinsic semiconductor material [16].

Next, we propose a grid-based modeling methodology in which we first learn the power model for every grid cell (representing a core component) as a function of the process parameters in that grid cell and V_{dd}. To quantify the impact of process parameters, we propose a model for total power consumption in a module (or core component) as:

$$P_m(\mathbf{X}) = P^{peak}_{m,dyn} \cdot f_{m,d}(\mathbf{X}) \cdot u_m(\mathfrak{F}) + P^{peak}_{m,sta} \cdot f_{m,s}(\mathbf{X}) \tag{3.6}$$

where $\mathbf{X} = [V_{dd}, L_G, \Phi_G, T_{SI}]$. V_{dd} is paired with $\mathfrak{F}$ for a set of PV parameters. $P^{peak}_{m,dyn}$ and $P^{peak}_{m,sta}$ are the peak dynamic and static power, respectively. $f_{m,d}(\mathbf{X})$ and $f_{m,s}(\mathbf{X})$ are two functions which characterize the dynamic/static power in terms of process parameters and V_{dd}. By using process variation maps, one can integrate all components of a specific core, and obtain the final $(p^\ell, \mathfrak{F}^\ell)$ data for a typical core as in Section 3.3.1.

We evaluate the dominant process parameters L_G, T_{SI} and Φ_G. We use PTM model cards for 16nm FinFET technology node to form a 23-stage Ring Oscillator (RO) in HSPICE to characterize the delay and total power, and a two-stage inverter chain to characterize the static power. We vary the process parameter values to get their effects on the performance and power. We also change the supply voltage V_{dd} from 0.35V to 1.05V (nominal value is 0.85V) to change operating mode from NTC all the way to TB. Refering to [41], the operation models are defined as: (1) NTC: $V_{dd} = [0.35, 0.55]$; (2) regular DVFS range: $V_{dd} = [0.55, 0.85]$; (3) TB: $V_{dd} = [0.85, 1.05]$. We change the V_{dd} and process parameters values to measure the frequency, dynamic power, and static power for the RO in HSPICE. Figure 3.2 shows HSPICE simulation results - the normalized maximum frequency, dynamic power and static power corresponding to the three process parameter variations for two cases: mild variation ($3\sigma = 10\%$) and extreme variation ($3\sigma = 30\%$), with respect to different V_{dd} values. It shows mild (extreme) variation introduce up to **26.13% (81.89%)** variation in delay,

19.42% (51.59%) in dynamic, and **3.55X (217X)** in leakage power. Furthermore, these variations tend to have a higher impact on delay near threshold, while for dynamic/static power they tend to be higher in nominal and turbo modes.

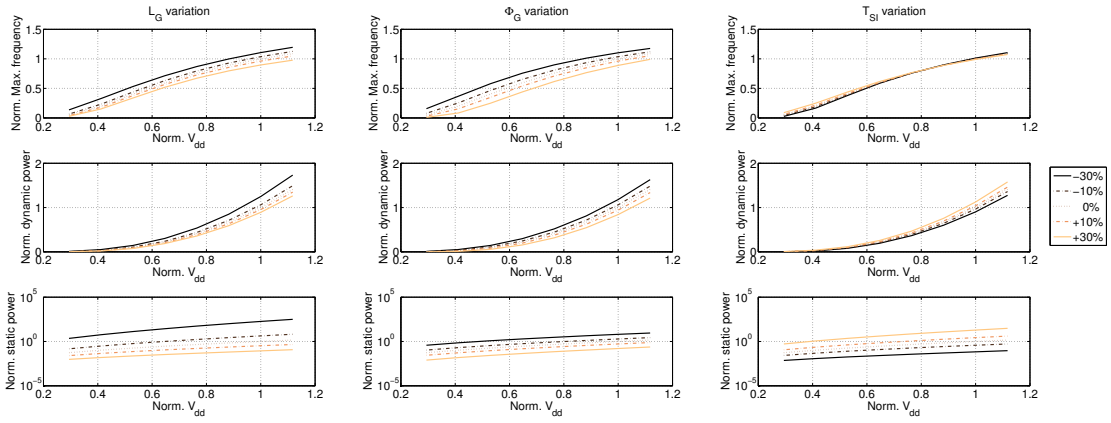

Figure 3.2: Maximum frequency, dynamic power and static power with respect to process variations. All three PV parameters have a significant impact on static power. They also affect delay and dynamic power across a wide range of operating frequency.

For frequency and dynamic power, we choose a multivariable polynomial regression model. For static power, an exponential multivariable polynomial regression model is a better choice. That is because the static power follows an exponential dependency with respect to the threshold voltage.

Let's consider the dynamic power (for simplicity, we denote it as $f_m(\mathbf{X})$) as example to illustrate how we establish our power model for a module. After experimenting with different multivariable polynomial regression models, we finally choose the one with a total degree of up to four:

$$f_m(\mathbf{X}) = \sum_i c_i \cdot V_{dd}^p \cdot L_G^q \cdot \Phi_G^r \cdot T_{SI}^s \tag{3.7}$$
$$p, q, r, s \in \{\mathbb{N}, 0\}, \quad p + q + r + s \leq 4.$$

Using a similar learning-based regression process as shown in Section 3.3.1, we achieve a Root-Mean-Square-Percentage-Error (RMSPE) as low as **0.25%** for the dynamic power in the cross validation phase. The same technique applies to the regression of the maximum

frequency $\mathfrak{F}$ with cross validation RMSPE of **0.38%**. For static power, the exponential multivariable polynomial model achieves a cross validation RMSPE of **0.96%**.

We continue to collect the power and frequency for the cores in the CMPs. As mentioned before, we use grid-based PV maps to simulate the PVs in CMPs. Specifically, there are three variation sources, die level, core level and core component level. In our experimental setup, one die has 4×4 cores, and each core is composed of 8×8 grid cells, which is illustrated in Figure 3.1. Therefore, the total variation $\delta_{total_{i,j}}$ for each grid cell j in core i is composed of three Gaussian RVs: δ_D in that die, δ_{C_i} in core i and δ_{R_j} in grid cell j as described in Equation 3.5. Given a set of PV maps, the power consumption for each core only depends on the resulting voltage/frequency pair while considering process variations as described above. By using the accurate power-frequency model we get from the detailed per-core variation information, we achieve unique values for $\mathfrak{F}$, f_d and f_s for each V_{dd} of each core, based on the following rules. $\mathfrak{F}$ is the minimum frequency across all the grid cells in the core so as to satisfy the timing within each core, while f_d or f_s is the sum of $f_{m,d}$ or $f_{m,s}$ for all grid cells in that core for a given V_{dd}. After this integration step, we get the pair $(p^\ell, \mathfrak{F}^\ell)$ for each core. Due to this integration of PVs, our proposed power models are implicitly heterogeneous, even for a homogeneous multi-core architecture.

3.3.2 Learning the utilization-power relationship

After $\hat{f}_d$ and $\hat{f}_s$ are obtained, we now learn the utilization-power relationship, $u(\mathfrak{F})$. As prior work [4] has pointed out, the dynamic power can be approximated as a linear function of IPC, since IPC approximately represents the activity rate of a processing core. Therefore, $u(\mathfrak{F})$ can be expressed as:

$$u(\mathfrak{F}) = c_1 \cdot IPC + c_2 \tag{3.8}$$

where c_1 and c_2 are fitting coefficients. From the dataset, a positive correlation between IPC and dynamic power consumption has been observed. In other words, higher IPC contributes to higher dynamic power dissipation, and vice versa, which leads to a positive slope c_1.

Furthermore, c_2 is also positive since power is still consumed even for a very low (or close to zero) IPC. Therefore, both c_1 and c_2 are positive and the convexity of model is maintained. Equation 3.8 is a constrained posynomial with $d = 1$ (the highest order of the polynomial is one), and we repeat the learning procedure described in Section 3.3.1 to learn $\hat{c}_1$, $\hat{c}_2$ with d set to one.

Table 3.1: Target architecture.

Parameters	**Values**
Number of cores	16
Nominal frequency	2660 MHz
Core model	Intel®-X86 Gainestown®-like
L2 caches	Private 256KB, 4-way SA, LRU
L3 caches	Shared 32MB, 16-way SA, LRU
DRAM	4GB
Technology	16nm node with FinFET

Here, we describe the dataset (p^ℓ, IPC^ℓ) used to learn the $\hat{c}_1$, $\hat{c}_2$ of $\hat{u}(\mathfrak{F})$. The operating frequency of (p^ℓ, IPC^ℓ) is fixed at the nominal value, and therefore the changes in dynamic power dissipation come only from the workload characteristics, not voltage/frequency scaling. We use Sniper [13] as the architectural simulator to collect IPCs (IPC^ℓ) and other required workload characteristics. The target architecture is described in Table 3.1. Default settings are used for the parameters not mentioned here. To demonstrate our approach, we use both PARSEC [3] and SPLASH-2 [89] benchmarks that contain a wide spectrum of multi-threaded parallel applications. For power, we use McPAT [58] to collect dynamic power traces (p^ℓ). The temperature here is set to 330K, a typical stable temperature for our target processors. McPAT also provides 16nm FinFET technology parameter set to simulate power, area, and timing under 16nm node. Therefore, McPAT can simulate power of Intel Gainestown-like cores in 16nm FinFET technology. Finally, the average error of the learned is $\hat{u}(\mathfrak{F})$ 2.84%. Similar results are also reported by others [4]. After $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}(\mathfrak{F})$ are obtained, we are in position to examine the overall learned model $\hat{P}(\mathfrak{F})$.

3.3.3 Model validation

So far, the accuracies of $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}(\mathfrak{F})$ have been determined separately, and therefore the accuracy of the overall learned $\hat{P}(\mathfrak{F})$ remains unknown. In this section, $\hat{P}(\mathfrak{F})$ is validated with the power consumption of a whole processor at different frequencies under various workload characteristics. By plugging $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}(\mathfrak{F})$ into Equation 3.2, the overall power function $\hat{P}(\mathfrak{F})$ can be expressed as:

$$\begin{aligned}\hat{P}(\mathfrak{F}) &= P_{dyn}^{peak} \cdot \hat{f}_d(\mathfrak{F}) \cdot \hat{u}(\mathfrak{F}) + P_{sta}^{peak} \cdot \hat{f}_s(\mathfrak{F}) \\ &= P_{dyn}^{peak}\Big(\sum_{i=0}^{5} \hat{\alpha}_i \mathfrak{F}^i\Big)\Big(\hat{c}_1 \cdot IPC + \hat{c}_2\Big) + P_{sta}^{peak}\Big(\sum_{j=0}^{6} \hat{\alpha}'_j \mathfrak{F}^j\Big)\end{aligned} \tag{3.9}$$

We collect the power traces, both dynamic and static power, from McPAT with the settings described in Section 3.3.2. In addition to the nominal frequency (2.66GHz), the frequencies are also set to the range (from 3.06GHz to 1.46GHz) at which McPAT has been extensively validated. Please note that NTC is not included here, since McPAT has not been validated for NTC voltage values. Furthermore, only the power values at the nominal frequency are used to train $\hat{u}(\mathfrak{F})$ as described in Section 3.3.2. In other words, the power values calculated not at the nominal frequencies are not involved in any part of the training process of $\hat{u}(\mathfrak{F})$ (and thus $\hat{P}(\mathfrak{F})$) – they are "clean" to test the accuracy of $\hat{P}(\mathfrak{F})$.

Table 3.2: Validation error of the learned $\hat{P}$.

Benchmark	blackscholes	canneal	dedup	fluidanimate
3.06GHz	3.72%	1.12%	10.19%	3.05%
1.46GHz	5.10%	3.69%	12.08%	3.57%
Benchmark	streamcluster	swaptions	vips	ferret
3.06GHz	7.17%	1.49%	12.56%	1.90%
1.46GHz	9.20%	1.77%	14.26%	2.56%
Benchmark	barnes	fmm	lu.cont	ocean.cont
3.06GHz	7.02%	4.79%	6.22%	6.47%
1.46GHz	5.95%	3.68%	6.83%	7.69%
Benchmark	radiosity	radix	water	**Average**
3.06GHz	2.24%	5.11%	0.98%	**4.94%**
1.46GHz	1.09%	5.55%	1.02%	**5.33%**

Table 3.2 illustrates that the overall average error is **4.94%** and **5.33%** for the two modes, respectively, or **5.14%** on average across various benchmarks in both PARSEC and SPLASH-2. This confirms that the learned model can accurately describe the power-performance relationship under different voltage/frequency levels and work variations.

3.4 Constrained optimization

We construct a convex power model expressed in Equation 3.2. As mentioned in Section 3.2, P_{dyn}^{peak} and P_{sta}^{peak} are two positive design parameters. f_d and f_s are two posynomial function of $\mathfrak{F}$. The remaining u is a linear function of IPC with two positive coefficients c_1, c_2. Therefore, Equation 3.2 is convex with respect to $\mathfrak{F}$ when given a certain IPC value.

In this chapter, we perform a limit study for ExDVFS, with and without PVs and considering various multi-threaded benchmarks on a CMP. To achieve that, we assume here that the voltage/frequency pairs are continuous across the wide operating range, which may not be practical, but offers a conservative upper bound on savings achieved when evaluating ExDVFS. We make use of convex program to formulate and analyze the two following optimal problems: (1) minimize energy consumption under iso-performance requirement; (2) maximize performance under iso-power conditions.

3.4.1 Constrained energy minimization

We first study the energy minimization under given throughput constraint. In this work, we assume that the workload characteristic, especially the IPC for each core in each control epoch can be obtained from performance counters. Based on this, this problem for an n-core CMP can be formulated as a convex program during each control epoch:

$$\begin{aligned} \text{Objective} \quad &: \quad arg\min_{\mathfrak{F}_i} \sum_{i=1}^{n} P_i(\mathfrak{F}_i) \\ \text{Subject to} \quad &: \quad \sum_{i=1}^{n} IPC_i \cdot \mathfrak{F}_i \geq Perf_{const} \qquad (3.10) \\ &\quad \mathfrak{F}_{min} \leq \mathfrak{F}_i \leq \mathfrak{F}_{max} \ , \forall i \qquad (3.11) \end{aligned}$$

As IPC_i for each core i is given, the power function P_i of that core is a convex function of $\mathfrak{F}_i$. There are two constraints that need to be satisfied for each ExDVFS control epoch: (1) the throughput should be at least equal to a specific value to meet the iso-performance requirement, and (2) the operating frequency $\mathfrak{F}$ for each core should be in proper operating ranges, which is defined by our operating ranges in HSPICE. Please note that the power model P_i varies from one core to another due to the PVs.

3.4.2 Constrained throughput maximization

We also consider the dual problem of performance improvement under power constraint, in other words, how much the variation-aware ExDVFS approach can improve the throughput under given power budget. Similarly, given IPC (IPC) for all cores in each ExDVFS control epoch, we can express this problem in a n-core CMP as:

$$\begin{aligned} \text{Objective} \quad &: \quad arg\max_{\mathfrak{F}_i} \sum_{i=1}^{n} IPC_i \cdot \mathfrak{F}_i \\ \text{Subject to} \quad &: \quad \sum_{i=1}^{n} P_i(\mathfrak{F}_i) \leq Power_{const} \qquad (3.12) \\ &\quad \mathfrak{F}_{min} \leq \mathfrak{F}_i \leq \mathfrak{F}_{max} \ , \forall i \qquad (3.13) \end{aligned}$$

The objective is a linear function of $\mathfrak{F}$. There are two constraints: (1) the total power consumption is at most equal to the power budget, which is convex constraint, and (2) the frequency range constraint is the same as Equation 3.11. The above is still a convex optimization problem since the objective is the maximum of a concave function and the constraints are convex.

Convex optimization problem can be solved efficiently by Interior Point method and Dual methods [6]. In this work, we implement the Interior Point method by using Matlab® to obtain the optimal $\mathfrak{F}$ that minimizes the energy consumption of a CMP while satisfying the throughput and other physical constraints. The Interior Point method is extremely fast, taking only **5ms** in Matlab to select the best $\mathfrak{F}$ each control epoch. Therefore, implementing

it as part of the OS kernel will be much faster, which opens the possibility of its applicability in an online setting.

3.5 Implementation flow

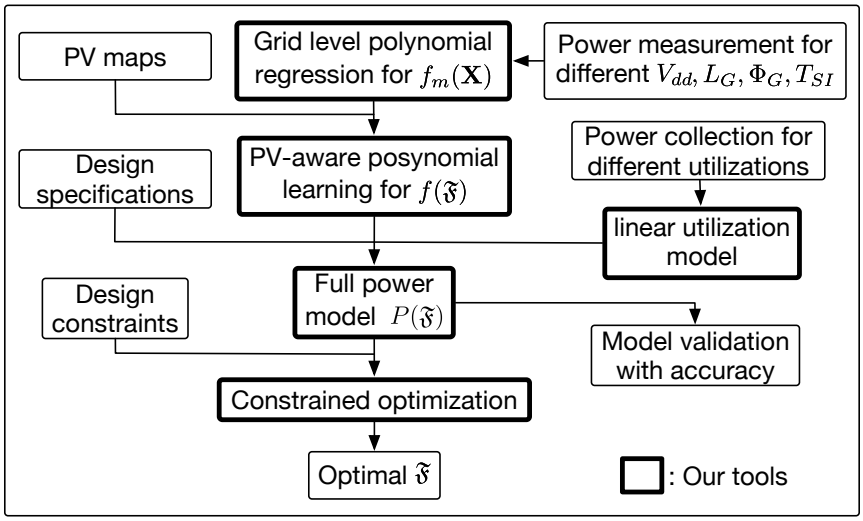

Figure 3.3: Implementation flow for the proposed framework.

The flowchart is provided in Figure 3.3. To begin with, we collect the dataset $(p^\ell, \mathfrak{F}^\ell)$ for learning $\hat{f}(\mathfrak{F})$ that captures the changes of power consumptions at wide-range operating frequencies (from NTC to turbo mode) as described in Section 3.3.1. To collect the variation-aware dataset for each core, we make the power measurement across various V_{dd} and dominant process parameters (L_G, T_{SI}, and Φ_G), and perform the regression for grid level power $\hat{f}_m(\mathbf{X})$ and the integration of PV maps. Next, we learn $\hat{u}(\mathfrak{F})$ that models the dynamic power changes according to the workload variations as described in Section 3.3.2. The design-specific P_{dyn}^{peak} is set to 70% of thermal design power (TDP)[2] for a core, and P_{sta}^{peak} is set to 30% TDP, which is approximately an average proportion for our experimental results and is also similar to results for Intel Xeon core in McPAT paper [58]. By plugging $\hat{f}_d(\mathfrak{F})$, $\hat{f}_s(\mathfrak{F})$ and $\hat{u}(\mathfrak{F})$ into Equation 3.2, $\hat{P}$ is obtained. We then validate the learned $\hat{P}$ with the performance and power values provided by Sniper and McPAT executing PARSEC and

[2]TDP refers to the average maximum power a processor can dissipate to avoid overheating.

SPLASH-2 benchmarks at different operating frequencies, as described in Section 3.3.3. The average RMSPE is **5.14%**. Furthermore, the overhead of voltage transitions for DVFS is less than 9ns [65] and therefore is negligible during each control epoch (1ms). We emphasize that the proposed learning framework is generic and is not restricted to a certain simulator or application. To perform the limit study of the maximum benefits of deploying wide-range operations from NTC to turbo mode, the workload characteristics of each processing core along with user-specified constraints are fed into the optimization framework described in Section 3.4.1 and 3.4.2 to select the best $\mathfrak{F}$ for each control epoch. Note that workload characteristics from other sources, such as on-chip performance counters, can also be plugged into the proposed framework for calculating the best $\mathfrak{F}$. Finally, the performance constraint in Equation 3.10 and the power constraint in Equation 3.12 are set to the throughput achieved and power consumed, respectively, under the nominal V_{dd} and $\mathfrak{F}$ for each benchmark. To reduce the randomness, our results are based on at least ten independent sets of PV maps.

3.6 Experimental results

In this section, we establish our experiments and perform the convex optimization as formulated in Section 3.4. We set normalized operating frequency from 0.05 to 1.22 (nominal $\mathfrak{F}$= 2.66GHz), which covers from NTC up to turbo mode.

3.6.1 Iso-performance energy optimization

For simplicity, we use ExDVFS to denote the DVFS over wide operating ranges: NTC, nominal and turbo mode. To achieve a comprehensive analysis of the effects due to PVs, we compare the following approaches:

- **Baseline**: the approach without ExDVFS or variation-aware adaption. All cores run at the same nominal V_{dd} and conservative $\mathfrak{F}$ (the slowest $\mathfrak{F}$ across all cores).

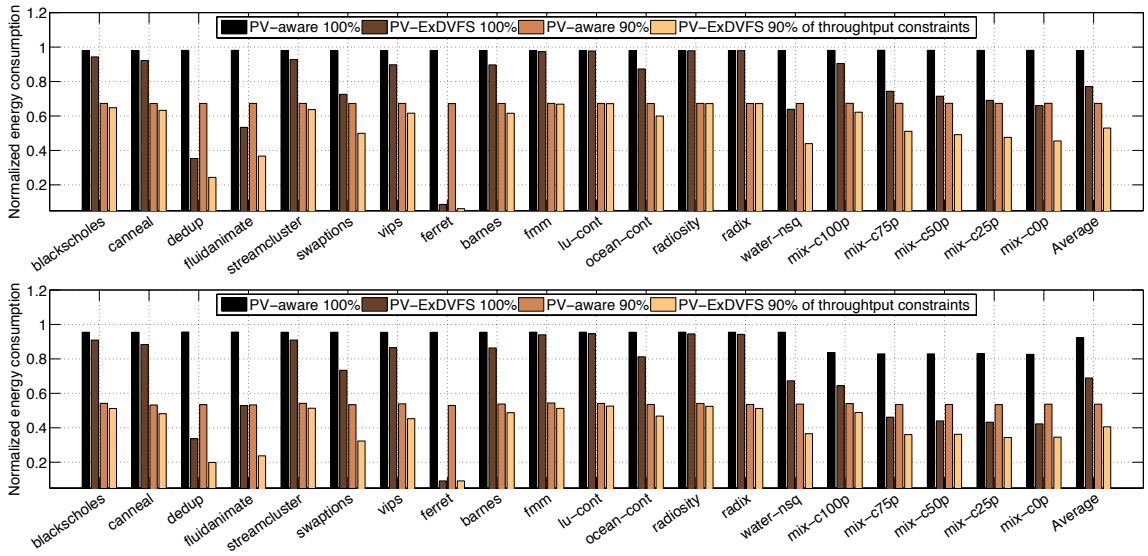

Figure 3.4: Average energy minimization results across 10 variation maps under 100% and 90% throughput constraints. **Top**: 3σ (standard deviation) of process parameters is 10% of their nominal values. PV-ExDVFS achieves the lowest energy points, the average reduction compared with the baseline approach is 22.88% for 100% throughput (hard) constraint and 47.13% for 90% throughput (soft) constraint. **Bottom**: 3σ is 30%. The corresponding energy reduction of PV-ExDVFS is 31.09% for hard constraint and 59.59% for soft constraint.

- **PV-aware**: the approach only integrating fine-grained PV information. In this case, cores run individually at the highest $\mathfrak{F}$ for the same V_{dd}. No dynamic adaptation is performed.

- **PV-ExDVFS**: the newly proposed approach based on fine-grained PV information in addtion to ExDVFS. In this case, $V_{dd}/\mathfrak{F}$ levels are customized for each core based on the PV information, and the optimal levels are chosen in a workload-aware fashion as described in Section 3.6.1 and 3.6.2.

We evaluate the energy minimization under iso-performance constraints by comparing the experimental results of the above approaches on different workloads and their mixtures from PARSEC and SPLASH-2 benchmark suites. The evaluations are based on region-of-interest (ROI) [13], excluding the initialization and cleanup phases for parallel benchmarks. Each workload mixture is composed of benchmarks randomly chosen from the two suites to represent a kind of heterogeneous multi-threaded workload. For example, *mix-c75p* means

75% of threads are core-bound while the rest are memory-bound. The classification is similar to the one used in [3][89]. The results of all approaches normalized by the baseline case (**Baseline**) are listed in Figure 3.4. All results are the average improvements across ten randomly generated variation maps. Furthermore, we evaluate the case with "soft" constraints. Namely, we set the total throughput at 90% of the baseline in Equation 3.10 to explore the case where we further reduce energy in exchange of only 10% of total throughput.

Figure 3.4(top) illustrates that PV-ExDVFS achieves the lowest energy points, the average reduction compared with the baseline approach is **22.88%** for 100% throughput (hard) constraint and **47.13%** for 90% throughput (soft) constraint. From the figure, we can see that PVs have a great impact on the power and performance in CMPs. Without ExDVFS algorithms, CMPs can still achieve **32.70%** energy reduction on average with knowledge of only process variation information for the soft constraint case, but the benefit increases by a half when ExDVFS is enabled. We note that among the PARSEC and SPLASH-2 benchmarks analyzed, *ferret* is the most imbalanced benchmark in terms of computation requirements per thread and therefore benefits the most from an extended DVFS paradigm. Indeed, in this case we observe a **10X** power reduction at iso-performance, while for 10% performance drop, a **20X** power reduction is observed. To further evaluate the effects of PV induced on CMPs, we perform similar comparisons on CMPs with extreme PV level (3σ = 30%). Comparing the top and bottom figures of Figure 3.4, we find that more energy is saved in all cases. For the hard constraints, an additional **5.68%** and **8.21%** are saved by PV-aware and PV-ExDVFS, respectively. For the soft constraints of 90% throughput, the corresponding additional savings are **13.58%** and **12.46%**.

3.6.2 Iso-power performance optimization

Similar to the analysis in Section 3.6.1, in this section we also evaluate the three approaches using the same baseline. We also consider the "soft" constraints. Namely, we increase the total power to 110% of the baseline in Equation 3.12 to find out how further to improve

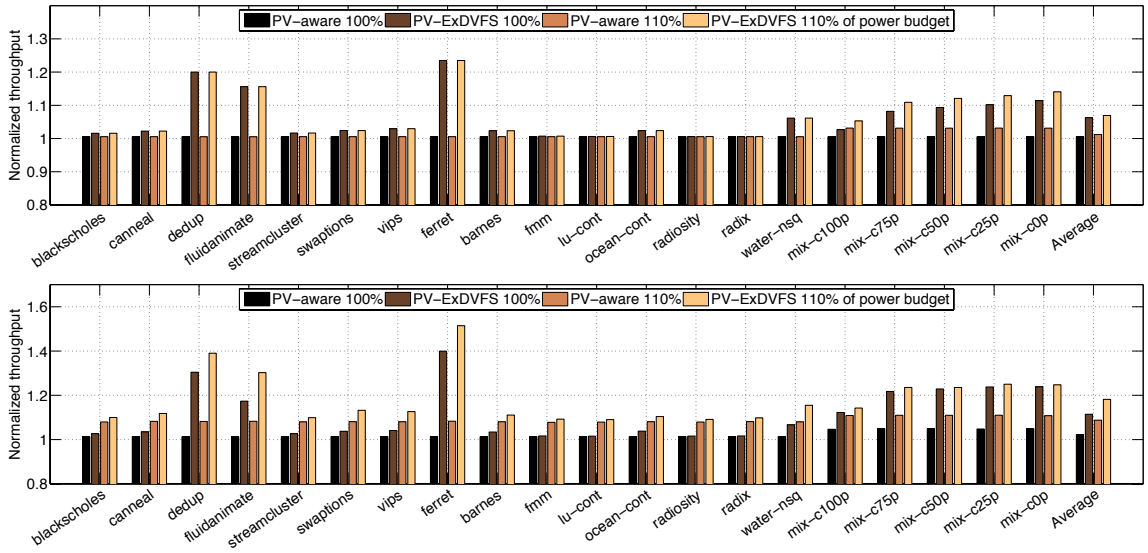

Figure 3.5: Average throughput improvement results across 10 variation maps with 100% and 110% power budgets. **Top**: 3σ (standard deviation) of process parameters is 10% of their nominal values. PV-ExDVFS achieves the highest throughput points, the average improvement compared with the baseline approach is 6.25% for 100% power (hard) budget and 6.93% for 90% power (soft) budget. **Bottom**: 3σ is 30%. The corresponding throughput improvement of PV-ExDVFS is 11.46% for hard constraint and 18.17% for soft constraint.

throughput in exchange of 10% more power consumption. The results of all approaches normalized by the baseline case are listed in Figure 3.5.

Figure 3.5(top) shows that the average improvement in the throughput in PV-ExDVFS are **6.25%** under 100% power budget (hard constraint) and **6.93%** under 110% power budget (soft constraint) when 3σ for PVs is 10%. For PV-aware approach, the performance increase is **1.54%** or **2.19%** with hard or soft power constraint, which appears to be a modest result for performance improvement. However, when the PV level increases to 3σ of 30%, the improvements become significant. Figure 3.5(bottom) illustrates the average improvements in throughput under the same constraints as in the top case. It is clear that PV-ExDVFS gains **11.46%** or **18.17%** in the hard or soft constraint cases respectively. Correspondingly, the gains of throughput for PV-aware approach also reach to **3.16%** or **8.78%** under hard or soft constraint, which are higher than the lower PV levels in Figure 3.5(top).

3.6.3 Impact of discrete V/F levels

PV-ExDVFS is based on the assumption that V/F levels are continuous across extended operating range. We have proven (given the convexity of the model) that PV-ExDVFS achieves the theoretical optimal result either in energy savings or throughput maximizations under given constraints. However, many algorithms or heuristics for energy savings or performance improvements are based on a few discrete V/F levels. Therefore, comparisons with state-of-the-art algorithms that operate under these assumptions help to quantify how far these algorithms are from the theoretical optimal provided by PV-ExDVFS. In this work, we discretize the normalized frequencies into four levels, $[0.05, 0.80, 1.00, 1.22]$, which represent the NTC, low, nominal and TB modes [23] [41]. In the iso-power throughput improvement part, MaxBIPS [40] guarantees optimal results for throughput maximization with discrete V/F levels. MaxBIPS is essentially an exhaustive search algorithm. While many heuristics exist for minimizing power under performance constraints, for fairness of results, we modify the MaxBIPS algorithm to perform iso-performance energy optimization through exhaustive search of state space. Therefore, we add PV effects to MaxBIPS and its modified version as the baseline for comparison. For further comparison, we also implement a PV-aware version of Steepest Drop [88], a state-of-the-art heuristic to accelerate throughput maximization under iso-power constraints. Combining the key idea of Steepest Drop and our formulations, we add PV effects and develop a heuristic for iso-performance energy savings. Eight-core CMPs are chosen because the run time for MaxBIPS is too large for systems with 16 cores or more. As stated before, to achieve fair comparisons, MaxBIPS and Steepest Drop are modified to include PV effects (denoted by PV-MaxBIPS and PV-Steepest Drop in the sequel).

Figure 3.6 shows the relative energy savings and throughput improvement for PV-ExDVFS and PV-Steepest Drop when compared with exhaustive search (PV-MaxBIPS). On average, PV-ExDVFS consume 9.50% less energy under iso-performance constraints than PV-MaxBIPS for mild ($3\sigma = 10\%$) PV level and 3.71% less for extreme ($3\sigma = 30\%$) PV level. In addition, PV-ExDVFS achieve 4.54% more throughput under iso-power constraints than

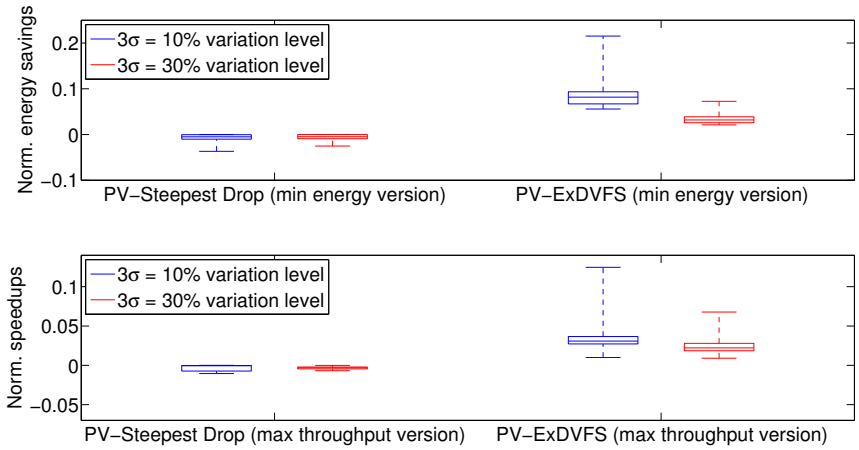

Figure 3.6: Energy savings (top) and speedups (bottom) for PV-Steepest Drop [88] and PV-ExDVFS compared with the baseline PV-MaxBIPS [40] *i.e.*, exhaustive search of best energy (top) and performance (bottom) configuration.

PV-MaxBIPS for mild ($3\sigma = 10\%$) PV level and 3.35% more for extreme ($3\sigma = 30\%$) PV level. We also observe that as one of the best heuristics for performance improvement under iso-power conditions, PV-Steepest Drop is about 1% away from PV-MaxBIPS, which is also stated in [40] for the simpler case of throughput improvement without PV effects. The comparison between PV-MaxBIPS and the theoretical limit given by PV-ExDVFS shows that up to 21.53% extra energy savings and 13.45% extra performance improvements are possible with increased number of V/F levels.

3.6.4 Impact of core cluster size

In this work, we target per-core DVFS, namely a core cluster size of one. However, not all existing platforms have this setting. Our algorithm is adaptable for other core cluster sizes. Indeed, by adding extra constraints on the final frequency/voltage choices (*i.e.*, same V/F for all cores in the same cluster), one can perform PV-ExDVFS for eight-core CMPs with core cluster size eight (chip-wise V/F domain), four, two, and one (per-core). By doing so, one can see the impact of core cluster size on our algorithm by comparing these four setups.

We set PV-ExDVFS for CMPs with one chip-wise V/F domain (core cluster size eight) as the baseline, by which all results from the other three setups are normalized by. The relative energy savings and throughput improvement results for the other three setups compared with the baseline are shown in Figure 3.7. As it can be seen, there is a steady increase for either energy savings or throughput improvement with decreased core cluster size. On average, per-core PV-ExDVFS (core cluster size equals one) save 17.75% more energy under mild PV level than the baseline and 22.84% more energy under extreme PV level. In addition, per-core PV-ExDVFS (core cluster size equals one) achieves 4.68% more throughput under mild PV level than the baseline and 8.45% more throughput under extreme PV level.

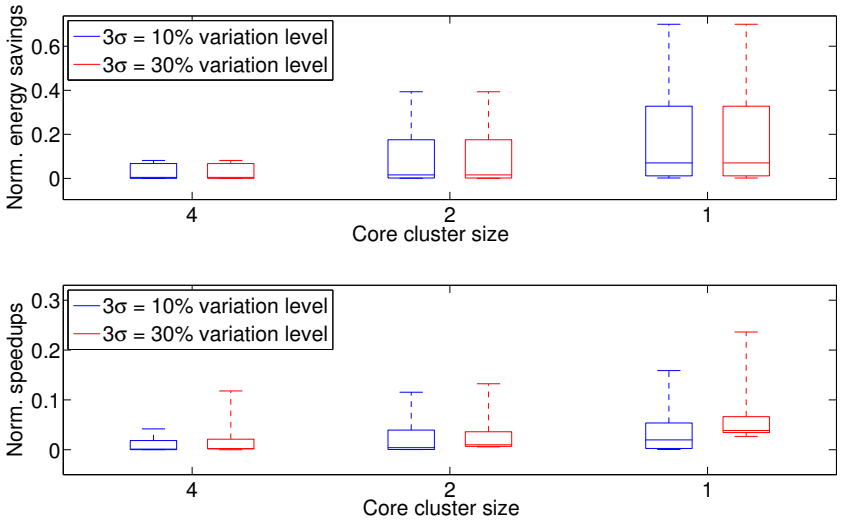

Figure 3.7: Energy savings (top) and speedups (bottom) for PV-ExDVFS with different core cluster sizes compared with the baseline (PV-ExDVFS with chip-wise V/F domain).

3.6.5 Comparisons of bulk CMOS and FinFET

In bulk-CMOS-based 20-core CMP, algorithms proposed in [84] achieve 12-17% increase in throughput with a given power budget. Our work in this chapter shows that in FinFET-based 16-core CMP, one can get 11.46% performance increase under power constraints in extreme PV level. In [44], 22.33% energy is saved through extended range DVFS under a given power budget in CMOS-based CMPs. In comparison, our framework reduces energy consumption

up to 31.09% under similar constraints, but with FinFET-based CMPs. This shows that with similar configurations, bulk CMOS and FinFET implementations produce benefits that trend similarly, with slightly better results achieved for FinFET designs. Furthermore, our work is **unique** in showing that these benefits significantly scale with core count and PV levels for FinFET implementations.

3.6.6 Effects of core count in FinFET CMPs

We also explore the effects of core count on ability of PV-ExDVFS to determine the optimal energy savings or performance improvement. We vary the core count from four all the way up to 128, with each configuration being run for ten variation maps. Figure 3.8(top) shows

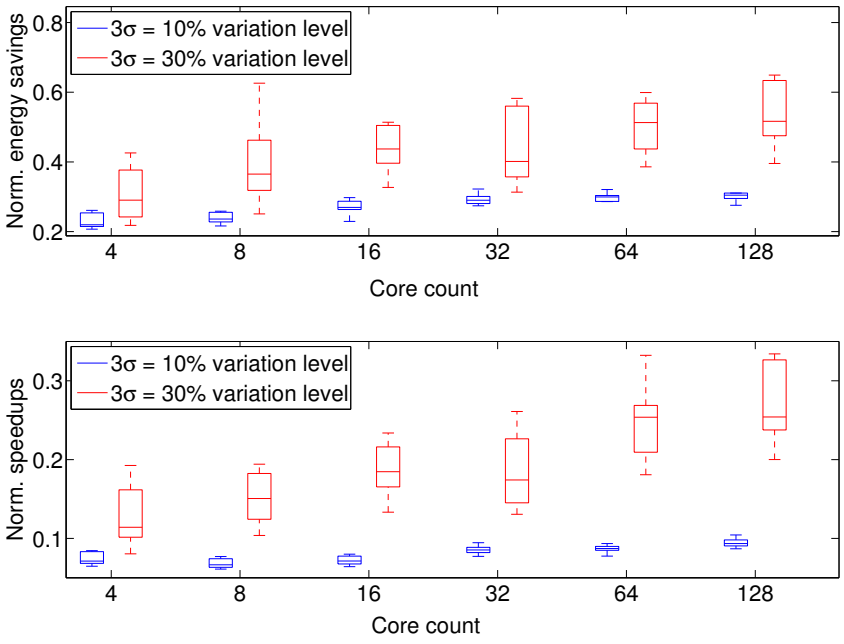

Figure 3.8: Energy saving and speedup further improve as number of cores and PV level increase. **Top**: energy savings under 100% throughput constraint in mild (10%) and extreme (30%) PV levels. **Bottom**: Speedups with 100% power budget in mild and extreme PV levels.

the optimal energy savings for given hard throughput constraints and Figure 3.8(bottom)

shows the optimal speedups[3] for given hard power budgets with each core count under each variation level. Figure 3.8 illustrates the improvements significantly increase as core count goes up, either for energy savings or speedups, subject to mild (10%) or extreme (30%) variation level. However, these increasing trends slow down when core count is large (over 32 in our experiments). For each fixed core count, PV-ExDVFS in the presence of extreme PVs provides better benefits than the mild PV one. In addition, extreme PV case introduces larger variance than the mild variation case. As see in Figure 3.8, one can conclude that as technology nodes keep decreasing (variation level is increasing) and core count continues to go up, PV-ExDVFS can save about **60% in energy** or increase **performance throughput** by **30%** without using any other technique. Furthermore, while the variance in savings is higher for extreme PVs (red, higher bars) than for mild ones (blue, lower bars), they are largely independent on the core count which demonstrate the stability of the results.

3.7 Discussion

In this chapter, we adapt the model-selecting technique and LOOCV from machine learning to learn the best PV-aware constrained-posynomial $\hat{P}$ for modeling the workload-dependent power-frequency relationship over an extended range for FinFET-based CMPs. Based on the convexity provided by the learned $\hat{P}$, two optimization frameworks are proposed: energy minimization under throughput constraints and throughput maximization under power constraints. Experimental results shows PV-ExDVFS achieves an average (1) **22.88%** or **31.09%** power reduction under iso-performance conditions and (2) **6.25%** or **11.46%** throughput improvement under iso-power conditions, with mild (3σ = 10%) or extreme (30%) PV levels, respectively. We validate the effectiveness of PV-ExDVFS by quantifying the impact of discrete V/F levels and core cluster sizes. In addition, further experiments **uniquely** show the benefits from our methods significantly increase with increasing core

[3]Speedup here is referred as normalized throughput improvement relative to the baseline case, assuming workloads are perfectly balanced.

count and variation level, either in energy savings or throughput improvements. Our work also shows that iso-performance energy savings or iso-power performance improvements in FinFET technology are similar to those obtained for bulk CMOS, but the effect of PVs magnifies these benefits in the case of FinFET implementations.

Chapter 4

Temperature effect inversion in multi-core systems

4.1 Chapter overview

Power and thermal issues have become the major constraints in developing high-performance microprocessors. Several hardware- and software-based techniques exist to resolve them. Among them, Dynamic Voltage and Frequency Scaling (DVFS) is widely used in both academia and industry, mainly to reduce the power consumption while maintaining the same performance. Upscaling of voltage/frequency (V/F) for multi-core processors has also been explored by processor manufacturers, *e.g.*, in Intel's Turbo Boost [15]. Similar techniques are also discussed in other work [88] [45]. For simplicity, we use the term "Turbo" in this chapter to refer to techniques that increase performance while keeping power consumption under a given budget. As the number of cores integrated in a single chip increases, the complexity of Turbo algorithms becomes one of the key points for real application.

CMOS technology evolution provides fundamental advances in mitigating power and thermal problems. FinFET has become the next generation CMOS technology of choice, especially for sub-20 nm technologies. Because of the different structures in planar vs. tri-dimensional CMOS devices, their power and performance characteristics are likewise

distinct [54]. One of these stark differences is the behavior of FinFET gate delay as a function of temperature. This phenomenon present in FinFET devices is called temperature effect inversion (TEI) [57] and refers to the reverse dependency between gate delay and temperature: gate delay *decreases* as temperature increases. In contrast, the gate delay of planar CMOS *increases* when temperature goes up. Figure 9 in [68] shows clearly that CMOS transistor delay grows as temperature increases in superthreshold voltage region[1].

The advantage of TEI is obvious: FinFET-based designs can operate faster at higher temperature. Therefore, it is important to quantify how TEI affects FinFET-based multi-core systems. Furthermore, existing Turbo algorithms are fast in current multi-core systems. However, they may not be fast enough for many-core systems comprised of hundreds of cores. Can we further reduce the runtime overhead without losing performance or energy efficiency for future large-scale multi-core systems? After performing a detailed analysis of TEI, our answer is yes.

In this chapter, we first characterize how much extra performance TEI brings to existing Turbo algorithms. After further analysis of TEI in multi-V/F level systems, we observe TEI-induced temperature-dependent "sweet spots", which are essentially locally best V/F levels. By exploiting these sweet spots first, one can greatly reduce the size of search space to decrease the runtime for performing DVFS. Based on these sweet spots, we propose fast algorithms, including Temperature Effect Inversion-aware Turbo Boost (TEI-Turbo) and its energy minimizing counterpart TEI-aware Low Power (TEI-LP) which we compare against corresponding state-of-the-art approaches.

4.1.1 Chapter contributions

Compared with previous state-of-the-art techniques, our work makes the following novel contributions:

[1]Supply voltage is larger than threshold voltage.

- Our work is the first to characterize the impact of TEI on power and performance of multi-core systems. Our experimental results show that on average **15.70%** throughput improvement or **31.26%** energy saving can be achieved in steady state for a TEI-aware DVFS policy over a TEI-agnostic one.

- We discover TEI-induced sweet spots in multi-level operating regions, which are uniquely associated with FinFET-based processors. These V/F sweet spots are strongly dependent on chip temperature.

- Based on these sweet spots, we propose a fast algorithm called TEI-Turbo, to determine the best performance under given power budget. This algorithm has a worst-case complexity of $O(nlog(n))$, which is better than state-of-art existing Turbo algorithms, like Steepest Drop [88].

- For power optimization, we propose an algorithm dubbed TEI-LP to determine the optimal power for a given throughput constraint. TEI-LP also has a worst-case complexity same as TEI-Turbo ($O(nlog(n))$).

- We evaluate our proposed algorithm on a multi-core simulator [13] with various multi-threaded benchmarks [3] [89]. Experimental results show that TEI-Turbo achieves an average of **45.9×** in runtime speedup and only **0.22%** loss of performance, when compared with TEI-aware version of existing state-of-the-art algorithms. Similarly, TEI-LP achieves an average of **55.3×** in runtime speedup and only **0.68%** more in energy consumption, when compared with TEI-aware version of existing state-of-the-art algorithms.

4.2 Performance and power models for TEI analysis

As the focus of this chapter is on multi-core systems, performance and power are the two main metrics to quantitatively characterize them. In this section, we introduce both performance metrics and power models used in this chapter below.

4.2.1 Performance

One of the key metrics to measure the performance of a multi-core system is the total throughput (TP), *i.e.*, the total number of instructions executed per unit time for all cores. Given the average number of instructions per cycle (IPC), and the operating frequency ($\mathfrak{F}$), for a chip-multiprocessor (CMP) with n cores, the total throughput can be calculated as:

$$TP = \sum_{i=1}^{n} IPC_i \cdot \mathfrak{F}_i \tag{4.1}$$

where i is the index for cores. IPC can be obtained at runtime from hardware counters (such as the Performance Counter Monitor by Intel [87]). The total throughput is essentially the aggregate instructions per second (IPS) for all cores.

4.2.2 Dynamic power

The power consumption in CMOS circuits can be divided into dynamic power and static power. Dynamic power for a core can be expressed as: $P_{dyn} = \alpha C_L V_{dd}^2 \cdot \mathfrak{F}$, where α is the activity factor, C_L is the capacitance for output load, and $\mathfrak{F}$ is operating frequency.

However, this model cannot be used as such in real systems. In addition, prior work [4] pointed out that dynamic power has strong correlation with IPC, since IPC approximately represents the activity rate (or utilization) of a processing core. Therefore, similar to [4], we assume a linear dependency to characterize dynamic power for each core:

$$P_{dyn} = V_{dd}^2 \cdot \mathfrak{F} \cdot (a \cdot IPC + b) \tag{4.2}$$

where a and b are coefficients that need to be determined.

To obtain the model in Equation 4.2, we use the multi-core system-level performance simulator Sniper [13] and power simulator McPAT [58] to collect data. We run multi-threaded benchmarks from PARSEC [3] and SPLASH-2 [89] on FinFET-based CMPs under three different V/F states (nominal: 0.85 V/3.0 GHz, high: 1.05 V/3.5 GHz, and low: 0.55 V/1.7 GHz).

In addition, we develop a machine learning-based approach to learn the best model for each benchmark. The process is divided into two phases: training phase and cross-validation phase [5]. We implement a ten-fold cross validation approach [5] to learn the best coefficients for Equation 4.2. Generally, the data are divided randomly into ten equal subsets. The first one is chosen as the cross-validation (testing) set, the rest as training set. For the training phase, a linear regression is performed on the training set to obtain the coefficients (a, b). In the validation phase, this set of coefficients are applied to the validation set to obtain the squared percentage error for each data point. We repeat this process for all ten subsets, so that each data point is associated with a squared percentage error. Based on these, we can easily obtain the Root-Mean-Square-Percentage-Error (RMSPE) in the validation phase.

Table 4.1 shows the RMSPE for the validation phase of the dynamic power model in Equation 4.2. The average RMSPE across all benchmarks in Table 4.1 is 6.47%.

Table 4.1: Validation error of the learned dynamic power.

Name	blks.	bdyt.	cnnl.	ddp.	fcsm.	frrt.
RMSPE(%)	4.14	14.01	2.68	4.91	4.36	9.25
Name	fldn.	frqm.	rytr.	strm.	swpt.	vips
RMSPE(%)	3.22	2.33	4.09	6.98	3.71	14.06
Name	x264	brns.	chlk.	fft	fmm	lu
RMSPE(%)	7.33	6.29	11.81	3.65	8.73	3.49
Name	ocn.	rdst.	rdx	wtr.	**AVERAGE**	
RMSPE(%)	11.46	8.35	3.64	3.99	**6.47**	

4.2.3 Static power

For planar CMOS devices, the leakage-temperature loop is well known: when temperature increases, leakage power increases, which further determines the temperature to increase,

and so on so forth. The change in leakage power in this loop is mainly caused by subthreshold leakage. What happens in the case of FinFET devices? The leakage-temperature loop still exists, but is affected by TEI effect. Figure 4.1 shows the leakage power for 16 nm FinFET two-stage FO4 inverter chains. Several models have been proposed to characterize the leakage power of FinFET circuits with respect to V_{dd} and temperature [75].

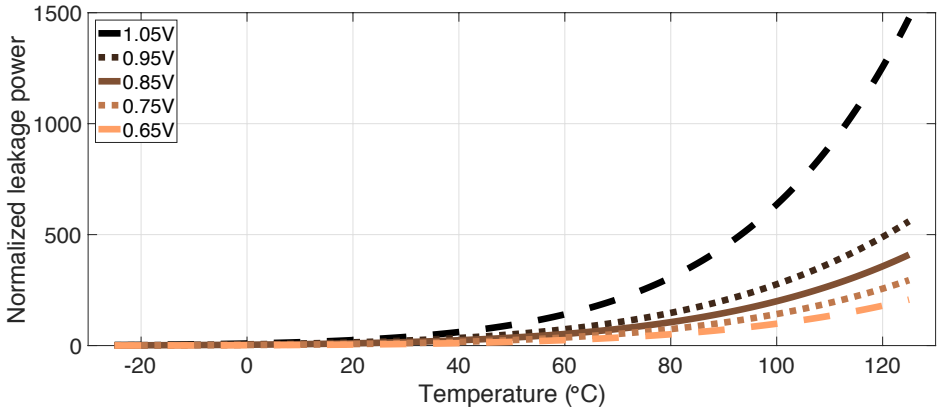

Figure 4.1: Normalized leakage power under a wide range of temperature values for FinFET.

In multi-core systems, we propose a simple system-level model to characterize static power for cores. The static power can be expressed as:

$$P_{sta} = V_{dd} \cdot C(\theta) = V_{dd} \cdot c_0 \cdot \exp(c_1 \cdot \theta^2 + c_2 \cdot \theta + c_3) \tag{4.3}$$

where c_0 is the normalization factor. c_1, c_2 and c_3 are the coefficients to determine. c_1 models the nonlinear part of $log(P_{sta})$ with respect to temperature (θ), while c_2 and c_3 model the corresponding linear part. The reason for choosing a quadratic function in the exponent instead of a linear one is that simulation data shows $log(P_{sta})$ behaves as a sublinear function of θ. A quadratic function with negative value for c_1 can accurately model this trend.

To validate this model, we collect leakage power data from the widely used system-level power simulator McPAT [58]. We modify McPAT to make it support 16 nm FinFET. We update the subthreshold current (I_{sub}) model with SPICE simulation based on PTM models. Similar to the process in Section 4.2.2, we implement cross validation techniques [5] to train and validate the model in Equation 4.3 across a wide temperature range (from -25 °C to 125

°C) and voltage range (from 0.45 V to 1.05 V). The results shown in Table 4.2 illustrate that the RMSPE is only 1.67% in a ten-fold cross-validation process. 1.67% indicates the Equation 4.3 is an accurate system-level static power model, especially considering the wide operating temperature and voltage range.

Table 4.2: Learning leakage power model.

	c_0	c_1	c_2	c_3	RMSPE
Value	0.154	-0.000101	0.0498	-2.50	1.67%

4.3 Optimization methodology

In this section, we first discuss the TEI-induced dynamic voltage-frequency pairing. We then introduce the concepts of "sweet spots" resulting from TEI. Based on that, we discuss two fast algorithms: (i)TEI-Turbo for performance boosting under given power budgets and (ii) TEI-LP for energy saving with given throughput constraints. Finally, we discuss the complexity of the proposed algorithms.

4.3.1 Temperature-dependent V/F pairs

Due to TEI, it is important to quantify the effect of temperature on voltage/frequency pair selection. Prior work [33] has used the frequency of a ring oscillator (RO) as a sufficiently accurate approximation for the frequency of a core. Similarly, we use a RO to quantify the TEI effect on V/F pairing. We apply the ten-fold cross validation method to perform model selection for the maximum operating frequency $\mathfrak{F}_{max}$ for a given supply voltage V_{dd} with given temperature θ. When tested with a large set of various models, the model shown in Equation 4.4 achieves the best cross-validation error, which is the result of model selection technique [5]. In Equation 4.4, d_0 models the nonlinear part of $\mathfrak{F}_{max}$ with respect to V_{dd}, while d_1 quantifies the coupling effect from V_{dd} and θ on $\mathfrak{F}_{max}$. d_2, d_3, and d_4 are coefficients

that model the linear part from V_{dd} and θ.

$$\mathfrak{F}_{max} = d_0 \cdot V_{dd}^2 + d_1 \cdot V_{dd} \cdot \theta + d_2 \cdot \theta + d_3 \cdot V_{dd} + d_4 \tag{4.4}$$

Table 4.3 shows the values of the coefficients and the RMSPE for the test phase. The results confirm the accuracy of our temperature-dependent V/F pairing model.

Table 4.3: Learning temperature-dependent V/F pairs.

	d_0	d_1	d_2	d_3	d_4	RMSPE
Value	-4.27	0.0042	0.0052	10.6	-2.66	0.49%

Since V/F levels are always discrete in real multi-core systems, Equation 4.4 can not be directly used. Instead, it provides a direct guidance for the dynamic matching, namely the smallest available V_{dd} whose $\mathfrak{F}_{max}$ is larger than the required frequency at a given temperature θ.

To illustrate these guidelines, let us determine a set of V/F levels first. Like in prior work [33], we use a RO to determine V/F levels other than nominal (0.85 V, 3.0 GHz). Due to TEI, the worst-case timing happens at the lowest temperature (-25 °C in this chapter). Therefore, one V/F-pairing example is illustrated in Table 4.4 for the worst-case delay (-25 °C). This example includes 13 V/F pairs, which is close to typical values for Intel Xeon processors (15 V/F levels on Xeon E5-2670 processor [50]).

Table 4.4: V/F levels for worst case (-25 °C).

Level	0	1	2	3	4	5	6
$\mathfrak{F}$	1.05	1.40	1.70	2.00	2.25	2.45	2.65
V_{dd}	0.45	0.50	0.55	0.60	0.65	0.70	0.75
Level	7	8	9	10	11	12	
$\mathfrak{F}$	2.85	3.00	3.15	3.30	3.40	3.50	
V_{dd}	0.80	0.85	0.90	0.95	1.00	1.05	

As temperature changes, the V/F pairings are different, as shown by Equation 4.4. The process is shown in Figure 4.2.

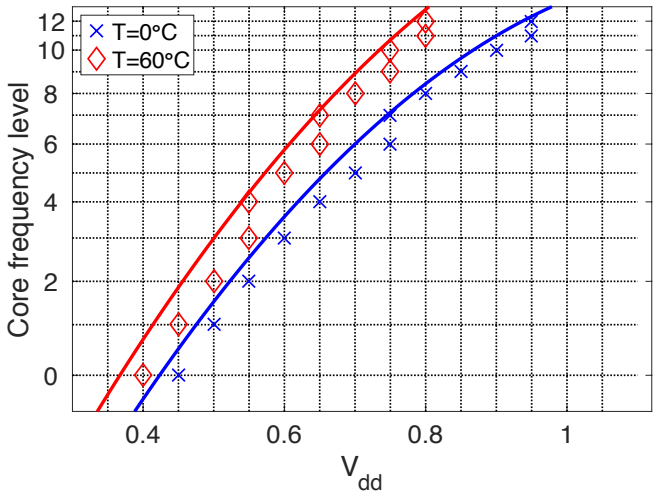

Figure 4.2: Dynamic voltage frequency pairings for different temperatures.

4.3.2 TEI-induced sweet spots

Energy efficiency (EE) is defined as throughput per unit power consumption, as shown for any given core in Equation 4.5.

$$\begin{aligned} EE =& \frac{TP}{P_{total}} = \frac{IPC \cdot \mathfrak{F}}{V_{dd}^2 \cdot \mathfrak{F} \cdot (a \cdot IPC + b) + C(\theta) \cdot V_{dd}} \\ =& \frac{1}{V_{dd}} \cdot \frac{1}{a \cdot V_{dd} + \frac{b \cdot V_{dd}}{IPC} + \frac{C(\theta)}{IPC \cdot \mathfrak{F}}} \end{aligned} \tag{4.5}$$

where $a, b > 0, C(\theta) > 0, IPC > 0$.

By studying EE in Equation 4.5, we can see that energy efficiency is dependent on V_{dd}, $\mathfrak{F}$, θ, and IPC. As V_{dd} increases, EE decreases. However, the trend is opposite when looking at $\mathfrak{F}$. We can easily see that EE generally increases as V/F levels are decreased and therefore, the global best EE is achieved near the lowest V/F levels. However, the lowest V/F levels are characterized by very low performance levels which are unlikely to meet requirement or make use of available power budget. Therefore, to meet the performance requirement or exploit the power budget, it becomes desirable to search operating V/F levels with better EE locally, especially around the performance requirement or power budget.

With TEI, we observe the sweet spots, which are the operating V/F levels with relatively better EE in their local ranges. Later in the chapter (Section 4.3.3), we will see that these

sweet spots can help us find the better EE quickly around the requirement or power budget. The V/F level j is defined as a sweet spot (SS) if it satisfies the following conditions:

$$EE'_j = EE_{j-1} + (EE_{j+1} - EE_{j-1}) \cdot \frac{f_j - f_{j-1}}{f_{j+1} - f_{j-1}} \tag{4.6}$$

$$\frac{EE_j - EE'_j}{EE'_j} > threshold \tag{4.7}$$

where $threshold$ is 5% in this chapter. EE'_j is the value interpolated from the adjacent two states. The rationale behind this definition is that EE is generally decreasing as V/F level increases. TEI effect can slow down this decreasing trend or even reverse it for some local states. This interpolation method plus a certain threshold can select these states as sweet spots.

Figure 4.3 shows the normalized energy efficiencies for different frequency levels. From the figure we can see that Level 1, 4, 7, 10 will be chosen as SSs.

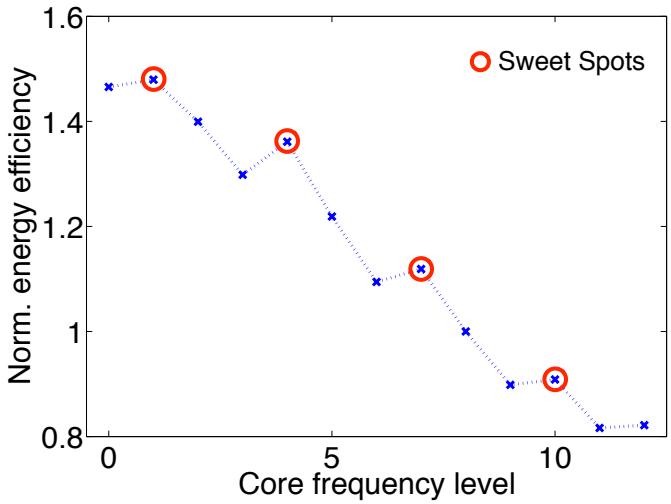

Figure 4.3: Normalized energy efficiency across all frequency levels.

Let us consider Figure 4.4 as an example to illustrate the process of determining sweet spots at a given temperature. Please note the V/F level numbers mentioned here are referring only to the frequency levels, since the corresponding voltage levels change due to TEI. For example, when temperature increases to 42 °C, V_{dd} at Level 8 drops from 0.75 V to 0.7 V (which the red/rightmost arrow indicates in Figure 4.4), while V_{dd} at Level 7 remains 0.7 V.

The formula in Equation 4.5 clearly shows that the drop in V_{dd} will increase the EE of Level 8. At this temperature, Level 8 has the same V_{dd} as Level 7, but with higher frequency than Level 7. According to Equation 4.5, these levels have comparable EE values. Level 9 gets a voltage drop to 0.75 V at 33 °C (which the blue/leftmost arrow indicates in Figure 4.4), and stays at 0.75 V around 42 °C. There is no change of its EE at 42 °C. Therefore, the clear increase of EE for Level 8 will designate it as a sweet spot based on Equation 4.7. We can also see that when temperature is between 33 °C and 42 °C, Level 9 becomes a sweet spot instead of Level 8.

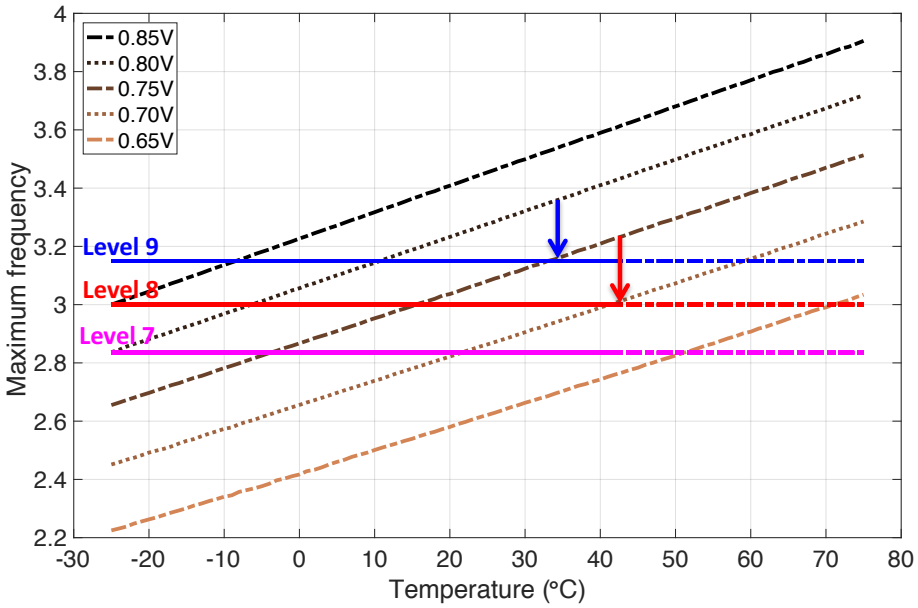

Figure 4.4: Example process of determining sweet spots.

From the example above, we can conclude that: (1) sweet spots are based on relative power-performance difference between adjacent levels, therefore are highly local; (2) sweet spots vary as operating temperature changes.

4.3.3 Efficient TEI-aware V/F scaling

Due to thermal constraints, power is always subject to certain requirements. Power-limited multi-core systems with increasing number of cores are likely to dominate future designs

[42]. Therefore, an efficient online algorithm to determine the optimal throughput under a given power budget becomes a necessity.

However, performance optimization via V/F scaling can be formulated as a NP-hard problem [46], which is what MaxBIPS [40] implements via exhaustive state search. One of the state-of-the-art efficient heuristics is Steepest Drop [88], which is based on a directed local search method. The Steepest Drop algorithm has computational complexity of $O(mnlog(n))$, where m is the number of V/F levels, n is the number of cores. Can we do better in runtime, but without much performance loss? Based on the observation of sweet spots in Section 4.3.2, we propose a fast algorithm, dubbed TEI-Turbo as shown in Algorithm 1. TEI-Turbo is essentially making use of TEI-induced sweet spots to efficiently determine the V/F level and then fine-tunes it to reach the final decision. It consists of three phases which are detailed below.

The first phase (line 4 - line 11) has two functions. One is to identify idle cores and set them into the lowest V/F state, while the other is to put all active cores into a heap ranked by utilization (as discussed in Section 4.2.2, IPC is used as a proxy for utilization). The rest of TEI-Turbo algorithm only focuses on active cores, therefore total power budget is reduced by the power for those idle cores (line 7).

The second phase (line 12 - line 28) calculates sweet spots based on current temperature and chooses V/F levels so resulting power fits within updated power budget. Based on the method discussed in Section 4.3.2, we determine the sweet spots for the current temperature. SS in line 12 is a list of all sweet spots, including the highest and lowest levels, which determine the lower and upper bounds for power. We then calculate the power consumption of all active cores in the same state listed in SS ($PwSS$). If the available power budget is between two adjacent power values in $PwSS$, then the higher sweet spot state will be chosen (line 20 - line 25).

Algorithm 1 TEI-Turbo

1: Input: IPC, $Budget$, N, θ
2: Output: VF levels for all cores
3: *active_cores*: a min-heap for active cores ranked by IPC
4: **for** $i = 0; i < N; i++$ **do**
5: **if** $IPC_i == 0$ **then**
6: $VF_i = lowest$
7: $Budget \leftarrow Budget - Power_{i,lowest}$
8: **else**
9: Push (i, IPC_i) to the heap *active_cores*
10: **end if**
11: **end for**
12: SS: all sweet spots, from highest to lowest levels
13: $PwSS$: total power for cores in each SS state
14: **if** $Budget \geq PwSS[first]$ **then**
15: $VF_i = highest$, $\forall i$ in *active_cores*
16: **return**
17: **else if** $Budget < PwSS[last]$ **then**
18: **return** CASE_INFEASIBLE
19: **else**
20: **for** $i = 0; i < length(SS) - 1; i++$ **do**
21: **if** $PwSS[i] > Budget > PwSS[i+1]$ **then**
22: $s\_c = i$
23: $VF_j = SS[s\_c]$, $\forall j$ in *active_cores*
24: $active\_power = PowerSS[s\_c]$
25: **Break**
26: **end if**
27: **end for**
28: **end if**
29: **while** `True` **do**
30: Pop the core i with smallest IPC in *active_cores*
31: Downgrade core i from state $SS[s\_c]$ to $SS[s\_c+1]$
32: $temp\_power \leftarrow$ active power after this downgrade
33: **if** $temp\_power \geq Budget$ **then**
34: $VF_i = SS[s\_c+1]$
35: $active\_power \leftarrow temp\_power$
36: **else**
37: **for** $j = SS[s\_c] - 1; j \geq SS[s\_c+1]; j--$ **do**
38: Downgrade from state $SS[s\_c]$ to j
39: $temp\_power \leftarrow$ active power after downgrade
40: **if** $temp\_power \leq Budget$ **then**
41: $VF_i = j$
42: **return**
43: **end if**
44: **end for**
45: **end if**
46: **end while**

The final phase (line 29 - line 46) is a fine-tuning process to find the V/F level in a best effort manner. It starts from V/F choice from the second phase and tries to downgrade the core with the lowest IPC from current SS state to next lower SS state, until power budget is satisfied. Since there are several other states between SS states, we need to divide the last downgrade step which aims to meet power budget into several steps checking all intermediate states between the two adjacent SS states (line 37 - line 44). After this process, TEI-Turbo returns the best found V/F level decision.

Algorithm 2 TEI-LP

```
1: Input: IPC, throughput_constr, N, θ
2: Output: VF levels for all cores
3: active_cores: a max-heap for active cores ranked by IPC
4: for i = 0; i < N; i++ do
5:     if IPC_i == 0 then
6:         VF_i = lowest
7:     else
8:         Push (i, IPC_i) to the heap active_cores
9:     end if
10: end for
11: SS: all sweet spots, from highest to lowest levels
12: TP_SS: total throughput for cores in each TP_SS state
13: if throughput_constr ≥ TP_SS[first] then
14:     return CASE_INFEASIBLE
15: else if throughput_constr < TP_SS[last] then
16:     VF_i = lowest, ∀i in active_cores
17:     return
18: else
19:     for i = 0; i < length(SS) − 1; i++ do
20:         if TP_SS[i] > throughput_constr > TP_SS[i + 1] then
21:             s_c = i + 1
22:             VF_j = SS[s_c], ∀j in active_cores
23:             curr_tp = TP_SS[s_c]
24:             Break
25:         end if
26:     end for
27: end if
28: while True do
29:     Pop the core i with largest IPC in active_cores
30:     Upgrade core i from state SS[s_c] to SS[s_c − 1]
31:     temp_tp ← temporary throughput after this upgrade
32:     if temp_tp < throughput_constr then
33:         VF_i = SS[s_c − 1]
34:         curr_tp ← temp_tp
35:     else
36:         for j = SS[s_c] + 1; j ≤ SS[s_c − 1]; j++ do
37:             Upgrade from state SS[s_c] to j
38:             temp_tp ← temporary throughput after upgrade
39:             if temp_tp ≥ throughput_constr then
40:                 VF_i = j
41:                 return
42:             end if
43:         end for
44:     end if
45: end while
```

In addition, the ideas of the sweet spots can also be applied in the dual problem: the minimum power consumption to meet the throughput constraints. We name this as TEI-LP algorithm, the low-power mode of TEI-aware algorithm.

The pseudo code for TEI-LP is shown in Algorithm 2. The general idea of TEI-LP is to make use of sweet spots to find the minimum power consumption in a fast way with a given throughput constraint.

The first phase (line 4 - line 10) has two functions. One is to identify idle cores and set them into the lowest V/F state, while the other is to put all active cores into a heap ranked by utilization (as discussed in Section 4.2.2, IPC is used as a proxy for utilization). The rest of TEI-LP algorithm only focuses on active cores, which is similar as TEI-Turbo.

The second phase (line 11 - line 27) finds the sweet spots based on current temperature and chooses V/F levels to meet the required throughput. Based on the method discussed in Section 4.3.2, we determine the sweet spots for the current temperature. SS in line 11 is a list of all sweet spots, including the highest and lowest levels, which determine the lower and upper bounds for throughput. We then calculate the total throughput of all active cores in the same state listed in SS (TP_SS). If the throughput requirement is between two adjacent throughput values in TP_SS, then the lower sweet spot state will be chosen (line 19 - line 26).

The final phase (line 28 - line 45) is a fine-tuning process to find the V/F level in a best effort manner. It starts from V/F choice from the second phase and tries to upgrade the core with the highest IPC from current SS state to next higher SS state, until throughput requirement is met. Since there are several other states between the two SS states, we need to divide the last upgrade step which aims to meet throughput into several steps checking all intermediate states between the two adjacent SS states (line 36 - line 43). After this process, TEI-LP returns the best found V/F level decision.

4.3.4 Complexity analysis

The similar structure of TEI-Turbo and TEI-LP indicates that they have the same complexity. Therefore, we only need to analyze the complexity of TEI-Turbo.

In our proposed algorithm TEI-Turbo, the three stages are separate. Therefore, we can analyze them one by one. The first stage is essentially composed of a step iterating across all n cores and a heap sort of no more than n cores. Thus, its complexity is $O(nlog(n))$. The second stage iterates over m V/F levels. Its complexity is $O(m)$. In the third stage, the

major loop iterates over n cores, resulting in a complexity of $O(n)$. In real systems, m is usually much smaller than n in real large systems. Therefore, we conclude that TEI-Turbo has a worst-case complexity of $O(nlog(n))$. Similarly, we can conclude that TEI-LP has a worst-case complexity of $O(nlog(n))$.

In summary, both TEI-Turbo and TEI-LP have a runtime complexity of $O(nlog(n))$. For comparison, the Steepest Drop algorithm has a runtime complexity of $O(mnlog(n))$.

4.4 Experimental results

In this section, we first introduce the experimental setup. we then discuss how TEI affects the state-of-the-art existing algorithm for performance improvement or energy saving. Finally, we comprehensively evaluate our proposed algorithms against state-of-the-art algorithms.

4.4.1 Experimental setup

We use the Sniper [13] multi-core simulator as the performance simulator. The default architecture used in this chapter is listed in Table 4.5. The voltage/frequency scaling control epoch is set as 1 ms. Each core runs one thread, without any context switches or simultaneous multi-threading. Furthermore, DVFS is performed independently on each core. Unless stated

Table 4.5: Target architecture.

Parameters	**Values**
Number of cores	16
Number of core states	13
Nominal frequency	3.0 GHz
Core model	Intel Nehalem-like
L2 caches	Private 256KB, 8-way SA, LRU
L3 caches	Shared 32MB, 16-way SA, LRU
DRAM	8GB
Technology	16 nm FinFET

otherwise, the experimental results are based on the default architecture. We modify the

system-level power simulator McPAT [58] as mentioned in Section 4.2.3 to support 16 nm technology node with an accurate temperature-leakage model.

With respect to thermal effects, we use HotSpot [38] to obtain the die temperature. To implement a real-time temperature feedback to simulate leakage, we modify HotSpot to run one time during each control epoch to obtain instantaneous die temperatures throughout all simulations. Based on area estimation provided by McPAT, we set up the floorplan for the 16-core CMP (with detailed configurations for each core) as in Figure 4.5. The total die area is 126.6 mm^2. In the Hotspot configuration, the heat spreader size is set to 0.013m×0.013m based on the chip size. The heat sink size is set to 0.026m×0.026m accordingly. The ambient temperature is 27 °C. The parameters not mentioned here are left unchanged as the default values. In this work, the results are based on running multi-threaded benchmarks one at a

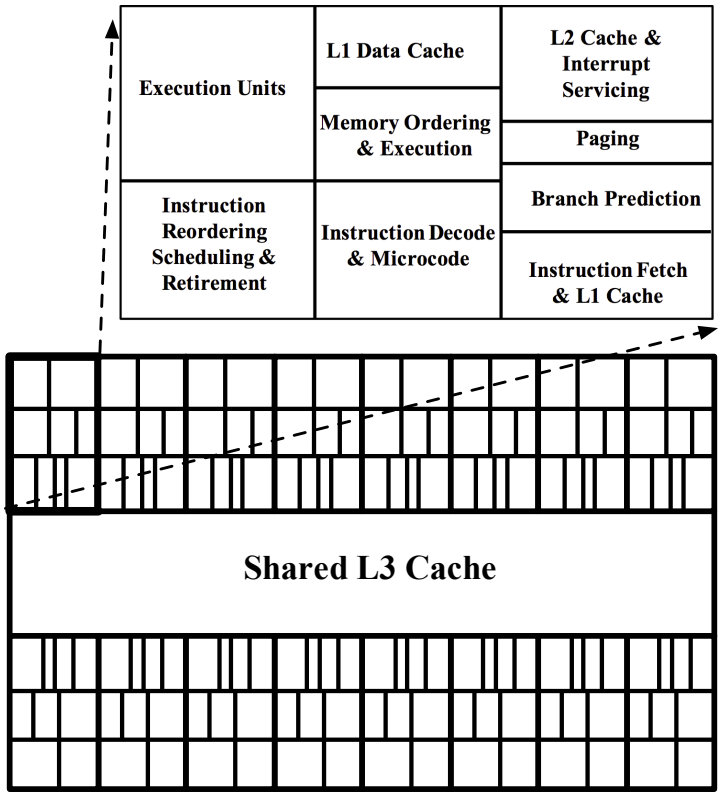

Figure 4.5: Simplified floorplan for a 16-core nehalem-like [2] chip-multiprocessor with only cores and L3 cache.

time from PARSEC [3] and SPLASH-2 [89] on Sniper from start until completion, except explicit statements.

4.4.2 The impact of TEI

From the discussion of TEI in Section 4.3.1, we can conclude that the TEI effect makes the V/F pairs no longer fixed at different operating temperatures. However, we assume that in a control epoch, the temperatures across CMPs are stable. Therefore, V/F pairs are fixed for each control epoch. Since V/F pairings are based on the temperature, we assume that the temperature during each control epoch is stable. In reality, the temperature values are always changing. To quantify the temperature changes for each core, we test with various benchmarks and different temperature scenarios. The results are shown in Figure 4.6, with the mean as 0.15 °C and the median as 0.03 °C. Figure 4.6 shows that most of the temperature changes during one control epoch are less than 1.0 °C. In rare cases, the change is over 1.0 °C, but less than 2.5 °C, with the maximum value as 2.33 °C. To implement V/F

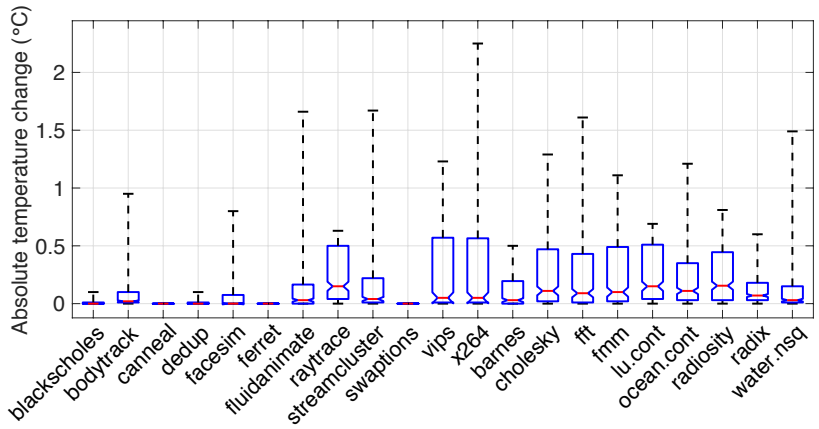

Figure 4.6: The absolute temperature changes for each core during one control epoch (1 ms) for different benchmarks.

pairings in real system, an appropriate margin (*e.g.* 2.5 °C) should be added for the transition temperatures. Since the margin would be added uniformly throughout all the transition temperatures, there would be no effect on the experimental results shown later in this chap-

ter. In addition, thanks to our proposed algorithm, the control epoch can be shortened without significant overhead. Therefore, the temperature margin could be even smaller.

Performance improvement

To illustrate the impact of TEI in performance improvement under a given power budget, we implement Steepest Drop [88], a heuristic to determine the optimal throughput under a given power budget. For comparison, we set the original version of Steepest Drop as the Baseline, namely TEI-unaware Steepest Drop. We combine TEI and Steepest Drop to define a new policy: TEI-aware Steepest Drop. We set the same power budget for both policies and compare the total throughput in each case for different temperatures due to temperature-dependence for TEI. The results of total throughput for TEI-aware Steepest Drop are normalized with respect to the corresponding TEI-unaware Steepest Drop baseline.

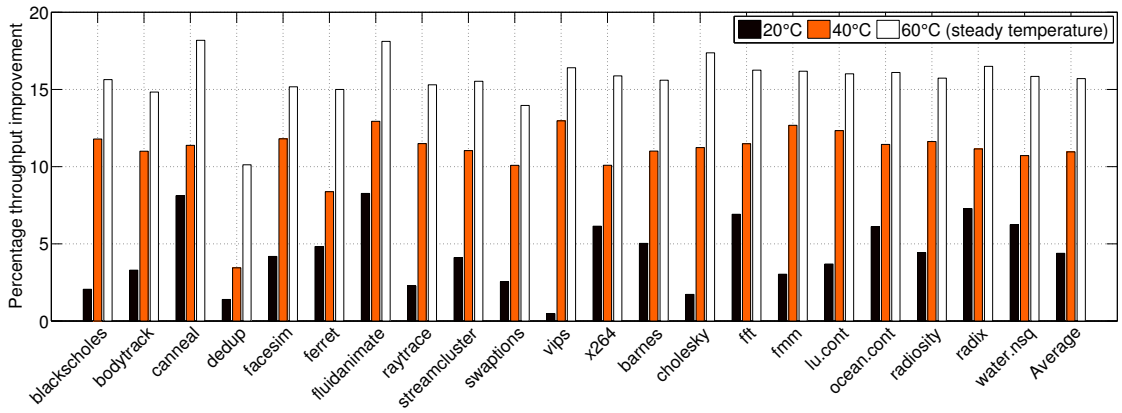

Figure 4.7: Normalized throughput improvement of TEI-aware Steepest Drop algorithm over TEI-unaware version.

Figure 4.7 shows the relative throughput improvement of TEI-aware Steepest Drop over the TEI-unaware case. From Figure 4.7, we can easily see that the relative throughput improvement brought by the TEI-aware policy increases when temperature goes up. The average throughput improvement is 4.39% under 20 °C, and 10.95% under 40 °C. When CMPs work in steady state (under current power budgets, steady state is around 60 °C), the TEI-aware Steepest Drop achieves an average of 15.70% performance improvement over TEI-unaware Steepest Drop. Therefore, we can see that, even without changing current policies,

TEI can still bring an extra throughput enhancement of around **15.70%** on average in steady state.

Energy saving

To quantify the impact of TEI in energy saving with a given throughput constraint, we implement a directed local search algorithm named Steepest Rise, inspired by Steepest Drop [88]. Steepest Rise here is a heuristic to determine the nearly optimal energy consumption with given throughput constraint. It starts with the lowest V/F states and chooses to upgrade the core with the highest ratio of performance difference over energy difference. This is repeated until the performance is met or all the cores are in the highest states. For comparison, we set the TEI-unaware Steepest Rise as the Baseline. We combine TEI and Steepest Rise to develop a new policy: TEI-aware Steepest Rise. We set the same throughput constraint for both policies and compare the total energy consumption in each case for different temperatures similar to Section 4.4.2.

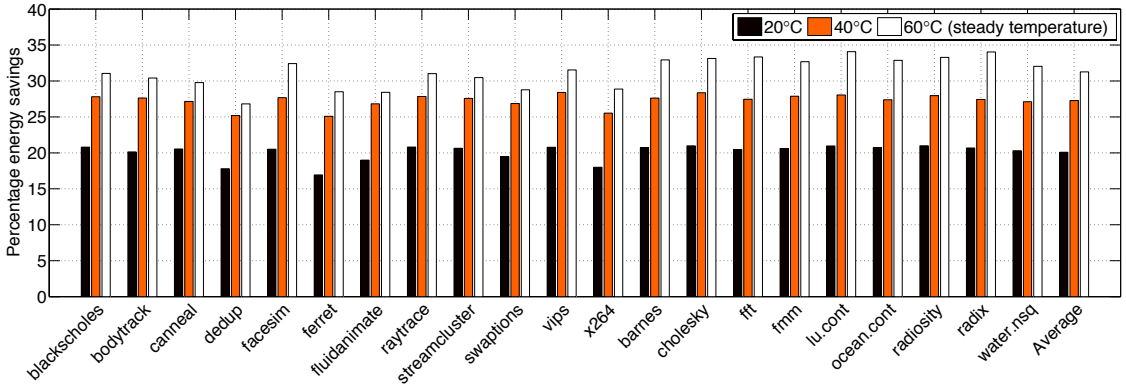

Figure 4.8: Normalized energy saving of TEI-aware Steepest Rise algorithm over TEI-unaware version.

Figure 4.8 shows the extra energy saving of TEI-aware Steepest Rise over the TEI-unaware case. From Figure 4.8, we can see that higher temperature yields better saving in energy consumption. The average energy saving is 20.09% under 20 °C, 27.28% under 40 °C, and 31.26% under 60 °C. Therefore, without changing current policies, TEI still brings an extra energy saving of around **31.26%** in steady state (around 60 °C).

4.4.3 Evaluation of TEI-aware algorithms

In this section, we evaluate the TEI-aware algorithms (TEI-Turbo and TEI-LP) proposed in Section 4.3.3 by comparing their performance, energy, runtime, and energy efficiency against existing state-of-the-art algorithms. The evaluation is divided into two separate parts: one for TEI-Turbo and the other for TEI-LP.

TEI-Turbo algorithm

After quantifying the impact of TEI on Steepest Drop, we set TEI-aware Steepest Drop as the Baseline to evaluate our proposed algorithms TEI-Turbo in this section. The reason why we don't use MaxBIPS as the baseline is because MaxBIPS is too slow to be an option for an online V/F scaling policy in the experiment setting with 16 cores and 13 V/F levels.

We also quantify how much performance is lost by the use of our heuristics vs. existing state-of-the-art algorithms based on Steepest Drop. Thus, we first compare the throughput each algorithm achieves. We run different benchmarks with default setup under steady state operation. Figure 4.9 shows the relative throughput for TEI-Turbo compared with the baseline, TEI-aware Steepest Drop. On average, the throughput of TEI-Turbo is only **0.22%** less than the Baseline.

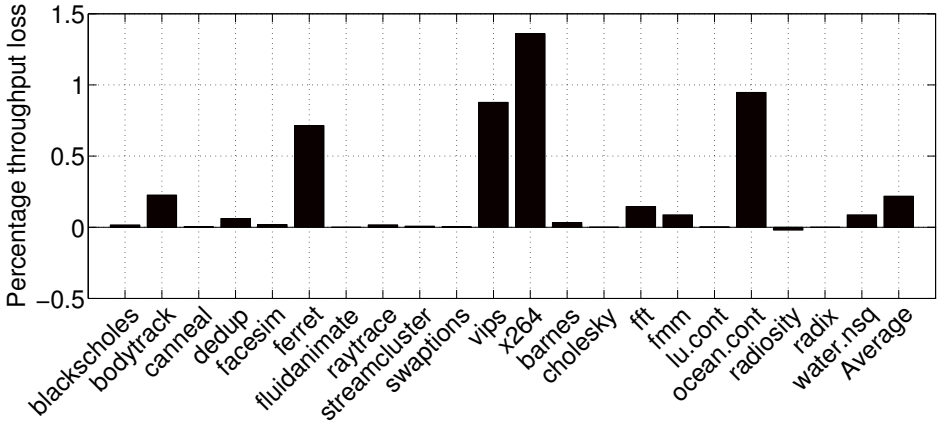

Figure 4.9: Relative throughput loss of TEI-Turbo compared with TEI-aware Steepest Drop.

It it easy to notice a slightly abnormal case in Figure 4.9, the benchmark *radiosity*, which has slightly better results for TEI-Turbo than for TEI-aware Steepest Drop. Since Steepest Drop is also a heuristic, its results cannot guarantee optimality. Sometimes, TEI-Turbo beats TEI-aware Steepest Drop, like in the case of benchmark *radiosity*. With further analysis, we found that TEI-Turbo performs better in the benchmarks with balanced or slightly imbalanced workloads between different threads. In these scenarios, TEI-Turbo can quickly narrow down the search space with the help of sweet spots to determine the find decision on V/F levels. Sometimes, TEI-Turbo could be better than TEI-aware Steepest Drop since TEI-Turbo use more sweet spots, such as for the benchmark *radiosity*. If the benchmark is heavily imbalanced, TEI-Turbo ends up being disadvantaged in narrowing down the search space with sweet spots. Therefore, TEI-Turbo may lose slightly more throughput compared with the Baseline. However, the throughput loss is still very small: less than 1.5% even in the worst case for benchmark *x264*.

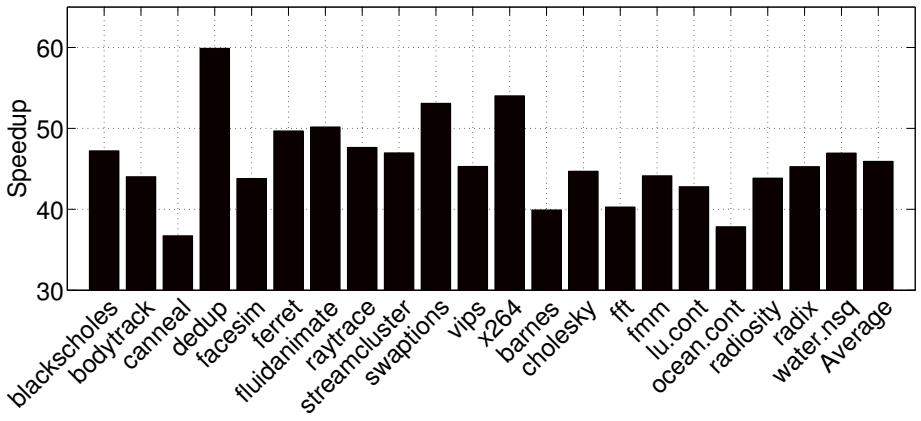

Figure 4.10: Relative runtime speedup of TEI-Turbo over TEI-aware Steepest Drop.

Second, we evaluate the runtime for each algorithm in each control epoch. We implement both TEI-Turbo and TEI-aware Steepest Drop in Python to perform a fair comparison. Figure 4.10 illustrates the relative speepdup ($\frac{1}{Runtime}$) for TEI-Turbo compared with the Baseline (TEI-aware Steepest Drop) for different benchmarks. The average speedup is **45.9**×, which sets a big advantage for TEI-Turbo.

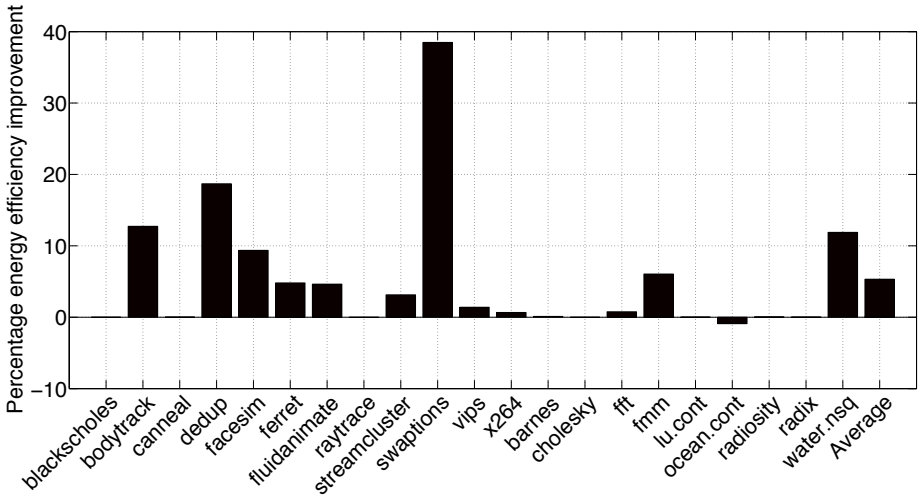

Figure 4.11: Energy efficiency improvement of TEI-Turbo over TEI-aware Steepest Drop.

In addition to that, TEI-Turbo shows better energy efficiency than TEI-aware Steepest Drop. As shown in Equation 4.5, energy efficiency in this chapter is defined as: $\frac{Throughput}{Power}$, namely the throughput achieved per unit power consumption. Although energy efficiency is not the primary target of TEI-Turbo, experimental results shows that TEI-Turbo outperforms TEI-aware Steepest Drop from this perspective. Figure 4.11 shows TEI-Turbo achieves **5.32%** more energy efficiency on average than TEI-aware Steepest Drop does, with a best case of up to 40% for benchmark *swaptions*.

A further investigation of energy efficiency for the two algorithms indicates an advantage of TEI-Turbo over TEI-aware Steepest Drop when dealing with ultra-low utilization cores. TEI-Turbo sets aside those ultra-low utilization cores into the lowest V/F level, while Steepest Drop does not. Therefore, in some phase of a multi-threaded program, some cores are in ultra-low utilization states. TEI-Turbo sets the corresponding cores directly into lowest V/F level. However, Steepest Drop can't guarantee to reach this better choice. It starts from the highest level, downgrades V/F levels one by one until power budget is satisfied. In many cases, these ultra-low utilization cores are still in some intermediate states, which consume more power without gaining more performance. Benchmark *swaptions* is one representative for this scenario.

TEI-LP algorithm

We evaluate our proposed algorithm TEI-LP in this section. Low power mode is the version of our proposed algorithm (-LP version) that minimizes power while preserving the throughput. For comparison, we use the modified version of TEI-aware Steepest Rise as mentioned in Section 4.4.2 for comparison. Same as in Section 4.4.2, we denote this method as TEI-aware Steepest Rise.

In low-power mode, we first compare the energy consumption each algorithm achieves. We run different benchmarks with default setup under steady state operation. Figure 4.12 shows the relative energy consumption for TEI-LP compared with the Baseline: TEI-aware Steepest Rise. On average, the energy consumption of TEI-LP is **0.68%** more than the Baseline.

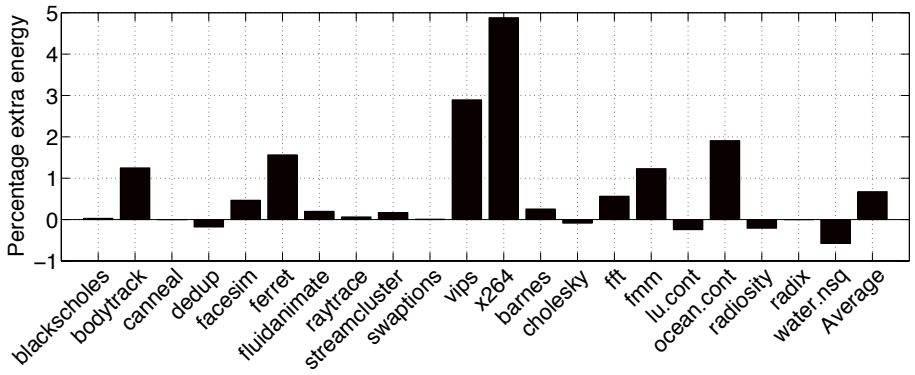

Figure 4.12: Relative energy consumption for TEI-LP compared with TEI-aware Steepest Rise.

From Figure 4.12, we can see that TEI-LP performs better than the Baseline in some benchmarks, like *water.nsq*, etc. Since TEI-aware Steepest Rise is also a heuristic, its results cannot guarantee optimality, which we have shown in Section 4.4.3. With further analysis, we found that TEI-LP performs better in the benchmarks with balanced or slightly imbalanced workloads between different threads, similar to TEI-Turbo. In these scenarios, TEI-LP can quickly narrow down the search space with the help of sweet spots to determine the find decision on V/F levels. Sometimes, TEI-LP ends up being better than TEI-aware Steepest Rise since TEI-LP use more sweet spots, such as in the case of benchmark *water.nsq*. If the

benchmark is heavily imbalanced, TEI-LP ends up being disadvantaged in narrowing down the search space with sweet spots. Therefore, TEI-LP may end up with a configuration slightly less energy efficient compared with TEI-aware Steepest Rise. However, the energy overhead is still very small: less than 5% even in the worst case for benchmark *x264*.

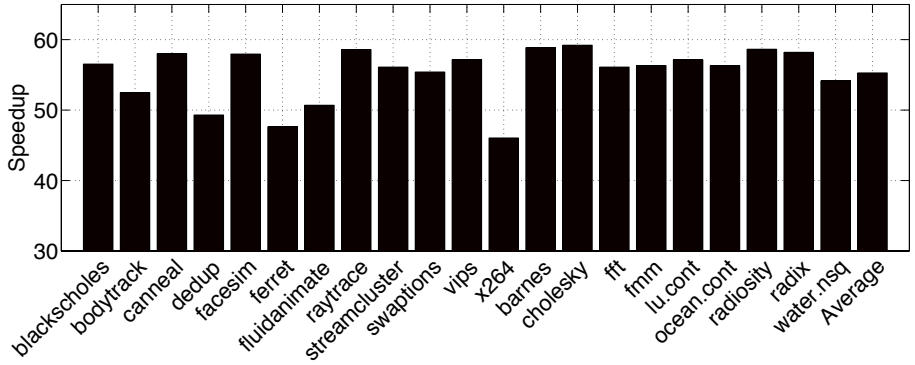

Figure 4.13: Relative runtime speedup of TEI-LP over TEI-aware Steepest Rise.

Second, we evaluate the runtime for each algorithm in each control epoch. We implement both TEI-LP and TEI-aware Steepest Rise in Python to perform a fair comparison. Figure 4.13 illustrates the relative speepdup for TEI-LP compared with the Baseline for various benchmarks. The average speedup is **55.3**×, which sets a big advantage for TEI-LP in online implementation.

In terms of energy efficiency, however, TEI-LP generally shows slightly worse energy efficiency than TEI-aware Steepest Rise. Figure 4.14 shows TEI-LP achieves **0.67%** less energy efficiency on average than TEI-aware Steepest Rise does, with a worst case of up to 5% for benchmark *x264*. Comparing Figure 4.11 and Figure 4.14, we can see that TEI-LP does not have the same behavior as TEI-Turbo in energy efficiency. TEI-Turbo makes better decisions on ultra-low utilization cores than the Baseline. Since ultra-low utilization cores have nearly no contribution to the total throughput, TEI-LP will not see a benefit in performance like TEI-Turbo does when it uses the power slack created by low V/F level operation.

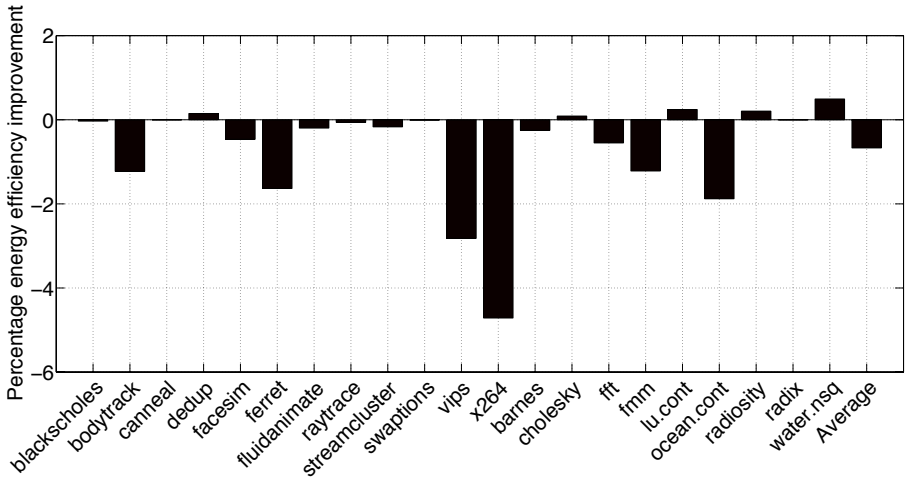

Figure 4.14: Energy efficiency improvement of TEI-LP over TEI-aware Steepest Rise.

4.4.4 Scalability of TEI-Turbo algorithm

Scalability or the ability to use an increased number of cores is one of the most important characteristics for modern CMPs. In this section, we discuss two orthogonal dimensions to characterize the scalability for TEI-Turbo. One dimension is the number of V/F levels, while the other is the number of cores in a single CMP. In this section, we continue to use TEI-aware Steepest Drop as the Baseline to gauge the efficiency of TEI-Turbo.

Number of V/F levels scalability

We first vary the number of frequency levels to see the performance of TEI-Turbo compared with the TEI-aware Steepest Drop. For fair comparison, we fixed the number of cores at the default value 16, and test the results over all benchmarks used in Section 4.4.3. Figure 4.15(a) to (c) show the average results of TEI-Turbo against the Baseline on all benchmarks.

From Figure 4.15(a), we can see that the average throughput improvement of TEI-Turbo for any number of V/F levels listed is less than 1% away from the optimal of TEI-aware Steepest Drop. We also notice from Figure 4.15(a) that for the case of five V/F levels, TEI-Turbo outperforms TEI-aware Steepest Drop in throughput improvement, similar to the case of benchmark *swaptions* discussed in Section 4.4.3. The reason is also the same: Steepest Drop and TEI-Turbo are both heuristics. Based on the results shown in Figure 4.15(b),

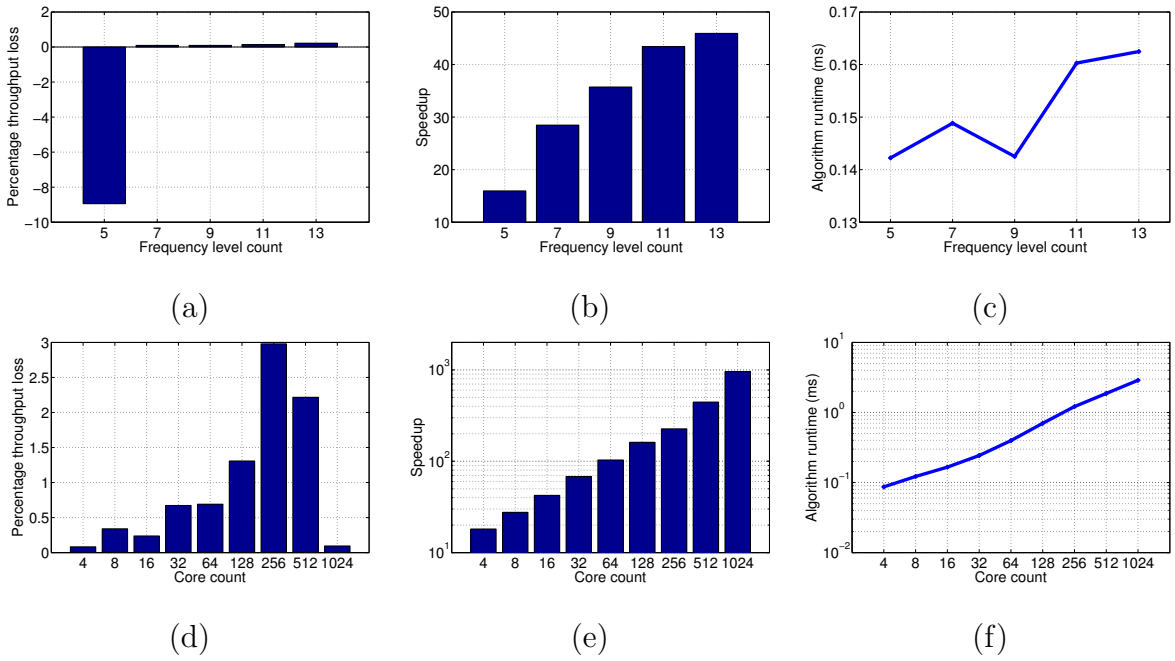

Figure 4.15: Scalability of TEI-Turbo compared with TEI-aware steepest drop: (a) Throughput loss with respect to frequency level count; (b) Speedup with respect to frequency level count; (c) Runtime of TEI-Turbo with respect to frequency level count; (d) Throughput loss with respect to core count; (e) Speedup with respect to core count; (f) Runtime of TEI-Turbo with respect to core count.

we can conclude that the speedup of TEI-Turbo over the Baseline increases linearly when the number of V/F levels increases. Considering the analysis of complexity in Section 4.3.4, TEI-Turbo has runtime complexity of $O(nlog(n))$, while the Baseline has $O(mnlog(n))$, where m is the number of frequency levels and n is number of cores. When n is fixed, the speedup of TEI-Turbo against the baseline increase linearly with m, which is confirmed by the experimental results. Figure 4.15(c) shows the runtime of TEI-Turbo averaged across all benchmarks. It shows a good scalability of TEI-Turbo with respect to the number of V/F levels.

Core count scalability

Due to the increasing trend in core count, it is important to evaluate the scalability of TEI-Turbo. This time, we keep the number of V/F levels fixed at the default value 13. We vary the core count from four all the way up to 1024. Similar to Section 4.4.4, we compare the

average results over all benchmarks, which are shown in Figure 4.15 from (d) to (f). Since a large number of cores (like 1024) is not realistic to simulate directly on HotSpot. To test the scalability of our algorithm, we combined multiple 16-core floor plans together to get the power and thermal information for core counts over 16. For a given number of cores/threads, not all benchmarks can run that number of threads together. All benchmarks can have a thread level parallelism of eight to 64 threads. Of the total, four benchmarks can run on a quad-core, eleven benchmarks can run on a 128-core system. Only two benchmarks can run on a 256- to 1024-core system. Therefore, if any single benchmark cannot be run on a given configuration 128-core and over, we run multiple copies of the same benchmark to collect data. By doing so, we can collect data for all benchmarks running on configurations with core counts varying from eight to 1024.

Figure 4.15(d) demonstrates that TEI-Turbo's best throughput is up to an average of 3% less than the Baseline. On the other hand, as shown in Figure 4.15(e), the speedup of TEI-Turbo against the Baseline increases fast when core count increases. Even in the case of the four-core setting, TEI-Turbo still achieves an average of 18.1× speedup against the Baseline. For a 1024-core CMP, this speed up increase to 957×. The average runtimes for all core counts shown in Figure 4.15(f) demonstrate a good scalability of TEI-Turbo with respect to core count: only an average of 33.3× in runtime slowdown when core count increases from four to 1024. The value of 33.3 is different than the result obtained for the worst case analysis because it is calculated based on the average runtime, which is different than the worst-case.

4.4.5 Scalability of TEI-LP algorithm

In this section, we discuss two orthogonal dimensions to characterize the scalability for TEI-LP, similar to Section 4.4.4. One dimension is the number of V/F levels, while the other is the number of cores in a CMP. In this section, we used the TEI-aware Steepest Rise as the Baseline to gauge the efficiency of TEI-LP.

Number of V/F levels scalability

We first vary the number of frequency levels to evaluate how TEI-LP performs compared with the TEI-aware Steepest Rise. For fair comparison, we fixed the number of cores at the default value 16, and test the results over all benchmarks used in Section 4.4.3. Figure 4.16(a) to (c) show the average results of TEI-LP against the Baseline on all benchmarks.

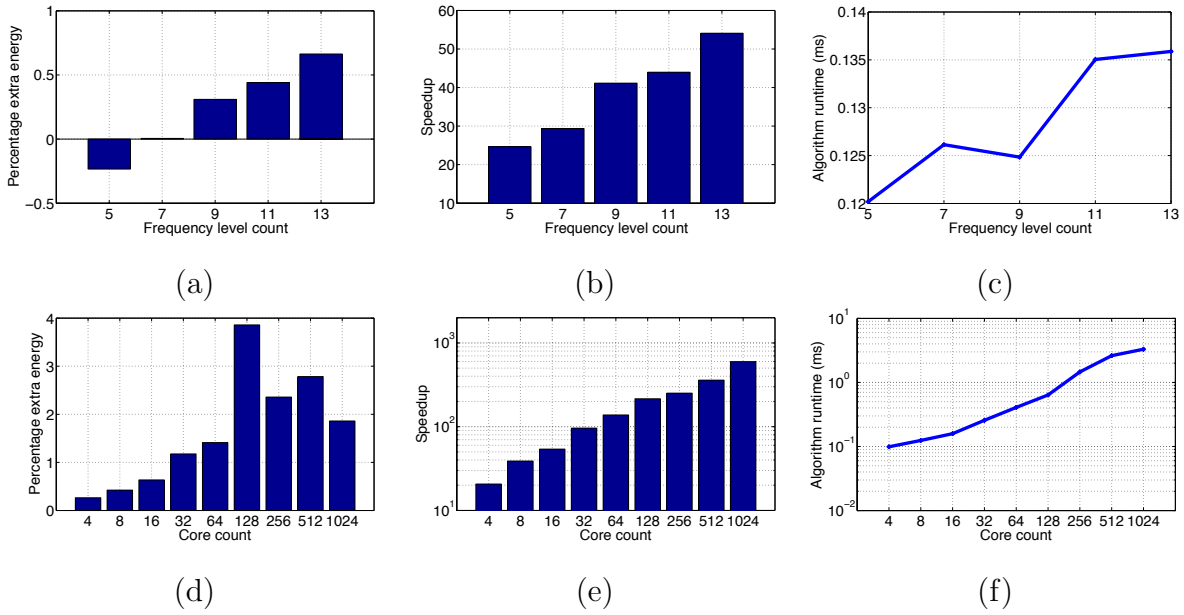

Figure 4.16: Scalability of TEI-LP compared with TEI-aware steepest rise: (a) Extra energy consumption with respect to frequency level count; (b) Speedup with respect to frequency level count; (c) Runtime of TEI-LP with respect to frequency level count; (d) Extra energy consumption with respect to core count; (e) Speedup with respect to core count; (f) Runtime of TEI-LP with respect to core count.

From Figure 4.16(a), we can see that the average energy savings of TEI-LP for any number of V/F levels listed is less than 1% away from the level achieved by TEI-aware Steepest Rise. We also notice from Figure 4.16(a) that for the case of five V/F levels, TEI-LP outperforms TEI-aware Steepest Rise. The reason is also the same: Steepest Drop and TEI-LP are both heuristics, which cannot guarantee the optimality. Based on Figure 4.16(b), we can conclude that the speedup of TEI-LP over the Baseline increases linearly when the number of V/F levels increases, which agrees with the results of complexity analysis in

Section 4.3.4. Figure 4.16(c) shows the average runtime of TEI-LP for all benchmarks. TEI-LP shows a good scalability with respect to the number of V/F levels, similar to TEI-Turbo.

Core count scalability

In this section, we keep the number of V/F levels fixed at the default value 13. We vary the core count from four all the way up to 1024. We compare the average results over all benchmarks, which are shown in Figure 4.16 from (d) to (f). The average values are based on the data collected as mentioned in Section 4.4.4.

Figure 4.16(d) demonstrates that TEI-LP's energy consumption is up to an average of 4% more than the Baseline. On the other hand, as shown in Figure 4.16(e), the speedup of TEI-LP against the Baseline increases fast with core count. Even in the case of the four-core setting, TEI-LP still achieves an average of 20.7× speedup against the Baseline. For a 1024-core CMP, this speed up increase to 598×. The average runtimes for all core counts shown in Figure 4.16(f) demonstrate a good scalability of TEI-LP with respect to core count: only an average of 33.4× in runtime slowdown when core count increases from four to 1024.

4.5 Discussion

In this chapter, we discuss the impact of temperature effect inversion (TEI) on FinFET-based multi-core systems. We construct accurate system-level power and performance models and apply them to evaluate the impact of TEI. We discover the TEI-induced sweet spots, which are locally better V/F level choices. Based on these sweet spots, we propose our fast algorithms, including TEI-Turbo determining the maximum throughput with a given power budget and TEI-LP determining the minimum energy consumption for a given throughput constraint. Experimental results shows that in a 16-core CMP setup, our algorithms achieve an average speedup of 45.9× with losing only 0.22% in throughput (TEI-Turbo), or an average speedup of 55.3× with only 0.68% more energy consumption (TEI-LP), compared

with TEI-aware versions of existing state-of-the-art algorithms. Results also show excellent scalability of our algorithms in both core count and number of V/F levels.

Chapter 5

Aging-reduction in multi-core systems

5.1 Chapter overview

Power and thermal issues have become the major constraints in designing high-performance microprocessors. Several hardware- and software-based techniques have been proposed to increase performance while keeping power under a given Thermal Design Power (TDP) budget. Nevertheless, as CMOS technology aggressively scales down to deca-nanometer technology nodes, these issues have also emerged as important reliability threats throughout the system lifetime. Aging concerns have therefore gathered a significant momentum and they have already triggered extensive research on aging modeling and mitigation techniques for older planar MOSFET-based system design.

FinFET has been widely chosen as the next generation CMOS technology. Prior art has already explored TEI as means of operating FinFET-based processors at higher temperature under iso-power or iso-frequency operation [9]. However, these works do not account for the correlation between the TEI-induced operating regimes and the aging mechanisms of FinFETs. On the other hand, existing system-level aging-aware methodologies consider the evolution of aging under the assumption of constant temperature, frequency and V_{dd} conditions throughout the entire lifetime [82] [25]. However, typical modern FinFET-based processors are characterized by dynamically changing operating V/F levels, temperatures,

or performance characteristics; a typical example is the 15 V/F levels on Xeon E5-2670 processor [50].

Therefore, the *key motivation* of this chapter is to quantify the interplay between TEI and aging effects in FinFET-based multi-core systems under multiple voltage/frequency levels, and to select the level that would optimally decrease aging under power/ performance constraints. To the best of our knowledge, we are *first* to provide a comprehensive model of both TEI- and aging-aware power and performance characteristics of a FinFET-based core in the context of large scale multi-core systems. Based on the **key insight** that V_{dd} is the dominant factor for both power and aging issues, our experimental results show that by considering a combined multivariate objective for power and aging while exploiting the TEI effect, FinFET-based systems can inherently trigger aging reduction. To this end, we propose *AgingMin*, an algorithm that selects the optimal V/F operating points throughout the system lifetime, while accounting for time-dependent evolution of both TEI and aging mechanisms.

5.1.1 Chapter contributions

Compared to existing works, our work makes the following *novel contributions*:

- This is the first work that simultaneously characterizes the impact of both TEI and aging effects on power and performance of multi-core systems. To this end, we propose a multivariate polynomial model to learn the core frequency and power under TEI and aging effects. Our learning framework achieves fitting with error always less than 4% when compared to detailed HSPICE and core-level simulation.

- We exploit TEI to effectively select the V/F operational points throughout the system lifetime under performance constraints. Our experimental results show that aging effects can be reduced by up to 53.59% by exploiting the TEI effect when compared to a TEI-agnostic approach.

- We provide a system-wide investigation of the *voltage acceleration* mechanism previously reported from raw wafer-level measurements [52], which shows that the supply voltage V_{dd} plays a key role in the aging of the multi-core systems. Based on this *key insight*, we propose an aging-aware algorithm called *AgingMin*, to determine the optimal V/F operating regions throughout the system lifetime under given performance constraints.

- We evaluate our proposed algorithm on multi-core simulator [13] with various multi-threaded benchmarks [3] [89]. Experimental results show *AgingMin* improves the 10-year system lifetime by an average of 1.61 years while introducing less than 1% power overhead when compared to state-of-the-art techniques.

5.2 Performance and power models

5.2.1 Introducing TEI- and aging-awareness

To enable a representative evaluation of FinFET-based many-core systems, it is important to incorporate TEI- and aging-awareness within our proposed performance and power models. To this end, let us first reexamine the effect of TEI and aging on the nominal frequency of a RO, as presented in Figure 2.2. We observe that both phenomena exhibit significant temperature and V_{dd} dependence. Moreover, for the same temperature and V_{dd} conditions, we observe the frequency degradation increases from three to ten years of lifetime compared to the nominal value[1].

Hence, the performance and power characteristics of a FinFET-based core depend on the V_{dd}, temperature, and the threshold voltage degradation $\Delta V_{th}(t)$ throughout the lifetime, which in turn is a function of temperature θ, V_{dd}, time t and duty factor df. To quantitatively characterize the overall TEI- and aging-aware frequency $\delta_{\mathfrak{F}}$ and static power δ_{Psta} shift, we

[1]These observations are consistent with the results and the terminology used by several research groups: *temperature effect inversion* [9], *temperature-* and *time-critical* [25], *voltage-* and *time- acceleration* [52].

use multivariate polynomial regression models:

$$\delta_{\mathfrak{F}}(t) = \sum_i c_i \cdot V_{dd}^p \cdot \theta^q \cdot \Delta V_{th,\text{pmos}}^r(t) \cdot \Delta V_{th,\text{nmos}}^s(t) \tag{5.1}$$

$$\delta_{P_{sta}}(t) = \prod_i \exp(c_i^{'} \cdot V_{dd}^{p'} \cdot \theta^{q'} \cdot \Delta V_{th,\text{pmos}}^{r'}(t) \cdot \Delta V_{th,\text{nmos}}^{s'}(t) \tag{5.2}$$

After model selection experimentation, we finally choose regression models with a total degree of up to four which provides best accuracy, *i.e.*, $p+q+r+s \leq 4$ and $p'+q'+r'+s' \leq 4$. Note that we use an exponential multivariate polynomial for static power, to account for the exponential dependency with respect to the threshold voltages. While not explicitly stated here, we note that dynamic power is implicitly affected by TEI- and aging-aware effects through the operating frequency.

As shown before, a ring oscillator (RO) provides a sufficiently accurate approximation for the core frequency [33] [9]. Hence, to learn $\delta_{P_{sta}}$ and $\delta_{\mathfrak{F}}$, we perform curve fitting via HSPICE simulations of Ring Oscillators (RO) in a 16 nm predictive technology model (PTM). We form a 23-stage RO to characterize the delay and total power, and a two-stage inverter chain to characterize the static power. To ensure an extensive analysis and accurate fitting of our model, we sweep across a wide range of θ, V_{dd} and $\Delta V_{th}(t)$ values to capture their effects on the performance and power. Given over $20,000$ Monte-Carlo HSPICE data points, we fit the expressions for $\delta_{P_{sta}}$ and $\delta_{\mathfrak{F}}$.

Figure 5.1 shows the scatter plot of $\delta_{P_{sta}}$ values predicted with Equation 5.2 versus the real frequency degradation reported by HSPICE, using a testing set of $2,000$ data points. Our proposed multivariate multinomial model achieves a negligible Mean-Squared-Prediction-Error (MSPE) of 3.11% across $2,000$ testing data. Similarly, with Equation 5.1 for predicting $\delta_{\mathfrak{F}}$ our model achieves an MSPE of 3.44% .

5.2.2 Power and performance metrics

As a representative metric to capture the TEI- and aging-induced performance degradation, we use the total throughput TP of CMP with n cores, *i.e.*, the total number of instructions

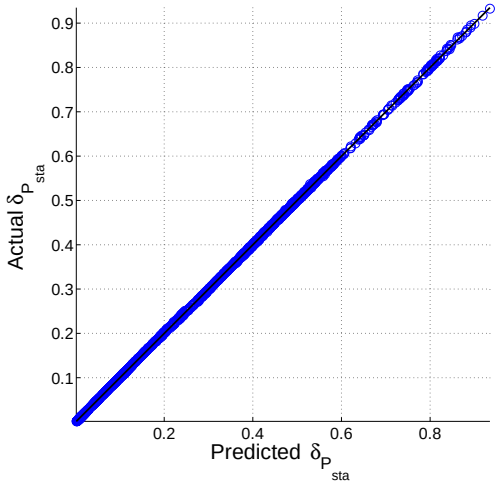

Figure 5.1: Prediction accuracy of $\delta_{P_{sta}}$ (Equation 5.2): Ideally, all points should lie along the diagonal.

committed per unit time for all cores. Given the average number of instructions committed per cycle (IPC), and the operating frequency ($\mathfrak{F}$), we have:

$$TP = \sum_{j=1}^{n} IPC_j \cdot \mathfrak{F}_j^{\text{nom}} \cdot (1 + \delta_{\mathfrak{F}_j}(t)) \tag{5.3}$$

It is worth noting that Equation 5.3 inherently captures the lifetime performance degradation due to thermal and aging implications at any point t of the core's lifetime.

Another important aspect of the core functionality is the impact of the threshold voltage, V_{dd} and temperature on the leakage characteristics of the core, mainly caused by subthreshold leakage. Following the same rationale behind the exponential dependency with respect to temperature θ, we use a simple system-level model to characterize static power of core j under thermal and aging effects:

$$P_{sta_j} = c \cdot V_{dd_j} \cdot \exp(\kappa_1 \cdot \theta_j^2 + \kappa_2 \cdot \theta_j + \kappa_3) \cdot (1 + \delta_{P_{sta_j}}(t)) \tag{5.4}$$

where c is the normalization factor and κ_1, κ_2 and κ_3 are the coefficients to determine. Since $\log(P_{sta_j})$ behaves as a sublinear function of θ, we choose a quadratic function in the exponent. Unlike existing modeling approaches for the same problem formulation [9], please note again that our model captures the TEI- and aging-induced power degradation at any point t of the core's lifetime.

To validate this model, we collect leakage power data from the widely used system-level power simulator McPAT [58] and we use HotSpot [38] to obtain the stable temperature. We fit the model across a wide temperature range (from -25°C to 125°C) and voltage range (from $0.45V$ to $1.05V$), achieving Root-Mean-Square-Percentage-Error (RMSPE) less than 2%.

Finally, prior work [4] has shown that dynamic power has strong correlation with IPC, since IPC approximately represents the activity rate (or utilization) of a processing core. Therefore, similar to [4], we capture dynamic power for each core j as:

$$P_{dyn_j} = V_{dd_j}^2 \cdot \mathfrak{F}_j^{\text{nom}} \cdot (1 + \delta_{\mathfrak{F}_j}(t))(a \cdot IPC_j + b) \tag{5.5}$$

where a and b are coefficients that need to be fitted. To this end, we use a multi-core system-level performance simulator, namely Sniper [13], and McPAT [58] to run multi-threaded benchmarks from PARSEC [3] and SPLASH-2 [89] on FinFET-based chip-multiprocessors (CMPs) for varying V/F levels. Across all the 22 benchmarks considered, we achieve an RMSPE of 6.47% on average.

5.3 Multi-objective optimization

In this Section, we discuss the interdependencies between thermal, aging and power considerations in FinFET-based multi-core systems. We then propose our aging-aware V/F selection algorithm, dubbed *AgingMin*, to decrease the aging process under given performance constraints.

5.3.1 Thermal, aging and power considerations

First, let us discuss the complexity of the problem that we effectively tackle in the remainder of the chapter. The goal is the implementation of an algorithm to perform V/F level selection so as to minimize aging effects while maintaining the target performance throughout the system lifetime. For the traditional multi-core systems with planar CMOS, this task is less

complicated thanks to the apparent coupling of temperature and leakage power: increased temperature increases leakage power which in turn increases temperature further, forming the leakage- temperature loop [67].

For FinFET-based multi-core system though, these coupling effects are complex: increased temperature leads to TEI-induced effects on the V/F pairings, which directly result in changes in the power characteristics of the core. Furthermore, leakage power is not only affected by temperature, but also by voltage. The changes in both dynamic power and leakage power will then change the temperature further. Nonetheless, higher temperature could be exploited for supply voltage reduction (*i.e.*, TEI-induced reduction in the power consumption), which finally decreases the temperature. Thus, the feedback loop used for identifying the optimal operating conditions for planar-CMOS-based systems [67], is no longer representative of thermal and power considerations in FinFET-based systems. This more complicated coupling effect with counteracting interdependencies has been previously studied and motivated by different groups [57] [9] [79].

In terms of lifetime considerations, wafer-level measurements in sub-20nm FinFET technology nodes confirm that the aging mechanisms are highly associated with the operating supply voltage and temperature, capturing the so-called *voltage acceleration* [52]. Hence, aging and power optimization problems cannot be decoupled. In fact, power optimization under performance constraints will result in near optimal working temperature. However, for decreasing the aging process, online approaches may deviate from reaching this optimal power consumption. Then, the working temperature will increase, which may finally increase the aging process. Therefore, decreasing the aging process is likely not equivalent to optimizing for power consumption as there may be competing criteria for both cases. Our work is the first to comprehensively take thermal effects, aging process and power consumption into consideration.

5.3.2 Proposed approach: *AgingMin*

As aforementioned in the previous Section, power optimization under performance constraints will result in significant benefits for thermal and aging reduction. However, the resulting power-optimal operating points are unlikely to optimize aging reduction. By carefully studying the power and aging behaviors, we propose an aging-aware V/F selection algorithm for FinFET-based multi-core systems, dubbed *AgingMin*. We first define a comprehensive score function as below:

$$S = \sum_{j=1}^{n} (V_{dd_j} - \lambda)^{\rho} \tag{5.6}$$

The score function is based on the consideration of both aging and power consumption together. The common thing between power and aging is that they are highly dependent on supply voltage. Indeed, performance degradation due to aging depends superlinearly on supply voltage. Furthermore, total power depends on supply voltage superlinearly for the dynamic portion and linearly for the static portion (Equations 5.4-5.5), while also being a function of clock speed and static power variability, both of which superlinearly dependent on supply voltage. To account for these effects in a single score function that can capture close to optimal operations from both a power and aging standpoint, we have experimented with various polynomial functions. The score function (Equation 5.6) effectively captures this dependency, thus being an ideal candidate. Based on extensive tries, the empirical values $\lambda = V_{th0}$ and $\rho = 3$ work well in decreasing the aging process when using an exhaustive search method to mitigate aging.

Given this score function, we propose an aging-aware V/F selection algorithm, dubbed *AgingMin*, as shown in Algorithm 3. Specifically, given the machine state (IPC), the performance requirement T_{min}, the temperature map θ and core count N, *AgingMin* finds the best V/F choices to decrease aging. For initialization, *AgingMin* calculates the V/F pairs for current temperature map and score functions defined in Equation 5.6 for each state in each core (Line 5-6). Then it initializes all cores with the lowest states and prepares the *Candidates* heap for potential state upgrade, ranked by *priority* values (Line 7-12). Then,

the algorithm moves on to achieve the performance requirement T_{min} through a loop (Line 13-24). For each iteration, *AgingMin* selects one core with the smallest *priority* value to upgrade one V/F level. The iteration continues until T_{min} is satisfied or *Candidates* is empty.

Let us consider the complexity of the *AgingMin*. Our proposed algorithm is composed of two phases, the initialization phase and iteration phase. We assume that total core state count is M. During the initialization part, the algorithm basically iterates over all states for all cores to record the necessary information, with a complexity of $O(MN)$. During the iteration part, in the worst case, the algorithm iterates over all states for all cores in the internal loop for up to two heap pops. This portion therefore has a complexity of $O(MNlog(N))$. In summary, *AgingMin* has a complexity of $O(MNlog(N))$.

Algorithm 3 *AgingMin*

1: Input: IPC, TP_{min}, N, θ
2: Output: VF levels for all cores
3: *Candidates*: a min-heap for potential VF level upgrade ranked by *priority*
4: //initialization
5: Calculate all the matching VF_i for each Core i
6: Calculate the Score values S_{ij} for Core i in State j
7: **for** $i = 0$; $i < N$; $i++$ **do**
8: $VF_i \leftarrow lowest$
9: $TP \leftarrow TP + TP_{i,lowest}$
10: $priority_i \leftarrow \Delta S_i / \Delta TP_i$ if Core i upgrade one VF level
11: Push $(i, priority_i)$ to the heap *Candidates*
12: **end for**
13: **while** (1) **do**
14: **if** $TP \geq TP_{min}$ **then**
15: **return**
16: **else if** *Candidates* is empty **then**
17: **return** INFEASIBLE
18: **else**
19: pop $(i, priority_i)$ from *Candidates*
20: $VF_i \leftarrow VF_i + 1$
21: $TP \leftarrow TP + \Delta TP_i$
22: push new $(i, priority_i)$ to *Candidates* if higher VF state is available
23: **end if**
24: **end while**

5.3.3 Accounting for thermal effects

As we stressed already, it is essential for our proposed model to account for the complex thermal interdependencies between aging, TEI, performance, and power. That is, *AgingMin* needs to determine the steady state operating temperature: since the V/F matching is based on the temperature, *AgingMin* needs to obtain the temperature information before running the algorithm. However, after the algorithm makes the final choice, there will be updated steady temperature, which could be different than the values used for the process of V/F matching. That would yield a matching that is not valid in the current temperature.

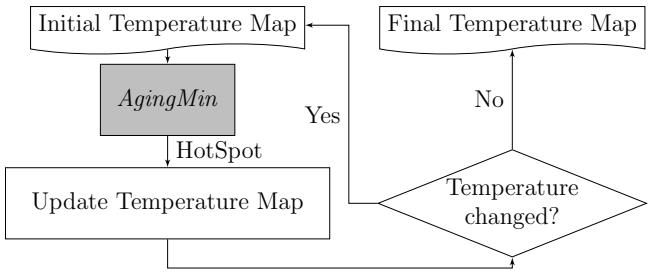

Figure 5.2: Proposed flow for determining the steady state operating temperature.

To effectively solve this problem, we propose an iterative approach as shown in Figure 5.2: we start with the temperature from the previous iteration of the algorithm to calculate the V/F pairing. Then we use *AgingMin* to find the best V/F pairs. The updated power map will be used to calculate the steady state operating temperature and use the updated temperature map to repeat the whole process. We execute the loop until the temperature map converges. In practice, the process converges very fast, with the average number of iterations being three or four.

5.4 Experimental results

In this section, we first present the experimental setup. We then evaluate the impact of TEI on decreasing the aging process. Finally, we comprehensively evaluate our proposed aging-aware algorithm against the best-to-date baseline implementation.

5.4.1 Experimental setup

Chip aging is a long-term phenomenon, as encapsulated by the power law assumption [52]. To this end, we use the notion of the "aging cycle", as it is already defined by long-term aging analyses from several groups [82] [25]: the data from finer-grained simulations are upscaled to the time range of the aging cycles, with granularity that can be flexibly set to varying duration from 1 month to years. This allows for the time-dependent evaluation of the aging mechanisms.

To collect the finer-grained data, we use the Sniper [13] multi-core simulator to profile the performance for typical applications. The default architecture used in this chapter is listed in Table 5.1. We adopt nine V/F levels with the default matching in Table 5.2. To enable a detailed exploration, the aging cycle is set as 1 month. Moreover, we modify the leakage power model in system-level power simulator McPAT [58] to incorporate the temperature- and aging-induced leakage power variations. Finally, we use HotSpot [38] to evaluate the thermal-related effects and to obtain the stable temperature. We adopt a typical 16-core floorplan for evaluating our architecture [9].

Table 5.1: Target architecture.

Parameters	**Values**
Number of cores	16
Number of core states	9
Nominal frequency	3.0 GHz
Core model	Intel Nehalem-like
L2 caches	Private 256KB, 8-way SA, LRU
L3 caches	Shared 32MB, 16-way SA, LRU
DRAM	8GB
Technology	16nm FinFET

Table 5.2: V/F level matching for θ =-25°C.

Level	0	1	2	3	4	5	6	7	8
$\mathfrak{F}$	1.05	1.40	1.70	2.00	2.25	2.45	2.65	2.85	3.00
V_{dd}	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85

5.4.2 TEI effect on aging

Our goal is to achieve the slowest aging while maintaining the target performance. Given the key insight that V_{dd} plays a key role in the both aging and power optimization, our problem boils down to identifying the proper V/F operating regime throughout the system lifetime. To this end, we propose *AgingMin* algorithm that employs directed local search (DLS) to identify the optimal V/F pairs under the given performance and aging constraints. It is worth noting that directed local search has been already assessed in terms of optimality and effectiveness in power or performance optimization in multi-core systems [88].

To this end, we use *power-oriented* DLS as the baseline to extensively evaluate the impact of TEI-induced effects on aging. In this case, we basically use DLS to optimize for power and gauge the effects of doing so on aging. To capture the TEI effect throughout the system lifetime, we implement two versions of DLS:

- TEI-Unaware DLS: No dynamic V/F matching that accounts for the TEI effect is considered. That is, the directed local search is performed on the nominally assumed, fixed V/F pairs.

- TEI-Aware DLS: Dynamic V/F matching is enabled to exploit the TEI effect, allowing the directed local search to converge to the best V/F selection.

This best-to-date, baseline implementation provides us with a representative comparison point against which we will later substantiate the optimality of our algorithm *AgingMin*.

We test the two versions of the DLS algorithm on various application scenarios profiled from multi-threaded benchmark suits PARSEC [3] and SPLASH-2 [89]. We further set different scales, namely low, medium, and high, in the performance requirements for further evaluations. Low performance requirement is set as the throughput achieved by all cores working at 1.5 GHz. The corresponding values for the medium and high performance are 2.2 GHz and 3.0 GHz. To illustrate the process of aging throughout the system's lifetime, we record the aging effects for a typical application scenarios for ten years (typical lifetime value)

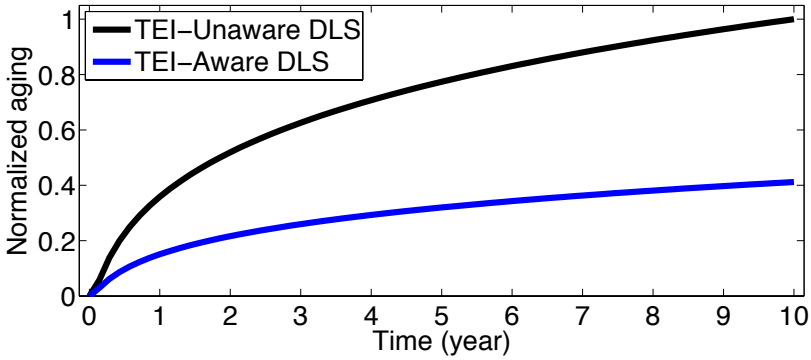

Figure 5.3: Improvement of TEI-Aware over TEI-Unaware DLS for aging reduction for a typical application.

under the two version of DLS algorithms separately for the high performance requirement, shown in Figure 5.3. From Figure 5.3, we can see that TEI effect uniformly and significantly reduces the aging rate. To quantify the impact of TEI comprehensively, we test them with various application scenarios for different performance requirement. The results are shown in Figures 5.4, 5.5, and 5.6.

Aging is evaluated as mean and variance values for ΔV_{th} over all cores. Figure 5.4 shows that TEI brings significant improvement in aging reduction under various applications scenarios with different performance requirements. More specifically, TEI enables an average of 27.73%, 36.46%, and 53.59% reduction in aging effects (as quantified by ΔV_{th}) for low, medium, and high performance constraints respectively. These values are average on 22 applications profiled from benchmarks from [3] [89], which are also applicable for the test in Section 5.4.3. As shown in Figure 5.5, TEI achieves an average of 17.52%, 19.59%, and 30.64% improvement in power savings for the different performance constraints respectively. This is not a surprise, as the objective of the DLS heuristic in this case was to optimize power. As a beneficial side effect from power savings, TEI reduces the average operating temperature by 1.91°C, 3.59°C, and 10.60°C for the different performance constraints, respectively.

Based on the general trend shown in Figures 5.4, 5.5, and 5.6, we observe that the higher the performance constraint is, the larger the aging reduction rates, power savings, and temperature reductions achieved by TEI effect. The reason behind this is that the higher

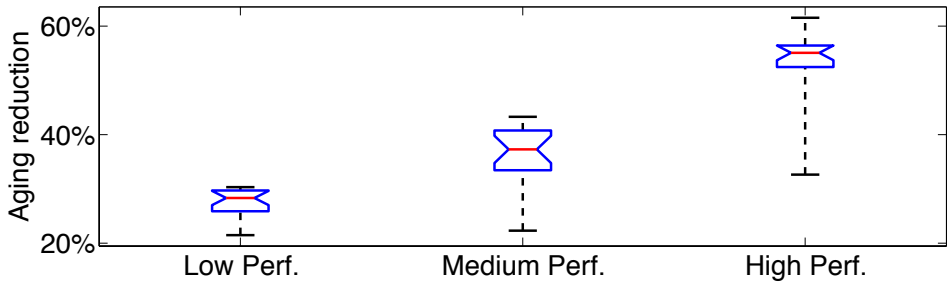

Figure 5.4: Improvement of TEI-Aware over TEI-Unaware DLS for aging reduction under various performance constraints.

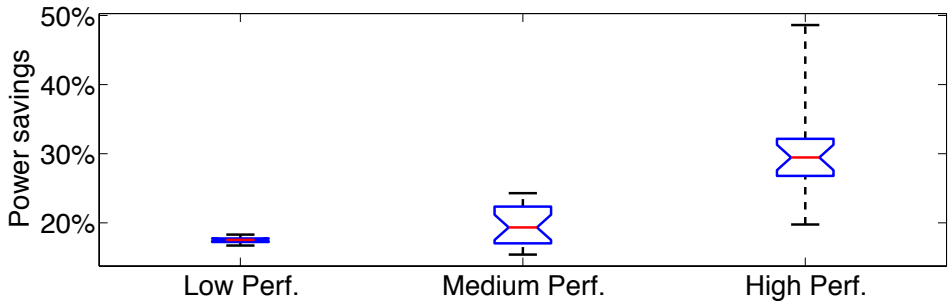

Figure 5.5: Improvement of TEI-Aware over TEI-Unaware DLS for power savings under various performance constraints.

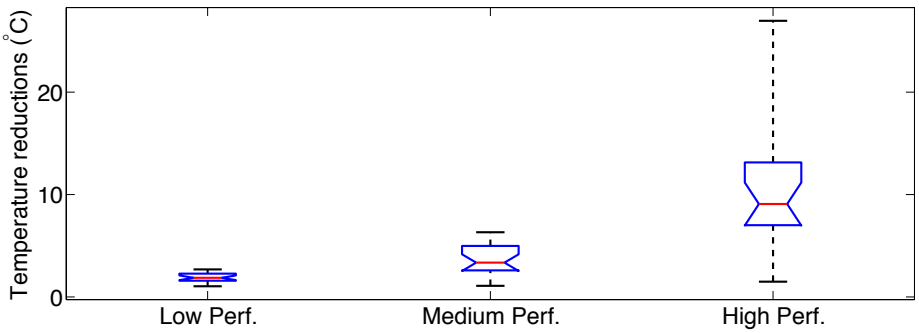

Figure 5.6: Improvement of TEI-Aware over TEI-Unaware DLS for temperature reduction under various performance constraints.

the performance needed, the larger the power consumption and temperature are. According to TEI, the higher temperature makes more room to downgrade V_{dd}, thus saving more power and achieving more temperature reduction, which confirms prior work in [57]. Therefore, aging gets further decreased from lower power and temperature.

By comparing the results of medium performance requirement to the others in Figures 5.4, 5.5, and 5.6, we can see that the aging reduction under the medium performance requirement is in the middle between low and high performance requirements while the power and energy savings under the medium performance requirement are closer to the low performance requirement.

5.4.3 Evaluation of *AgingMin*

Next, we substantiate the effectiveness of the proposed *AgingMin* algorithm in terms of the overall aging reduction achieved. To the best of our knowledge, there is no TEI-aware and aging-aware V/F selection algorithm for aging reduction while maintaining performance for FinFET-based multi-core systems. As already mentioned, we have implemented a TEI- and aging-aware DLS scheme, since it allows us to employ a representative comparison against a best-to-date baseline. In our evaluations below, we compare *AgingMin* with TEI-Aware DLS under various application scenarios and different performance requirements.

To enable long-term evaluation, we set the lifetime goal for a FinFET-based multi-core system to ten years. We then evaluate the aging reduction achieved from the both methods given this 10-years goal for the same application and performance requirements. The results are shown in Figure 5.7. More specifically, Figure 5.7 confirms the effectiveness of *AgingMin* in decreasing the aging process by showing that an average extended lifetime is 1.61, 0.70, and 0.55 years under low, medium, and high performance requirements respectively. As shown in Figure 5.8, on the contrary, the corresponding average power overhead for *AgingMin* compared with TEI-Aware DLS is only 0.19%, 0.81%, and 0.90% which are negligible. By comparing the results of medium performance requirement to the others in Figures 5.7 and 5.8, we can see that the lifetime extention and power overhead for *AgingMin* under the medium performance requirement are closer to the high performance requirement.

From the trend shown in Figures 5.7 and 5.8, we can see that it is harder to achieve additional aging benefits from using the combined score function for power and aging (defined

in Equation 5.6) for higher performance than in lower one. In addition, the lower additional benefit comes at a slightly higher power cost. However, considering the average power overhead is less that 1%, it is worthwhile to spend it for up to an average of 1.61-year extra lifetime. One thing to note here is that, *AgingMin* is compared with TEI-Aware DLS. When compared to the state-of-the-art TEI-unaware algorithm, the benefit would be an additive combination of both.

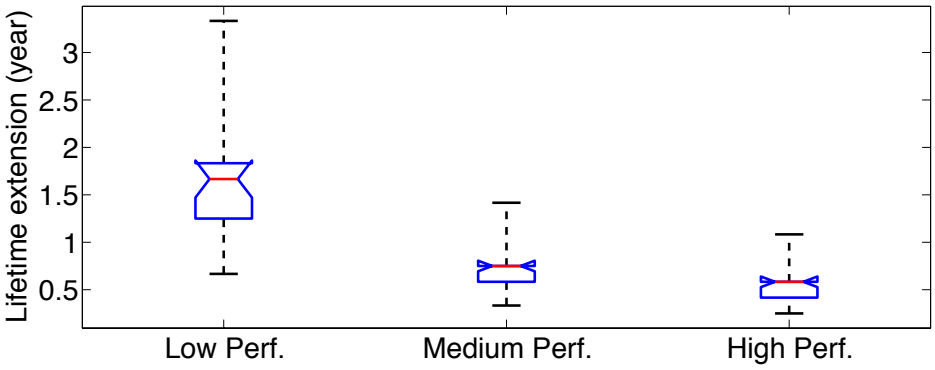

Figure 5.7: The extended lifetime of *AgingMin* against TEI-Aware DLS under different performance constraints.

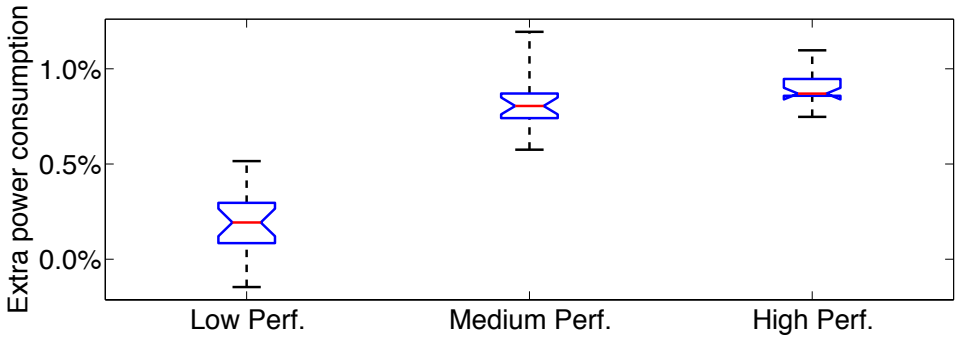

Figure 5.8: The extra power consumed from *AgingMin* against TEI-Aware DLS under different performance constraints.

5.5 Discussion

In this chapter, we are the *first* to provide a comprehensive evaluation of both TEI and aging effects on the performance and power of FinFET-based multi-core systems with multiple

voltage/frequency levels. To this end, we first propose a multivariate polynomial model to learn the TEI- and aging-aware performance and power degradation with error always less than 4% when compared to detailed HSPICE and core-level simulation. We then quantify the interplay between TEI and aging effects in FinFET-based multi-core systems under multiple voltage/frequency levels. Our experimental results show that aging effects can be reduced by up to 53.59% by exploiting the TEI effect. Moreover, we propose an aging-aware algorithm, dubbed *AgingMin*, to select the optimal TEI-aware voltage/frequency operation points for decreasing the aging effects. Experimental results show *AgingMin* improves the classic 10-year system lifetime by an average of 1.61 years while introducing less than 1% power overhead when compared to existing state-of-the-art directed local search algorithm.

Finally, it is worth noting that that *AgingMin* is an orthogonal algorithm to the existing aging reduction methods, such as thread mapping and dark silicon designs. Thus, our proposed algorithm can be flexibly incorporated within existing aging mitigation techniques to achieve further aging reduction, whose integration we leave for future work. Since multiple voltage/ frequency levels are available in many FinFET-based multi-core systems, *AgingMin* can be widely used to extend their lifetimes.

Chapter 6

Convolutional neural network: power and latency

6.1 Chapter overview

To identify energy-efficient CNNs, the use of accurate runtime, power, and energy models is of crucial importance. The reason for this is twofold. First, commonly used metrics characterizing CNN complexity (*e.g.*, total FLOPs[1]) are too crude to predict energy consumption for real platforms. Energy consumption depends not only on the CNN architecture, but also on the software/hardware platform. In addition, it is also hard to predict the corresponding runtime and power. Second, traditional profiling methods have limited effectiveness in identifying energy-efficient CNNs, due to several reasons: 1) These methods tend to be inefficient when the search space is large (*e.g.*, more than 50 architectures); 2) They fail to quantitatively capture how changes in the CNN architectures affect runtime, power, and energy. Such results are critical in many automatic neural architecture search algorithms [94]; 3) Typical CNNs are inconvenient or infeasible to profile if the service platform is different than the training platform.

[1]FLOP stands for "floating point operation".

Therefore, it is imperative to train models for power, runtime, energy consumption of CNNs. Such models would significantly help Machine Learning practitioners and developers to design accurate, fast, and energy-efficient CNNs, especially in the design space of mobile or embedded platforms, where power- and energy-related issues are further exacerbated.

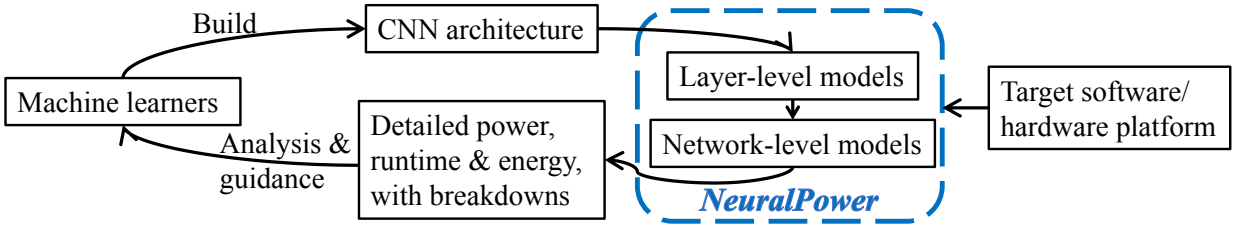

Figure 6.1: *NeuralPower* quickly predicts the power, runtime, and energy consumption of a CNN architecture during service phase. Therefore, *NeuralPower* provides the machine learners with analysis and guidance when searching for energy-efficient CNN architectures on given software/hardware platforms.

In this chapter, we develop a predictive framework for power, runtime, and energy of CNNs during the testing phase, namely *NeuralPower*, *without* actually running (or implementing) these CNNs on a target platform. The framework is shown in Figure 6.1. That is, given (a) a CNN architecture of interest and (b) the target platform where the CNN model will be deployed, *NeuralPower* can directly predict the power, runtime, and energy consumption of the network in service/deployment phase.

6.1.1 Chapter contributions

Compared to existing works, this chapter brings the following contributions:

- To the best of our knowledge, our proposed learning-based polynomial regression approach, namely *NeuralPower*, is the first framework for predicting the *power consumption* of CNNs running on GPUs, with an average accuracy of 88.34%.
- *NeuralPower* can also predict *runtime* of CNNs, which outperforms state-of-the-art analytical models, by achieving an improvement in accuracy up to 68.5% compared to the best previously published work.

- *NeuralPower* uses power and runtime predictions to predict the *energy consumption* of CNNs running on GPUs, with an average accuracy of 97.21%.
- In addition to total runtime and average power, *NeuralPower* also provides the detailed breakdown of runtime and power across different components (at each layer) of the whole network, thereby helping machine learners to identify efficiently the runtime, power, or energy bottlenecks.

6.2 Power and runtime modeling

In this section, we introduce our hierarchical power and runtime model framework -*NeuralPower*- and the data collection process. *NeuralPower* is based on the following key insight: despite the huge amount of different CNN variations that have been used in several applications, all these CNN architectures consist of basic underlying building blocks/primitives which exhibit similar execution characteristics per type of layer. To this end, *NeuralPower* first models the power and runtime of the key layers that are commonly used in a CNN. Then, *NeuralPower* uses these models to predict the performance and runtime of the entire network.

6.2.1 Layer-level power and runtime modeling

The first part of *NeuralPower* is layer-level power and runtime models. We construct these models for each type of layer for both runtime and power. More specifically, we select to model three types of layers, namely the *convolutional*, the *fully connected*, and the *pooling* layer, since these layers carry the main computation load during CNN execution – as also motivated by [69]. Nevertheless, unlike prior work, our goal is to make our model flexible for various combinations of software/hardware platforms without knowing the details of these platforms.

To this end, we propose a learning-based *polynomial regression model* to learn the coefficients for different layers, and we assess the accuracy of our approach against power and

runtime measurements on different commercial GPUs and Deep Learning software tools. There are three major reasons for this choice. *First*, in terms of model accuracy, polynomial models provide more flexibility and low prediction error when modeling both power and runtime. The *second* reason is the interpretability: runtime and power have clear physical correlation with the layer's configuration parameters (*e.g.*, batch size, kernel size, *etc.*). That is, the features of the model can provide an intuitive encapsulation of how the layer parameters affect the runtime and power. The *third* reason is the available amount of sampling data points. Polynomial models allow for adjustable model complexity by tuning the degree of the polynomial, ranging from linear model to polynomials of high degree, whereas a formulation with larger model capacity may be prone to overfitting. To perform model selection, we apply ten-fold cross-validation and we use Lasso to decrease the total number of polynomial terms. The detailed model selection process will be discussed in Section 6.3.1.

Layer-level runtime model: The runtime $\hat{T}$ of a layer can be expressed as:

$$\hat{T}(\mathbf{x}_T) = \sum_j c_j \cdot \prod_{i=1}^{D_T} x_i^{q_{ij}} + \sum_s c'_s \mathcal{F}_s(\mathbf{x}_T) \tag{6.1}$$

$$\text{where } \mathbf{x}_T \in \mathbb{R}^{D_T};\ q_{ij} \in \mathbb{N};\ \forall j,\ \sum_{i=1}^{D_T} q_{ij} \leq K_T.$$

The model consists of two components. The first component corresponds to the regular degree-K_T polynomial terms which are a function of the features in input vector $\mathbf{x}_T \in \mathbb{R}^{D_T}$. x_i is the i-th component of $\mathbf{x}_T$. q_{ij} is the exponent for x_i in the j-th polynomial term, and c_j is the coefficient to learn. This feature vector of dimension D_T includes layer configuration hyper-parameters, such as the batch size, the input size, and the output size. For different types of layers, the dimension D_T is expected to vary. For convolutional layers, for example, the input vector includes the kernel shape, the stride size, and the padding size, whereas such features are not relevant to the formulation/configuration of a fully-connected layer.

The second component corresponds to special polynomial terms $\mathcal{F}$, which encapsulate physical operations related to each layer (*e.g.*, the total number of memory accesses and the total number of floating point operations). The number of the special terms differs from

one layer type to another. For the convolutional layer, for example, the special polynomial terms include the memory access count for input tensor, output tensor, kernel tensor, and the total number of floating point operations for the all convolution computations. Finally, c'_s is the coefficient of the s-th special term to learn.

Based on this formulation, it is important to note that not all input parameters are positively correlated with the runtime. For example, if the stride size increases, the total runtime will decrease since the total number of convolutional operations will decrease. This observation motivates further the use of a polynomial formulation, since it can capture such trends (unlike a posynomial model, for instance).

Layer-level power model: To predict the power consumption $\hat{P}$ for each layer type during *testing*, we follow a similar polynomial-based approach:

$$\hat{P}(\mathbf{x}_P) = \sum_j z_j \cdot \prod_{i=1}^{D_P} x_i^{m_{ij}} + \sum_k z'_k \mathcal{F}_k(\mathbf{x}_P) \tag{6.2}$$

$$\text{where } \mathbf{x}_P \in \mathbb{R}^{D_P};\ m_{ij} \in \mathbb{N};\ \forall j,\ \sum_{i=1}^{D_P} m_{ij} \leq K_P.$$

where the regular polynomial terms have degree K_P and they are a function of the input vector $\mathbf{x}_P \in \mathbb{R}^{D_P}$. m_{ij} is the exponent for x_i of the j-th polynomial term, and z_j is the coefficient to learn. In the second sum, z'_k is the coefficient of the k-th special term to learn.

Power consumption however has a non-trivial correlation with the input parameters. More specifically, as a metric, power consumption has inherent limits, *i.e.*, it can only take a range of possible values constrained through the power budget. That is, when the computing load increases, power does not increase in a linear fashion. To capture this trend, we select an extended feature vector $\mathbf{x}_P \in \mathbb{R}^{D_P}$ for our power model, where we include the logarithmic terms of the features used for runtime (*e.g.*, batch size, input size, output size, *etc.*). As expected, the dimension D_P is twice the size of the input feature dimension D_T. A logarithmic scale in our features vector can successfully reflect such a trend, as supported by our experimental results in Section 6.3.

6.2.2 Network-level power, runtime, and energy modeling

We discuss the network-level models for *NeuralPower*. For the majority of CNN architectures readily available in a Deep Learning models "zoo" (as the one compiled by [43]), the whole structure consists of and can be divided into several layers in series. Consequently, using our predictions for power and runtime as building blocks, we extend our predictive models to capture the runtime, the power, and eventually the energy, of the entire architecture at the *network level*.

Network-level runtime model: Given a network with N layers connected in series, the predicted total runtime can be written as the sum of the predicted runtime $\hat{T}_n$ of each layer n:

$$\hat{T}_{total} = \sum_{n=1}^{N} \hat{T}_n \tag{6.3}$$

Network-level power model: Unlike the summation for total runtime, the average power can be obtained using both per layer runtime and power. More specifically, we can represent the average power $\hat{P}_{avg}$ of a CNN as:

$$\hat{P}_{avg} = \frac{\sum_{n=1}^{N} \hat{P}_n \cdot \hat{T}_n}{\sum_{n=1}^{N} \hat{T}_n} \tag{6.4}$$

Network-level energy model: From here, it is easy to derive our model for the total energy consumption $\hat{E}_{total}$ of an entire network configuration:

$$\hat{E}_{total} = \hat{T}_{total} \cdot \hat{P}_{avg} = \sum_{n=1}^{N} \hat{P}_n \cdot \hat{T}_n \tag{6.5}$$

which is basically the scalar product of the layer-wise power and runtime vectors, or the sum of energy consumption for all layers in the model.

6.2.3 Dataset collection

Experiment setup: The main modeling and evaluation steps are performed on the platform described in Table 6.1. To exclude the impact of voltage/frequency changing on the power and runtime data we collected, we keep the GPU in a fixed state and CUDA libraries ready to

use by enabling the persistence mode. We use `nvidia-smi` to collect the instantaneous power per 1 ms for the entire measuring period. Please note that while this experimental setup constitutes our configuration basis for investigating the proposed modeling methodologies, in Section 6.3.4 we present results of our approach on other GPU platforms, such as Nvidia GTX 1070, and Deep Learning tools, such as Caffe by [43].

Table 6.1: Target platform.

CPU / Main memory	Intel Core-i7 5820K / 32GB
GPU	Nvidia GeForce GTX Titan X (12GB DDR5)
GPU max / idle power	250W / 15W
Deep learning platform	TensorFlow 1.0 on Ubuntu 14
Power meter	NVIDIA System Management Interface

CNN architectures investigated: To comprehensively assess the effectiveness of our modeling methodology, we consider several CNN models. Our analysis includes state-of-the-art configurations, such as the AlexNet by [51], VGG-16 & VGG-19 by [76], R-CNN by [70], NIN network by [59], CaffeNet by [43], GoogleNet by [83], and Overfeat by [74]. We also consider different flavors of smaller networks such as vanilla CNNs used on the MNIST by [53] and CIFAR10-6conv [20] on CIFAR-10. This way, we can cover a wider spectrum of CNN applications.

Data collection for power/runtime models: To train the layer-level predictive models, we collect data points by profiling power and runtime from all layers of all the considered CNN architectures in the training set. We separate the training data points into groups based on their layer types. In this chapter, the training data include 858 convolution layer samples, 216 pooling layer samples, and 116 fully connected layer samples. The statistics can change if one needs any form of customization. For testing, we apply our learned models on the network in the testing set, and compare our predicted results against the actual results profiled on the same platform, including both layer-level evaluation and network-level evaluation.

6.3 Experimental results

In this section, we assess our proposed *NeuralPower* in terms of power, runtime, and energy prediction accuracy at both layer level and network level. Since the models for runtime and power are slightly different from one to another, we discuss them separately in each case. In addition, we validate our framework on other platforms to show the robustness of *NeuralPower*.

6.3.1 Layer-level model evaluation

Model selection

To begin with model evaluation, we first illustrate how model selection has been employed in *NeuralPower*. In general, *NeuralPower* changes the order of the polynomial (*e.g.*, D_T in Equation 6.1) to expand/shrink the size of feature space. *NeuralPower* applies Lasso to select the best model for each polynomial model. Finally, *NeuralPower* selects the final model with the lowest cross-validation Root-Mean-Square-Error (RMSE), which is shown in Figure 6.2.

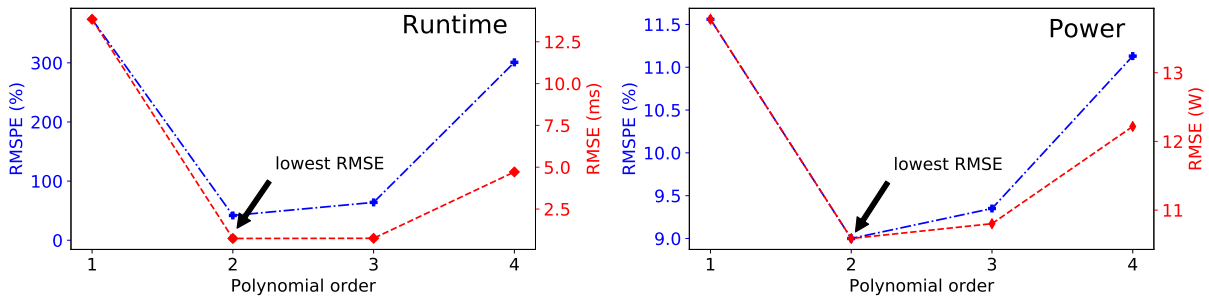

Figure 6.2: Comparison of best-performance model with respect to each polynomial order for the fully-connected layers. In this example, a polynomial order of two is chosen since it achieves the best Root-Mean-Square-Error (RMSE) for both runtime and power modeling. At the same time, it also has the lowest Root-Mean-Square-Percentage-Error (RMSPE).

Runtime Models

Applying the model selection process, we achieve a polynomial model for each layer type in a CNN. The evaluation of our models is shown in Table 6.2, where we report the Root-Mean-Square-Error (RMSE) and the relative Root-Mean-Square-Percentage-Error (RMSPE) of our runtime predictions for each one of the considered layers. Since we used Lasso in our model selection process, we also report the model size (*i.e.*, the number of terms in the polynomial) per layer. More importantly, we compare against the state-of-the-art analytical method proposed by [69], namely Paleo. To enable a comparison here and for the remainder of section, we executed the Paleo code on the considered CNNs. We can easily observe that our predictions based on the layer-level models clearly outperform the best published model to date, yielding an *improvement in accuracy* up to 68.5% (calculated from the differences of RMSPEs for pooling layer).

Table 6.2: Comparison of runtime models for common CNN layers – Our proposed runtime model consistently outperforms the state-of-the-art runtime model in both root-mean-square-error (RMSE) and the Root-Mean-Square-Percentage-Error (RMSPE).

Layer type	***NeuralPower***			Paleo [69]	
	Model size	RMSPE	RMSE (ms)	RMSPE	RMSE (ms)
Convolutional	60	39.97%	1.019	58.29%	4.304
Fully-connected	17	41.92%	0.7474	73.76%	0.8265
Pooling	31	11.41%	0.0686	79.91%	1.763

Convolutional layer: The convolution layer is among the most time- and power-consuming components of a CNN. To model this layer, we use a polynomial model of degree three. We select a features vector consisting of the batch size, the input tensor size, the kernel size, the stride size, the padding size, and the output tensor size. In terms of the special terms defined in Equation 6.1, we use terms that represent the total computation operations and memory accesses count.

Fully-connected layer: We employ a regression model with degree of two, and as features of the model we include the batch size, the input tensor size, and the output tensor size. It is worth noting that in terms of software implementation, there are two common

ways to implement the fully-connected layer, either based on default matrix multiplication, or based on a convolutional-like implementation (*i.e.*, by keeping the kernel size exactly same as the input image patch size). Upon profiling, we notice that both cases have a tensor-reshaping stage when accepting intermediate results from a previous convolutional layer, so we treat them interchangeably under a single formulation.

Pooling layer: The pooling layer usually follows a convolution layer to reduce the complexity of the model. As basic model features we select the input tensor size, the stride size, the kernel size, and the output tensor size. Using Lasso and cross-validation we find that a polynomial of degree three provides the best accuracy.

Power Models

As mentioned in Section 6.2.1, we use the logarithmic terms of the original features (*e.g.*, batch size, kernel size, *etc.*) as additional features for the polynomial model since this significantly improves the model accuracy. This modeling choice is well suited for the nature of power consumption which does not scale linearly; more precisely, the rate of the increase in power goes down as the model complexity increases, especially when the power values get closer to the power budget limit. For instance, in our setup, the Titan X GPU platform has a maximum power of 250W. We find that a polynomial order of two achieves the best cross validation error for all three layer types under consideration.

To the best of our knowledge, there is no prior work on power prediction at the layer level to compare against. We therefore compare our methods directly with the actual power values collected from TensorFlow, as shown in Table 6.3. Once again, we observe that our proposed model formulation achieves error always less that 9% for all three layers. The slight increase in the model size compared to the runtime model is to be expected, given the inclusion of the logarithmic feature terms, alongside special terms that include memory accesses and operations count. We can observe, though, that the model is able to capture the trends of power consumption trained across layer sizes and types.

Table 6.3: Power model for common CNN layers.

Layer type	***NeuralPower***		
	Model size	RMSPE	RMSE (W)
Convolutional	75	7.35%	10.9172
Fully-connected	15	9.00%	10.5868
Pooling	30	6.16%	6.8618

6.3.2 Network-level model evaluation

With the results from layer-level models, we can model the runtime, power, and energy for the whole network based on the network-level model (Section 6.2.2) in *NeuralPower*. To enable a comprehensive evaluation, we assess *NeuralPower* on several state-of-the-art CNNs, and we compare against the actual runtime, power, and energy values of each network. For this purpose, and as discussed in Section 6.2.3, we leave out a set of networks to be used only for testing, namely the VGG-16, NIN, CIFAR10-6conv, AlexNet, and Overfeat networks.

Runtime evaluation

Prior to assessing the predictions on the networks as a whole, we show the effectiveness of *NeuralPower* as a useful aid for CNN architecture benchmarking and per-layer profiling. Enabling such breakdown analysis is significant for machine learning practitioners, since it allows to identify the bottlenecks across components of a CNN.

For runtime, we use state-of-the-art analytical model Paleo as the baseline. In Figure 6.3, we compare runtime prediction models from *NeuralPower* and the baseline against actual runtime values of each layer in the NIN and VGG-16 networks. From Figure 6.3, we can clearly see that our model outperforms the Paleo model for most layers in accuracy. For the NIN, our model clearly captures that *conv4* is the dominant (most time-consuming) layer across the whole network. However, Paleo erroneously identifies *conv2* as the dominant layer. For the VGG-16 network, we can clearly see that Paleo predicts the runtime of the first fully-connected layer *fc6* as 3.30 ms, with a percentage prediction error as high as -96.16%. In contrast, our prediction exhibits an error as low as -2.53%. Since layer *fc6* is the

dominant layer throughout the network, it is critical to make a correct prediction on this layer.

From the above, we can conclude that our proposed methodology generally has a better accuracy in predicting the runtime for each layer in a complete CNN, especially for the layers with larger runtime values. Therefore, our accurate runtime predictions, when employed for profiling each layer at the network level, can help the machine learners and practitioners quickly identify the real bottlenecks with respect to runtime for a given CNN.

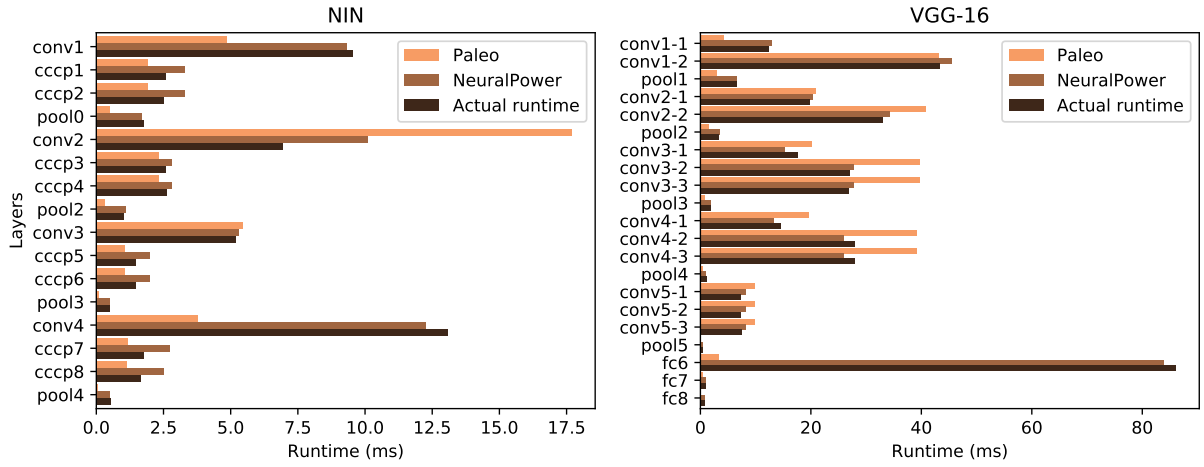

Figure 6.3: Comparison of runtime prediction for each layer in NIN and VGG-16: Our models provide accurate runtime breakdown of both network, while Paleo cannot. Our model captures the execution-bottleneck layers (*i.e.*, *conv4* in NIN, and *fc6* in VGG-16) while Paleo mispredicts both.

Having demonstrated the effectiveness of our methodology at the layer level, we proceed to assess the accuracy of the network-level runtime prediction $\hat{T}_{total}$ (Equation 6.3). It is worth observing that in Equation 6.3 there are two sources of potential error. First, error could result from mispredicting the runtime values $\hat{T}_n$ per layer n. However, even if these predictions are correct, another source of error could come from the formulation in Equation 6.3, where we assume that the sum of the runtime values of all the layers in a CNN provides a good estimate of the total runtime. Hence, to achieve a comprehensive evaluation of our modeling choices in terms of both the regression formulation and the summation in Equation 6.3, we need to address both these concerns.

To this end, we compare our runtime prediction $\hat{T}_{total}$ against two metrics. *First*, we compare against the actual overall runtime value of a network, notated as T_{total}. *Second*, we consider another metric defined as the sum of the actual runtime values T_n (and not the predictions) of each layer n:

$$\mathbb{T}_{total} = \sum_{n=1}^{N} T_n \tag{6.6}$$

Intuitively, a prediction value $\hat{T}_{total}$ close to both the $\mathbb{T}_{total}$ value and the actual runtime T_{total} would not only show that our model has good network-level prediction, but also that our underlying modeling assumptions hold.

We summarize the results across five different networks in Table 6.4. More specifically, we show the networks' actual total runtime values (T_{total}), the runtime $\mathbb{T}_{total}$ values, our predictions $\hat{T}_{total}$, and the predictions from Paleo (the baseline). Based on the Table, we can draw two key observations. First, we can clearly see that our model always outperforms Paleo, with runtime predictions always within 24% from the actual runtime values. Compared to the actual power value, our prediction have an RMSPE of 11.76%, or 88.24% in accuracy. Unlike our model, prior art could underestimate the overall runtime up to 42%. Second, as expected, we see that summing the true runtime values per layer does indeed approximate the total runtime, hence confirming our assumption in Equation 6.3.

Table 6.4: Performance model comparison for the whole network. We can easily observe that our model always provides more accurate predictions of the total CNN runtime compared to the best published model to date (Paleo). We assess the effectiveness of our model in five different state-of-the-art CNN architectures.

CNN name	Paleo [69] (ms)	***NeuralPower*** $\hat{T}_{total}$ (ms)	Sum of per-layer actual runtime $\mathbb{T}_{total}$ (ms)	Actual runtime T_{total} (ms)
VGG-16	345.83	373.82	375.20	368.42
AlexNet	33.16	43.41	42.19	39.02
NIN	45.68	62.62	55.83	50.66
Overfeat	114.71	195.21	200.75	197.99
CIFAR10-6conv	28.75	51.13	53.24	50.09

Power evaluation

We present a similar evaluation methodology to assess our model for network-level power predictions. We first use our methodology to enable a per-layer benchmarking of the power consumption. Figure 6.4 shows the comparison of our power predictions and the actual power values for each layer in the NIN and the VGG-16 networks. We can see that convolutional layers dominate in terms of power consumption, while pooling layers and fully connected layers contribute relatively less. We can also observe that the convolutional layer exhibits the largest variance with respect to power, with power values ranging from 85.80W up to 246.34W.

Another key observation is related to the fully-connected layers of the VGG-16 network. From Figure 6.3, we know layer *fc6* takes the longest time to run. Nonetheless, we can see in Figure 6.4 that its power consumption is relatively small. Therefore, the energy consumption related of layer *fc6* will have a smaller contribution to the total energy consumption of the network relatively to its runtime. It is therefore evident that using only the runtime as a proxy proportional to the energy consumption of CNNs could mislead the machine learners to erroneous assumptions. This observation highlights that power also plays a key role towards representative benchmarking of CNNs, hence illustrating further the significance of accurate power predictions enabled from our approach.

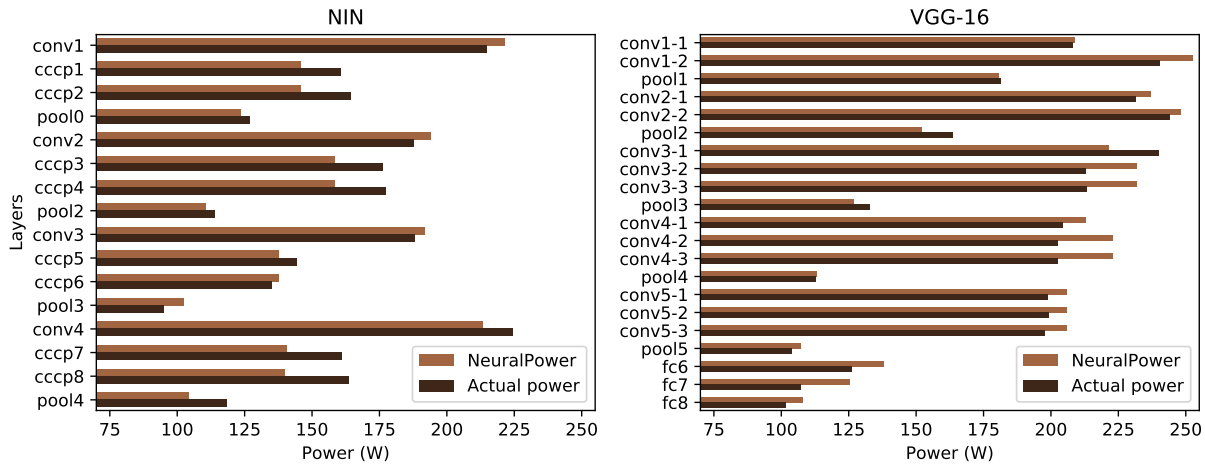

Figure 6.4: Comparison of power prediction for each layer in NIN and VGG-16.

Table 6.5: Evaluating our power predictions for state-of-the-art CNN architectures.

CNN name	***NeuralPower*** $\hat{P}_{total}$ (W)	Sum of per-layer actual power $\mathbb{P}_{total}$ (W)	Actual power P_{avg} (W)
VGG-16	206.88	195.76	204.80
AlexNet	174.25	169.08	194.62
NIN	179.98	187.99	226.34
Overfeat	172.20	168.40	172.30
CIFAR10-6conv	165.33	167.86	188.34

As discussed in the runtime evaluation as well, we assess both our predictive model accuracy and the underlying assumptions in our formulation. In terms of average power consumption, we need to confirm that the formulation selected in Equation 6.4 is indeed representative. To this end, besides the comparison against the actual average power of the network P_{avg}, we compare against the average value $\mathbb{P}_{avg}$, which can be computed by replacing our predictions $\hat{P}_n$ and $\hat{T}_n$ with the actual per-layer runtime and power values:

$$\mathbb{P}_{avg} = \frac{\sum_{n=1}^{N} P_n \cdot T_n}{\sum_{n=1}^{N} T_n} \tag{6.7}$$

We evaluate our power value predictions for the same five state-of-the-art CNNs in Table 6.5. Compared to the actual power value, our prediction have an RMSPE of 11.66%, or 88.34% in accuracy. We observe that in two cases, AlexNet and NIN, our prediction has a larger error, *i.e.*, of 10.47% and 20.48% respectively. This is to be expected, since our formulation for $\mathbb{P}_{avg}$ depends on runtime prediction as well, and as observed previously in Table 6.4, we underestimate the runtime in both cases.

Energy evaluation

Finally, we use Equation 6.5 to predict the total energy based on our model. To evaluate our modeling assumptions as well, we compute the energy value $\mathbb{E}_{total}$ based on the actual per-layer runtime and power values, defined as:

$$\mathbb{E}_{total} = \sum_{n=1}^{N} P_n \cdot T_n \tag{6.8}$$

We present the results for the same five CNNs in Table 6.6. We observe that our approach enables good prediction, with an average RMSPE of 2.79%, or 97.21% in accuracy.

Table 6.6: Evaluating our energy predictions for state-of-the-art CNN architectures.

CNN name	***NeuralPower*** $\hat{E}_{total}$ (J)	Sum of per-layer actual energy $\mathbb{E}_{total}$ (J)	Actual energy E_{total} (J)
VGG-16	77.312	73.446	75.452
AlexNet	7.565	7.134	7.594
NIN	11.269	10.495	11.465
Overfeat	33.616	33.807	34.113
CIFAR10-6conv	8.938	8.453	9.433

6.3.3 Energy-precision ratio

In this subsection, we propose a new metric, namely *Energy-Precision Ratio* to be used as a guideline for machine learners towards accurate, yet energy efficient CNN architectures. We define the metric as:

$$EPR = Error^{\alpha} \cdot EPI \tag{6.9}$$

where $Error$ is the data classification error of the CNN under consideration, and EPI is the energy consumption per data item classified. Different values of the parameter α dictate how much importance is placed on the accuracy of the model, since a larger α places more weight on the CNN classification error. To illustrate how α affects the results, in Table 6.7 we compute the EPR score values for VGG-16, AlexNet, and NIN, all trained on ImageNet datasets (as $Error$ value we use their Top-5 error). We also use our predictive model for energy, and we compute the energy consumption per image classification.

Intuitively, a lower EPR value indicates a better trade-off between energy-efficiency and accuracy of a CNN architecture. For instance, we can see that while VGG-16 has the lowest error, this comes at the price of increased energy consumption compared to both NIN and AlexNet. Hence, for $\alpha = 1$ both AlexNet and NIN have a smaller EPR value. In this case, a machine learner of an embedded Internet-of-Things (IoT) system could use the *Energy-Precision Ratio* to select the most energy efficient architecture. On the contrary, with $\alpha = 4$,

i.e., when accuracy is being heavily favored over energy efficiency, the EPR value of VGG-16 is smaller than EPR value of AlexNet.

Table 6.7: EPR metric for different CNN architectures and Energy-per-Image (EPI) values. Network choices could be different for different α values: AlexNet for $\alpha = 1, 2, 3$, VGG-16 for $\alpha = 4$.

CNN name	Top-5 Error	EPI (mJ)	EPR			
			$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$
VGG-16	7.30%	1178.9	86.062	6.283	0.459	**0.033**
AlexNet	17.00%	59.3	**10.086**	**1.715**	**0.291**	0.050
NIN	20.20%	89.6	18.093	3.655	0.738	0.149

With the recent surge of IoT and mobile learning applications, machine learners need to take the energy efficiency of different candidate CNN architectures into consideration, in order to identify the CNN which is more suitable to be deployed on a energy-constrained platform. For instance, consider the case of a machine learner that has to choose among numerous CNNs from a CNN model "zoo", with the best error of each CNN readily available. *Which network provides the best trade-off between accuracy for energy spent per image classified?* Traditionally, the main criterion for choosing among CNN architectures has been the data item classification accuracy, given its intuitive nature as a metric. However, there has not been so far an easily interpretable metric to trade-off energy efficiency versus accuracy.

Towards this direction, we can use our proposed model to quickly predict the total energy consumptions of all these different architectures, and we can then compute the EPR score to select the one that properly trades off between accuracy for energy spent per image classified. We postulate that the *Energy-Precision Ratio*, alongside our predictive model, could therefore be used as a useful aid for machine learners *towards energy-efficient CNNs.*

6.3.4 Models on other platforms and development frameworks

Our key goal is to provide a modeling framework that could be flexibly used for different platforms. To comprehensively demonstrate this property of our work, we extend our evalu-

ation to a different GPU platform, including desktop GPU - Nvidia GTX 1070, and mobile GPU - Nvidia Jetson TX1.

Results on Nvidia GTX 1070

We first apply our framework to another GPU platform, and more specifically to the Nvidia GTX 1070 with 8GB memory. We repeat the runtime and power data collection by executing Tensorflow, and we train power and runtime models on this platform. The layer-wise evaluation results are shown in Table 6.8. For these results, we used the same polynomial orders as reported previously for the TensorFlow on Titan X experiments. Moreover, we evaluate the overall network prediction for runtime, power, and energy values and we present the predicted values and the prediction error (denoted as Error) in Table 6.9. Based on these results, we can see that our methodology achieves consistent performance across different desktop GPU platforms.

Table 6.8: Runtime and power model for all layers using TensorFlow on GTX 1070.

Layer type	Runtime			Power		
	Model size	RMSPE	RMSE (ms)	Model size	RMSPE	RMSE (W)
Convolutional	10	57.38 %	3.5261	52	10.23%	9.4097
Fully-connected	18	44.50%	0.4929	23	7.08%	5.5417
Pooling	31	11.23%	0.0581	40	7.37%	5.1666

Table 6.9: Evaluation of *NeuralPower* on CNN architectures using TensorFlow on GTX 1070.

CNN name	Runtime		Power		Energy	
	Value (ms)	Error	Value (W)	Error	Value (J)	Error
AlexNet	44.82	17.40%	121.21	-2.92%	5.44	13.98%
NIN	61.08	7.24%	120.92	-4.13%	7.39	2.81%

Extending to other machine learning software environments: Caffe

Finally, we demonstrate the effectiveness of our approach across different Deep Learning software packages and we replicate our exploration on Caffe. To collect the power and

runtime data for our predictive models, we use Caffe's supported mode of operation for benchmarking, namely `time`. While this functionality benchmarks the model execution layer-by-layer, the default Caffe version however reports only timing. To this end, we extend Caffe's C++ codebase and we incorporate calls to Nvidia's NVML C++ API to report power values.

We present the per-layer accuracy for runtime and power predictions in Table 6.10. Furthermore, we evaluate our model on the AlexNet and NIN networks in Table 6.11. Please note that the execution of the entire network corresponds to a different routine under the Caffe framework, so a direct comparison is not possible. We instead compare against the Equations 6.6-6.8 as in the previous subsection. Same as for the TensorFlow case before (Table 6.8), we observe that in this case as well the runtime predictions exhibit a larger error in this platform when executing on the GTX1070 system. This is to be expected, since the GTX 1070 drivers do not allow the user to clock the GPU to a particular frequency state, hence the system dynamically selects its execution state. Indeed, in our collected datapoints we observed that Caffe's (and TensorFlow's previously) higher variance in the runtime values. To properly capture this behavior, we experimented with regressors other that second and third degree polynomials. In these results for Caffe in particular, we select linear models since they provided a better trade off between training error and overfitting.

Table 6.10: Runtime and power model for all layers using Caffe on GTX 1070.

Layer type	Runtime			Power		
	Model size	RMSPE	RMSE (ms)	Model size	RMSPE	RMSE (W)
Convolutional	32	45.58 %	2.2301	32	6.19%	11.9082
Fully-connected	18	48.41 %	0.6626	18	8.63%	8.0291
Pooling	30	37.38 %	0.1711	26	6.72 %	11.9124

Table 6.11: Evaluation of our model on CNN architectures using Caffe on GTX 1070.

CNN name	Runtime		Power		Energy	
	Value (ms)	Error	Value (W)	Error	Value (J)	Error
AlexNet	51.18	-31.97%	107.63	-5.07 %	5.51	35.42%
NIN	76.32	0.36 %	109.78	-8.89%	8.38	8.56%

Results on Nvidia Jetson TX1

In addition to the traditional high-performance GPUs, many of the CNNs are running on the mobile platforms to enable various services locally. Therefore, we apply our models on the mobile platform to show that our method are versatile and widely suitable for various platforms.

Table 6.12: Runtime and power model for all layers using TensorFlow on Jetson TX1.

Layer type	Runtime			Power		
	Model size	RMSPE	RMSE (ms)	Model size	RMSPE	RMSE (W)
Convolutional	44	20.00 %	0.437	48	21.69%	0.825
Fully-connected	12	32.64%	0.871	31	7.72%	0.103
Pooling	53	19.34%	0.2307	37	13.81%	0.315

We present the per-layer accuracy for runtime and power predictions in Table 6.12. Different than GTX Titan X or 1070, Jetson TX1 has very limited memory resource and power cap. Therefore, it cannot run large CNNs or CNNs with large batch size. For this reason, the CNN architecture set used for Jetson TX1 is a little bit different than the one used in the previous experiments. However, we can still see that *NeuralPower* is robust on mobile GPU platforms. Furthermore, we evaluate our model on the Cifar10-6conv and NIN networks in Table 6.13. Different with the results shown for GTX 1070, the results for Jetson TX1 show the trend as underestimating the runtime, and overestimating the power consumption. The similar thing is that they still have similar power, runtime, or energy consumption error level.

Table 6.13: Evaluation of *NeuralPower* on CNN architectures using TensorFlow on Jetson TX1.

CNN name	Runtime		Power		Energy	
	Value (ms)	Error	Value (W)	Error	Value (J)	Error
Cifar10-6conv	26.69	-13.10%	11.39	18.79%	0.304	3.23%
NIN	75.38	-8.41%	8.23	2.53%	0.620	-6.09%

According to these results, we can conclude that our methodology achieves consistent performance across different GPU platforms, thus enabling a scalable/portable framework from machine learning practitioners to use across different systems.

6.4 Discussion

It is important to note that the overhead introduced by *NeuralPower* is very limited. More specifically, *NeuralPower* needs to collect datasets to train the models, however, the overhead for training is very small, e.g., around 30 minutes for GTX Titan X. This includes data collection (under 10 minutes) and model training (less than 20 minutes). The process is done once for a new platform. However, the model training process can be further shorten by exploiting transfer learning, which exploits the similarity between different platforms. For example, we find that the best performing models for GTX 1070 have the same polynomial order as the corresponding models for GTX Titan X. Even for Jetson TX1, nearly all best performing models share the same polynomial order as the corresponding models for GTX Titan X, except the runtime model for fully-connected layers. In this case, the model training process, especially the model selection phase, can be greatly simplified. Therefore, the overhead would be negligible. If the training process starts from the scratch, the overhead can still be offset if the CNN architecture search space is large. Even if machine learners only evaluate a few CNN architectures, *NeuralPower* can still provide the detailed breakdown with respect to runtime, power, and energy to identify bottlenecks and possible improvement directions.

Chapter 7

Related work

In this chapter, we discuss related work with respect to two topics: technology-aware computing system design and application-driven computing system design.

In technology-aware computing system design, there is a significant body of work addressing power/performance optimization via DVFS [40] [33] [88] [73] [61] [7] [18] [47] [48] [60]. Isci *et al.* [40] proposed MaxBIPS, an exhaustive search algorithm to find the best V/F levels to maximize performance under a given power budget. When considering process variations, Teodorescu *et al.* [84] used linear programming LinOpt to find the optimal voltage and frequency levels for maximizing throughput, while Herbert *et al.* [34] employed algorithm tuning. Hardware-based approaches for DVFS were first introduced by Choudhary *et al.* [19] [62]. Sartori *et al.* [73] proposed three scalable approaches to improve many-core throughput under given power budget. Winter *et al.* [88] proposed Steepest Drop, a directed local search algorithm to explore best V/F level selection. Chen *et al.* [17] applied reinforcement learning for DVFS-based performance optimization under power limits.

When applying DVFS, the effect of process variations (PVs) becomes increasingly important, especially for FinFET technology. In fact, PVs have been extenstively explored, albeit mainly for bulk CMOS technology in the context of architectural and system level exploration [63] [64] [34] [35] [32] [81]. Agarwal *et al.* [1] proposed a statistical timing analysis for the PVs considering spatial correlations. Sarangi *et al.* [72] constructed a model of PV

and timing errors for microarchitects on bulk CMOS. For the FinFET technology, Xiong *et al.* [90] investigated the sensitivity of FinFET based devices to PVs. Chaudhuri *et al.* [16] established an accurate and fast 2D FinFET simulation model including the impact of PVs in FinFET.

In addition to the PVs, FinFET technology has another feature in focus, called temperature effect inversion (TEI). TEI has been studied for more than a decade [39] [49] [54] [9] [11] [28] [95] [96] [56] [27]. Huang *et al.* [39] showed that fabricated FinFETs operate faster when temperature increases in superthreshold voltage region. Kim *et al.* [49] analyzed this effect for both n- and p-channel FinFET and explained that it is caused by stress-induced bandgap narrowing. Soleimani *et al.* [79] and Lee *et al.* [54], also reported and discussed this effect. Lee *et al.* [57] exploited TEI in FinFET-based circuits and proposed a dynamic thermal management policy for FinFET-based mobile devices. Furthermore, they named this effect Temperature Effect Inversion.

The effects of PVs and TEI in FinFET technology are always related with thermal issues. Among them, the effect of aging is significant. Many papers aim to capture aging-induced performance degradation. Kukner *et al.* [52] has provided a comprehensive investigation of BTI modeling as technology scales from planar FET to advanced 3-D FinFET devices based on Ring Oscilator (RO) simulations. Weckx *et al.* [86] employed this exploration to both RO and SRAM based netlists. However, this research does not extend the analysis beyond gate-level simulations and it does not explore the aging implications to the system level. Recently, system-wide BTI-aware flows have been proposed, aiming to either capture the aging-induced path re-ordering of CPU modules [82] or to decrease aging on dark-silicon systems [25]. Nonetheless, all proposed system-level design methodologies consider planar CMOS technologies and they restrict their analysis to one frequency/voltage level. Such assumption is non-representative of the extended operating range available in modern systems [15].

For application-driven computing system design, we mainly discuss convolutional neural networks (CNNs). Prior art, *e.g.*, by [30], has identified the runtime overhead and power

consumption of CNNs execution to be a significant concern when designing accurate, yet power- and energy-efficient CNN models. These design constraints become increasingly challenging to address, especially in the context of two emerging design paradigms, *(i)* the design of larger, power-consuming CNN architectures, and *(ii)* the design of energy-aware CNNs for mobile platforms. Hence, a significant body of state-of-the-art approaches aim to tackle such runtime and power overhead, by accelerating the execution of CNNs and/or by reducing their power-energy consumption.

To enable CNN architectures that execute faster, existing work has investigated both hardware- and software-based methodologies. On one hand, with respect to hardware, several hardware platforms have been explored as means to accelerate the CNN execution. Examples of these platforms include FPGA-based methodologies by [92], ASIC-like designs by [93]. On the other hand, with respect to software-based acceleration, several libraries, such as cuDNN and Intel MKL, have been used in various deep learning frameworks to enable fast execution of CNNs, since these libraries provide specially-optimized primitives in CNNs to improve their runtime.

To reduce power or energy consumption, recent work has focused on limiting the energy and power consumption of CNNs. Several approaches investigate the problem in the context of hyper-parameter optimization [71] [78] [80]. For instance, [71] have proposed an automated customization methodology that adaptively conforms the CNN architectures to the underlying hardware characteristics, while minimally affecting the inference accuracy. [78] have used a pair of trainer-trainee CNNs to incorporate accuracy and cost into the design of neural networks. Beyond these methods which are based on hyper-parameter optimization, another group of novel approaches focuses solely on the energy-efficient CNN implementation assuming a given network architecture. These approaches include techniques that draw ideas from energy-aware computer system design, such as the methodologies by [30], [21] and [22].

However, there is no comprehensive methodology that models the runtime, power, and eventually the energy of CNN architectures. A work that shares similar insight with our

methodology is the *Paleo* framework proposed by [69]. In their approach, the authors present an analytical method to determine the runtime of CNNs executing on various platforms. However, their model cannot be flexibly used across different platforms with different optimization libraries, without detailed knowledge of them. More importantly, their approach cannot predict power and energy consumption.

Chapter 8

Conclusion and future work

In this thesis, we have showed that our learning-based modeling methodologies work not only for energy-aware computing systems, incorporating many scenarios like process variation, temperature effect inversion, aging effects, but also for application-driven computing systems, such as convolutional neural networks. In addition, we propose several fast and efficient optimization techniques to further save energy, increase performance, or extend lifetime of these computing systems. This chapter summarize the key results presented in the previous chapters to conclude this thesis.

First, we include PVs in power/performance modeling and optimization for multi-core systems. To be specific, we adapt the model-selecting technique and LOOCV from machine learning to learn the best PV-aware constrained-posynomial $\hat{P}$ for modeling the workload-dependent power-frequency relationship over an extended range for FinFET-based CMPs. Based on the convexity provided by the learned $\hat{P}$, two optimization frameworks are proposed: energy minimization under throughput constraints and throughput maximization under power constraints. Experimental results shows PV-ExDVFS achieves an average (1) **22.88%** or **31.09%** power reduction under iso-performance conditions and (2) **6.25%** or **11.46%** throughput improvement under iso-power conditions, with mild (3σ = 10%) or extreme (30%) PV levels, respectively. We validate the effectiveness of PV-ExDVFS by quantifying the impact of discrete V/F levels and core cluster sizes.

Second, in parallel with PVs, we discuss the impact of TEI and how to utilize TEI on multi-core systems. We construct accurate system-level power and performance models and apply them to evaluate the impact of TEI. We discover the TEI-induced sweet spots, which are locally better V/F level choices. Based on these sweet spots, we propose our fast algorithms, including TEI-Turbo determining the maximum throughput with a given power budget and TEI-LP determining the minimum energy consumption for a given throughput constraint. Experimental results shows that in a 16-core CMP setup, our algorithms achieve an average speedup of 45.9× with losing only 0.22% in throughput (TEI-Turbo), or an average speedup of 55.3× with only 0.68% more energy consumption (TEI-LP), compared with TEI-aware versions of existing state-of-the-art algorithms. Results also show excellent scalability of our algorithms in both core count and number of V/F levels.

Third, to relieve the potential thermal issues associated with TEI, we explore the methodology for aging reduction. We are the *first* to provide a comprehensive evaluation of both TEI and aging effects on the performance and power of FinFET-based multi-core systems with multiple voltage/frequency levels. Moreover, we propose an aging-aware algorithm, dubbed *AgingMin*, to select the optimal TEI-aware voltage/frequency operation points for decreasing the aging effects. Experimental results show *AgingMin* improves the classic 10-year system lifetime by an average of 1.61 years while introducing less than 1% power overhead when compared to existing state-of-the-art directed local search algorithm. It is worth noting that that *AgingMin* is an orthogonal algorithm to the existing aging reduction methods, such as thread mapping and dark silicon designs. Thus, our proposed algorithm can be flexibly incorporated within existing aging mitigation techniques to achieve further aging reduction.

Finally, we move to a higher level to model the application-aware computing systems, *i.e.*, the runtime, power, and energy consumption for CNNs. We propose *NeuralPower*, the first holistic framework to provide an accurate estimate of power, runtime, and energy consumption. The runtime model of *NeuralPower* outperforms the current state-of-the-art predictive model in terms of accuracy. Furthermore, *NeuralPower* can be used to provide an accurate breakdown of a CNN network, helping machine learners identify the bottlenecks

of their designed CNN models. Finally, we assess the accuracy of predictions at the network level, by predicting the runtime, power, and energy of state-of-the-art CNN configurations. *NeuralPower* achieves an average accuracy of 88.24% in runtime, 88.34% in power, and 97.21% in energy.

Future work in technology-aware computing system design can explore TEI actively in harsh environments, such as the high-performance processors in autonomous vehicles. As it is already known, TEI is dependent on temperature. Harsh environments require the processors to work with fast-changing and wide-ranging ambient temperatures. Therefore, it is important to explore TEI by controlling the operating temperature to save energy or increase performance under certain constraints. For future work in application-driven computing system design, one could extend the current models to work for other types of deep neural network on a wide variety of hardware platforms. Recurrent neural network is another popular type of deep neural networks which also needs accurate models. Nowadays, CNNs are running on various type of platforms, including FPGAs, ASICs, CPUs, etc. Therefore, it is important to extend current models to solve the challenges related to both new applications and new platforms.

Bibliography

[1] Aseem Agarwal, David Blaauw, and Vladimir Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 900. IEEE Computer Society, 2003.

[2] Lal Shimpi Anand. Nehalem - everything you need to know about intel's new architecture. *http://www.anandtech.com/show/2594/2*, 2008.

[3] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[4] W Bircher, Jason Law, Madhavi Valluri, and Lizy K John. Effective use of performance monitoring counters for run-time prediction of power. *University of Texas at Austin Technical Report TR-041104*, 1, 2004.

[5] Christopher M Bishop. *Pattern recognition and machine learning*. springer New York, 2006.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[7] Ermao Cai, Da-Cheng Juan, Siddharth Garg, Jinpyo Park, and Diana Marculescu. Learning-based power/performance optimization for many-core systems with extended-range voltage/frequency scaling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(8):1318–1331, 2016.

[8] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. In *Asian Conference on Machine Learning*, pages 622–637, 2017.

[9] Ermao Cai and Diana Marculescu. Tei-turbo: Temperature effect inversion-aware turbo boost for finfet-based multi-core systems. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 500–507. IEEE Press, 2015.

[10] Ermao Cai and Diana Marculescu. Temperature effect inversion-aware power-performance optimization for finfet-based multicore systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(11):1897–1910, 2017.

[11] Ermao Cai, Dimitrios Stamoulis, and Diana Marculescu. Exploring aging deceleration in finfet-based multi-core systems. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 111. ACM, 2016.

[12] YU Cao, Takashi Sato, Dennis Sylvester, Michael Orshansky, and C Hu. Predictive technology model. *Internet: http://ptm. asu. edu*, 2012.

[13] Trevor E Carlson, Wim Heirman, and Lieven 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*, page 52. ACM, 2011.

[14] Hongliang Chang and Sachin S Sapatnekar. Statistical timing analysis under spatial correlations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(9):1467–1482, 2005.

[15] James Charles, Preet Jassi, Narayan S Ananth, Abbas Sadat, and Alexandra Fedorova. Evaluation of the intel® core i7 turbo boost feature. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pages 188–197. IEEE, 2009.

[16] Sourindra Chaudhuri and Niraj K Jha. 3d vs. 2d analysis of finfet logic gates under process variations. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 435–436. IEEE, 2011.

[17] Zhuo Chen and Diana Marculescu. Distributed reinforcement learning for power limited many-core system performance optimization. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 1521–1526. EDA Consortium, 2015.

[18] Zhuo Chen, Dimitrios Stamoulis, and Diana Marculescu. Profit: priority and power/performance optimization for many-core systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[19] Puru Choudhary and Diana Marculescu. Power management of voltage/frequency island-based systems using hardware-based methods. *TVLSI*, 17(3):427–438, 2009.

[20] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131, 2015.

[21] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[22] Ruizhou Ding, Zeye Liu, Rongye Shi, Diana Marculescu, and RD Blanton. Lightnn: Filling the gap between conventional deep neural networks and binarized networks. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pages 35–40. ACM, 2017.

[23] Ronald G Dreslinski, Michael Wieckowski, David Blaauw, Dennis Sylvester, and Trevor Mudge. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proceedings of the IEEE*, 98(2):253–266, 2010.

[24] Siddharth Garg and Diana Marculescu. 3d-gcp: An analytical model for the impact of process variations on the critical path delay distribution of 3d ics. In *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, pages 147–155. IEEE, 2009.

[25] Dennis Gnad, Muhammad Shafique, Florian Kriebel, Semeen Rehman, Duo Sun, and Jörg Henkel. Hayat: Harnessing dark silicon and variability for aging deceleration and balancing. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 180:1–180:6, New York, NY, USA, 2015. ACM.

[26] T. Grasser, B. Kaczer, W. Goes, H. Reisinger, T. Aichinger, P. Hehenberger, P. J. Wagner, F. Schanovsky, J. Franco, M. Toledano Luque, and M. Nelhiebel. The paradigm shift in understanding the bias temperature instability: From reaction-diffusion to switching oxide traps. *IEEE Transactions on Electron Devices*, 58(11):3652–3666, Nov 2011.

[27] Xinfei Guo, Vaibhav Verma, Patricia Gonzalez-Guerrero, Sergiu Mosanu, and Mircea R Stan. Back to the future: Digital circuit design in the finfet era. *Journal of Low Power Electronics*, 13(3):338–355, 2017.

[28] Kyuseung Han, Jae-Jin Lee, Jinho Lee, Woojoo Lee, and Massoud Pedram. Tei-noc: Optimizing ultralow power nocs exploiting the temperature effect inversion. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(2):458–471, 2018.

[29] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[30] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Sebastian Herbert, Siddharth Garg, and Diana Marculescu. Exploiting process variability in voltage/frequency control. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(8):1392–1404, 2012.

[33] Sebastian Herbert and Diana Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pages 38–43. IEEE, 2007.

[34] Sebastian Herbert and Diana Marculescu. Variation-aware dynamic voltage/frequency scaling. In *HPCA*, pages 301–312. IEEE, 2009.

[35] Sebastian Herbert and Diana Marculescu. Characterizing chip-multiprocessor variability-tolerance. In *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pages 313–318. IEEE, 2008.

[36] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[37] Mark Horowitz, Thomas Indermaur, and Ricardo Gonzalez. Low-power digital design. In *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, pages 8–11. IEEE, 1994.

[38] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(5):501–513, 2006.

[39] Xuejue Huang, Wen-Chin Lee, Charles Kuo, Digh Hisamoto, Leland Chang, Jakub Kedzierski, Erik Anderson, Hideki Takeuchi, Yang-Kyu Choi, Kazuya Asano, et al. Sub-50 nm p-channel finfet. *Electron Devices, IEEE Transactions on*, 48(5):880–886, 2001.

[40] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proceedings of the 39th annual IEEE/ACM international symposium on microarchitecture*, pages 347–358. IEEE Computer Society, 2006.

[41] Shailendra Jain, Surhud Khare, Satish Yada, V Ambili, Praveen Salihundam, Shiva Ramani, Sriram Muthukumar, M Srinivasan, Arun Kumar, Shasi Kumar Gb, et al. A 280mv-to-1.2 v wide-operating-range ia-32 processor in 32nm cmos. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 66–68. IEEE, 2012.

[42] James Jeffers and James Reinders. *Intel Xeon Phi coprocessor high-performance programming*. Newnes, 2013.

[43] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[44] Da-Cheng Juan, Siddharth Garg, Jinpyo Park, and Diana Marculescu. Learning the optimal operating point for many-core systems with extended range voltage/frequency scaling. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2013 International Conference on*, pages 1–10. IEEE, 2013.

[45] Da-Cheng Juan and Diana Marculescu. Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 97–102. ACM, 2012.

[46] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[47] Ryan Gary Kim, Wonje Choi, Zhuo Chen, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. Imitation learning for dynamic vfi control in large-scale manycore systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(9):2458–2471, 2017.

[48] Ryan Gary Kim, Wonje Choi, Zhuo Chen, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. Wireless noc and dynamic vfi codesign: Energy efficiency without performance penalty. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(7):2488–2501, 2016.

[49] Sang-Yun Kim, Young Min Kim, Kwang-Ho Baek, Byung-Kil Choi, Kyoung-Rok Han, Ki-Heung Park, and Jong-Ho Lee. Temperature dependence of substrate and drain–currents in bulk finfets. *Electron Devices, IEEE Transactions on*, 54(5):1259–1264, 2007.

[50] Shin-gyu Kim, Hyeonsang Eom, Heon Y Yeom, and Sang Lyul Min. Energy-centric dvfs controlling method for multi-core platforms. *Computing*, 96(12):1163–1177, 2014.

[51] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[52] H. Kukner, P. Weckx, J. Franco, M. Toledano-Luque, M. Cho, B. Kaczer, P. Raghavan, Doyoung Jang, K. Miyaguchi, M. G. Bardon, F. Catthoor, L. Van der Perre, R. Lauwereins, and G. Groeseneken. Scaling of bti reliability in presence of time-zero variability. In *2014 IEEE International Reliability Physics Symposium*, pages CA.5.1–CA.5.7, June 2014.

[53] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[54] Chun-Yi Lee and Niraj K Jha. Cacti-finfet: An integrated delay and power modeling framework for finfet-based caches under process variations. In *Proceedings of the 48th Design Automation Conference*, pages 866–871. ACM, 2011.

[55] Chun-Yi Lee and Niraj K Jha. Fincanon: A pvt-aware integrated delay and power modeling framework for finfet-based caches and on-chip networks. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(5):1150–1163, 2014.

[56] Woojoo Lee, Kyuseung Han, Yanzhi Wang, Tiansong Cui, Shahin Nazarian, and Massoud Pedram. Tei-power: Temperature effect inversion–aware dynamic thermal management. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 22(3):51, 2017.

[57] Woojoo Lee, Yanzhi Wang, Tiansong Cui, Shahin Nazarian, and Massoud Pedram. Dynamic thermal management for finfet-based circuits exploiting the temperature effect inversion phenomenon. In *Proceedings of the 2014 international symposium on Low power electronics and design*, pages 105–110. ACM, 2014.

[58] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. The mcpat framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(1):5, 2013.

[59] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[60] Shoumik Maiti and Sudeep Pasricha. Delca: Dvfs efficient low cost multicore architecture. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pages 107–112. ACM, 2017.

[61] Diana Marculescu. Profile-driven code execution for low power dissipation. In *Low Power Electronics and Design, 2000. ISLPED'00. Proceedings of the 2000 International Symposium on*, pages 253–255. IEEE, 2000.

[62] Diana Marculescu and Puru Choudhary. Hardware based frequency/voltage control of voltage frequency island systems. In *Hardware/Software Codesign and System Synthesis, 2006. CODES+ ISSS'06. Proceedings of the 4th International Conference*, pages 34–39. IEEE, 2006.

[63] Diana Marculescu and Emil Talpes. Energy awareness and uncertainty in microarchitecture-level design. *IEEE Micro*, (5):64–76, 2005.

[64] Diana Marculescu and Emil Talpes. Variability and energy awareness: a microarchitecture-level perspective. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 11–16. IEEE, 2005.

[65] Timothy N Miller, Xiang Pan, Renji Thomas, Naser Sedaghati, and Radu Teodorescu. Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–12. IEEE, 2012.

[66] Vidyasagar Nookala, David J Lilja, and Sachin S Sapatnekar. Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis. In *Proceedings of the 2006 international symposium on Low power electronics and design*, pages 298–303. ACM, 2006.

[67] Santiago Pagani, Heba Khdr, Waqaas Munawar, Jian-Jia Chen, Muhammad Shafique, Minming Li, and Jörg Henkel. Tsp: thermal safe power: efficient power budgeting for many-core systems in dark silicon. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, page 10. ACM, 2014.

[68] Yu Pu, Xin Zhang, Jim Huang, Atsushi Muramatsu, Masahiro Nomura, Koji Hirairi, Hidehiro Takata, Taro Sakurabayashi, Shinji Miyano, Makoto Takamiya, et al. Misleading energy and performance claims in sub/near threshold digital systems. In *Proceedings of the International Conference on Computer-Aided Design*, pages 625–631. IEEE Press, 2010.

[69] Hang Qi, Evan R Sparks, and Ameet Talwalkar. Paleo: A performance model for deep neural networks. 2016.

[70] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[71] Bita Darvish Rouhani, Azalia Mirhoseini, and Farinaz Koushanfar. Delight: Adding energy dimension to deep neural networks. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 112–117. ACM, 2016.

[72] Smruti R Sarangi, Brian Greskamp, Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. Varius: A model of process variation and resulting timing errors for microarchitects. *Semiconductor Manufacturing, IEEE Transactions on*, 21(1):3–13, 2008.

[73] John Sartori and Rakesh Kumar. Three scalable approaches to improving many-core throughput for a given peak power budget. In *High Performance Computing (HiPC), 2009 International Conference on*, pages 89–98. IEEE, 2009.

[74] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[75] Donghwa Shin, Sung Woo Chung, Eui-Young Chung, and Naehyuck Chang. Energy-optimal dynamic thermal management: Computation and cooling power co-optimization. *Industrial Informatics, IEEE Transactions on*, 6(3):340–351, 2010.

[76] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[77] Saurabh Sinha, Greg Yeric, Vikas Chandra, Brian Cline, and Yu Cao. Exploring sub-20nm finfet design with predictive technology models. In *Proceedings of the 49th Annual Design Automation Conference*, pages 283–288. ACM, 2012.

[78] SC Smithsons, G Yang, WJ Gross, and WJ Gross. Neural networks designing neural networks: Multi-objective hyper-parameter optimization. *arXiv preprint arXiv:1611.02120*, 2016.

[79] S Soleimani, A Afzali-Kusha, and B Forouzandeh. Temperature dependence of propagation delay characteristic in finfet circuits. In *Microelectronics, 2008. ICM 2008. International Conference on*, pages 276–279. IEEE, 2008.

[80] Dimitrios Stamoulis, Ermao Cai, Da-Cheng Juan, and Diana Marculescu. Hyperpower: Power-and memory-constrained hyper-parameter optimization for neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*, pages 19–24. IEEE, 2018.

[81] Dimitrios Stamoulis and Diana Marculescu. Can we guarantee performance requirements under workload and process variations? In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pages 308–313. ACM, 2016.

[82] Dimitrios Stamoulis, Dimitrios Rodopoulos, Brett H Meyer, Dimitrios Soudris, Francky Catthoor, and Zeljko Zilic. Efficient reliability analysis of processor datapath using atomistic bti variability models. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 57–62. ACM, 2015.

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[84] Radu Teodorescu and Josep Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 363–374. IEEE Computer Society, 2008.

[85] M. Toledano-Luque, B. Kaczer, J. Franco, P. J. Roussel, M. Bina, T. Grasser, M. Cho, P. Weckx, and G. Groeseneken. Degradation of time dependent variability due to interface state generation. In *VLSI Technology (VLSIT), 2013 Symposium on*, pages T190–T191, June 2013.

[86] P. Weckx, B. Kaczer, M. Toledano-Luque, P. Raghavan, J. Franco, P. J. Roussel, G. Groeseneken, and F. Catthoor. Implications of bti-induced time-dependent statistics on yield estimation of digital circuits. *IEEE Transactions on Electron Devices*, 61(3):666–673, March 2014.

[87] Thomas Willhalm, Roman Dementiev, and Patrick Fay. Intel performance counter monitorl. *http://software.intel.com/en-us/articles/intel-performance-counter-monitor/*, online, accessed Jun. 2016.

[88] Jonathan A Winter, David H Albonesi, and Christine A Shoemaker. Scalable thread scheduling and global power management for heterogeneous many-core architectures. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pages 29–40. ACM, 2010.

[89] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The splash-2 programs: Characterization and methodological considerations. In *ACM SIGARCH Computer Architecture News*, volume 23, pages 24–36. ACM, 1995.

[90] Shiying Xiong and Jeffrey Bokor. Sensitivity of double-gate and finfetdevices to process variations. *Electron Devices, IEEE Transactions on*, 50(11):2255–2261, 2003.

[91] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. *arXiv preprint*, 2017.

[92] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, pages 161–170, 2015.

[93] Qian Zhang, Ting Wang, Ye Tian, Feng Yuan, and Qiang Xu. Approxann: an approximate computing framework for artificial neural network. In *2016 Design, Automation and Test in Europe Conference and Exhibition*, pages 701–706. IEEE, 2015.

[94] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[95] Yazhou Zu, Wei Huang, Indrani Paul, and Vijay Janapa Reddi. T i-states: Processor power management in the temperature inversion region. In *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pages 1–13. IEEE, 2016.

[96] Yazhou Zu, Wei Huang, Indrani Paul, and Vijay Janapa Reddi. Ti-states: Power management in active timing margin processors. *IEEE Micro*, 37(3):106–114, 2017.