3-21-2018

The Thermal-Constrained Real-Time Systems Design on Multi-Core Platforms -- An Analytical Approach

SHI SHA
Florida International University, ssha001@fiu.edu

DOI: 10.25148/etd.FIDC004089

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

THE THERMAL-CONSTRAINED REAL-TIME SYSTEMS DESIGN ON MULTI-CORE PROCESSORS – AN ANALYTICAL APPROACH

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Shi Sha

2018

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Shi Sha, and entitled The Thermal-Constrained Real-Time Systems Design on Multi-Core Processors – An Analytical Approach, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Arif Selcuk Uluagac

Nezih Pala

Raju Rangaswami

Wujie Wen

Gang Quan, Major Professor

Date of Defense: March 21, 2018

The dissertation of Shi Sha is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2018

DEDICATION

This dissertation is dedicated to my always encouraging, ever faithful parents and other family members. I also want to remember my grandparents. May you find peace and happiness in Paradise! Last but not least, I am grateful to my teachers, colleagues, friends, who assisted, advised and supported my research and efforts over the years.

ACKNOWLEDGMENTS

I wish to express my deepest appreciation to my major advisor, Dr. Gang Quan, who inspires and guides me by his encouragement, support and patience. His professional attitude and the passion to science and research have deeply impacted me.

I am grateful to all my Ph.D. committee members, Dr. Wujie Wen, Dr. Arif Selcuk Uluagac, Dr. Nezih Pala and Dr. Raju Rangaswami. Each of the members of my Dissertation Committee has provided me extensive personal and professional guidance and suggestions in improving the quality of this dissertation.

My Ph.D. research was supported in part by US National Science Foundation (NSF) grants CNS-0969013, CNS-0917021 and CNS-1018108.

ABSTRACT OF THE DISSERTATION

THE THERMAL-CONSTRAINED REAL-TIME SYSTEMS DESIGN ON MULTI-CORE PROCESSORS – AN ANALYTICAL APPROACH

by

Shi Sha

Florida International University, 2018

Miami, Florida

Professor Gang Quan, Major Professor

Over the past decades, the shrinking transistor size, benefited from the advancement of IC technology, enabled more transistors to be integrated into an IC chip, to achieve higher and higher computing performances. However, the semiconductor industry is now reaching a saturation point of Moore's Law largely due to soaring power consumption and heat dissipation, among other factors. High chip temperature not only significantly increases packing/cooling cost, degrades system performance and reliability, but also increases the energy consumption and even damages the chip permanently. Although designing 2D and even 3D multi-core processors helps to lower the power/thermal barrier for single-core architectures by exploring the thread/process level parallelism, the higher power density and longer heat removal path has made the thermal problem substantially more challenging, surpassing the heat dissipation capability of traditional cooling mechanisms such as cooling fan, heat sink, heat spread, etc., in the design of new generations of computing systems. As a result, dynamic thermal management (DTM), i.e. to control the thermal behavior by dynamically varying computing performance and workload allocation on an IC chip, has been well-recognized as an effective strategy to deal with the thermal challenges.

Different from many existing DTM heuristics that are based on simple intuitions, we seek to address the thermal problems through a rigorous analytical approach, to achieve the high predictability requirement in real-time system design. In this regard, we have made a number of important contributions. First, we develop a series of lemmas and theorems that are general enough to uncover the fundamental principles and characteristics with regard to the thermal model, peak temperature identification and peak temperature reduction, which are key to thermal-constrained real-time computer system design. Second, we develop a design-time frequency and voltage oscillating approach on multi-core platforms, which can greatly enhance the system throughput and its service capacity. Third, different from the traditional workload balancing approach, we develop a thermal-balancing approach that can substantially improve the energy efficiency and task partitioning feasibility, especially when the system utilization is high or with a tight temperature constraint. The significance of our research is that, not only can our proposed algorithms on throughput maximization and energy conservation outperform existing work significantly as demonstrated in our extensive experimental results, the theoretical results in our research are very general and can greatly benefit other thermal-related research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As a broad range of innovative applications emerge quickly, such as intelligent transportation systems, Internet of Things (IoT), artificial intelligence (AI) and beyond, the computational demands grow quickly. The explosive increments of data volume and complex workloads also urge the IC industry to create the next computing performance breakthrough in high-performance computing systems (HPC). Nowadays, real-time computing (RTC) has been widely adopted in scientific and industrial areas, e.g. real-time traffic control and medical device operations, etc. A real-time computing system with strict timing constraints in many mission-critical applications also call for guaranteed computing/service capacity, and should be robust enough to cope with unprecedented events and dynamic environments. All these drastically increased computational demands are driving computing systems to achieve a higher and higher computational capability, facing challenges from software complexity, system expansion, and hardware integration, etc.

To achieve a higher performance, the advancement of the IC technology enables more transistors to be integrated into a chip by shrinking the transistor size following the so-called "Moore's Law," i.e. the number of transistors in an IC doubles approximately every two years. Consequently, the power consumption on chip is increasing with the transistor count. Since increasing power consumption can directly translate to the raising temperature, both power and heat dissipations are becoming major obstacles in technology scaling. High temperature can degrade system performance [124], reliability [91], and even damage the chip permanently. For example, it has been reported that every $10-15°\text{C}$ temperature increment could result in a 50% reduction in the device's lifespan [140] and triple the hardware failure rate [161].

To alleviate the power/thermal barrier, multi-core processors, by taking advantage of thread or process-level parallelism, have become one of the promising solutions to achieve a better computational efficiency with a slower pace of power increment than single-core architecture. However, as the increasing number of cores continuously push the power density to a higher and higher level, the runtime thermal environment deteriorates, which becomes worse on 3D architectures [84]. The 3D IC technology stacks layers of cores vertically on top of each other to take advantage of shorter wires, higher data throughput, and larger memory bandwidth in comparison with 2D design [89]. However, the higher power density and longer heat removal path has made the thermal problem substantially more challenging than its 2D counterpart [85]. The thermal problem is a critical issue that limits the development of high-performance computing systems [57].

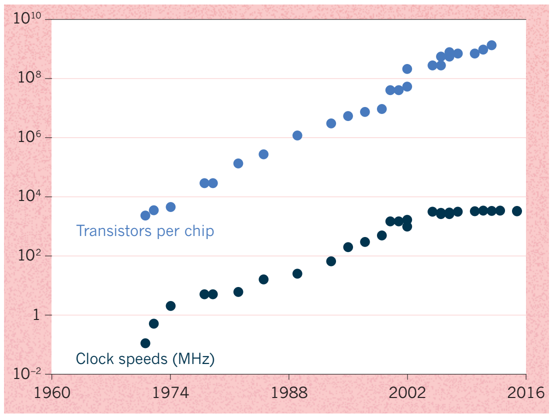
To mitigate the thermal crisis, some mechanical solutions have been explored, such as building heat sinks, heat spreaders, cooling fans or other advanced cooling mechanisms (e.g. embedded micro-channel liquid cooling on 3D processors [141] or using phase change coolant [29]). However, designing such a heat dissipation package is uneconomical if not infeasible [131], and it is unsuitable for hand-held devices [119]. More important, solely relying on the heat dissipation package cannot guarantee the temperature constraints. The violations of the temperature threshold may degrade the system throughput performance and cause real-time violations. Thus, it is not sufficient to be utilized in the real-time computing systems design. To this end, a variety of research efforts have been applied on different abstraction levels, including circuit-level, logic-level, architectural-level and the system-level.

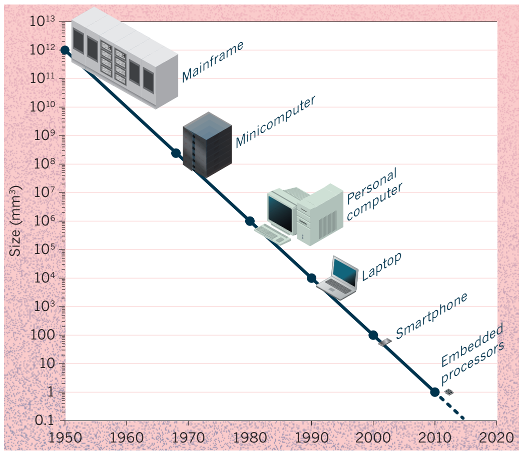
Our research employs real-time scheduling techniques on the system level. In particular, by properly controlling the computational behavior by dynamically varying computing performance and workload distribution on an IC chip, different design-optimization goals can be achieved under the thermal constraint, e.g. energy reduction, reliability enhancement and throughput maximization, etc. In what follows, we first introduce the crisis caused by soaring power and energy consumption in modern computing systems design. We then discuss the opportunities and challenges in addressing the thermal/power issue. Next, we introduce our research problem and our contributions. At last, we describe the organization of the dissertation.

1.1 The Increasing Power Consumption and Power Density of IC Chips

In the era of awaiting the ultra-low power of superconducting electronics for Quantum Computing, the pace of pursuing the next generation of high-performance devices never stops. Beyond the conventional technologies and architectures, on one hand, portable and implanted electronic devices, e.g. smartphones and user terminal of Internet of Things (IoT), call for a self-contained functionality within the scope of a small carbon area. On the other hand, high-performance computing devices, e.g. servers used in data centers, drive stationary computers to improve their performances to the next higher level of realizing supercomputing.

Both increasing the computational capability and decreasing the chip sizes, drive the semiconductor industry to keep on increasing the transistor count and transistor density. For example, cellphone application processors have increased the transistor count from 1 Billion in A5 to 2 Billion for A6 to 3 Billion for A6X. It is projected that the upcoming A9 will range from 2.7 to 4.5 Billion [31]. Although the shrinking feature sizes and FinFET technique can realize faster switching and lower the minimal power consumption of transistor operation, the chipset power and power density are still rapidly escalating to silicon limitations.





(a) (b)

Figure 1.1: (a) The "Moore's Law" doubles transistor per chip roughly every two years. The chip's clock speed also increases until 2004 when the speed scaling meets the barrier of the thermal limit. (b) As the IC power and size scaling, each generation of new electronic device emerges about every 10 years. [96]

Instead of integrating more transistors and increasing the running frequency on a monolithic single-core to pursue a higher performance, designing multi-core and many-core platforms, by exploring the thread/process level parallelism, help to lower the frequency and power consumption. The multi-core and many-core architecture approaches the "saturation point" of Moore's Law in a slower pace than single-core architecture. For example, the parallel execution scheme lowered the frequency scaling from 41% per year in 2001 to nearly 4% per year in 2011 [72], which substantially mitigated the exponential increments of power consumptions (in Figure 1.1(a)). However, the fast scaling of the processing core count and shrinking size (in Figure 1.1(b)) lead to soaring runtime temperature, which negatively impact system performance, reliability and increase the packaging and cooling cost. It is reported that near the year of 2029, the number of cores used in a data center can reach 10602, which is 30-fold of year 2015 [31]. To this end, the effective power and thermal-aware design methodologies are urgently demanded on multi-core platforms.

1.2 The Temperature Issue on Multi-Core Processors

In a multi-core regime, leveraging the system integration is widely adopted to achieve a higher performance, but building a larger SoCs/ NoCs with more processing cores results in thermal issues. For example, the emerging 3D multi-core architecture is recognized as one of the most promising solutions to achieve less delay and lower power consumptions by stacking layers of cores vertically on top of each other to take advantage of shorter wires, higher data throughput, and larger memory bandwidth in comparison with 2D design. However, the higher power density and longer heat removal path made the thermal problem substantially more challenging than a 2D design. As reported in [84], the vertical heat transfer rate of a 3D processor can be 16× that of the lateral one and the longer heat removal path in a 3D architecture may increase its core temperature by $17°C - 20°C$ compared with its 2D counterpart. As shown in Figure 1.2, the power density beyond the 100nm technology node is comparable with a nuclear reactor, and the power density of future electronics is still increasing. As multi-/many-core systems continuously grow to the level that is limited by the first advent of chip power budget or temperature limit, "dark/grey silicon" leaves a fraction of on-chip processing cores inactive. Then, the resource utilization is developed upon building an effective run-time workload mapping strategy, through a different patterning approach to seek a proper subgroup of active cores, such that thermal and power budget can be fully exploited. As reported in [37], at 22 nm technology node, 21% of a fixed-size chip must be powered off, and at 8 nm, this number grows to more than 50%. Essentially, temperature has become a first-class design constraint in modern computing systems design.

Besides the thermal crisis resulting from soaring transistor/power densities, the thermal management on multi-core processors is also challenged by non-uniformly distributed workload in both temporal and spatial dimensions. For example, on an Intel Xeon E5-2699 v3 CPU [12], the intra-die temperature difference can be up to 10°C and 24°C under balanced and unbalanced workload scenarios, respectively. The high local heat fluxes (known as “hotspot”) on multi-core platforms make the thermal management more complicated and urgent, because local hotspots may trigger the self-protection schemes and cause an unpredicted shut down of the processor. Thermal crisis has become one of the primary concerns in modern microprocessor design, because high temperature can substantially degrade system performance [124], reliability [91], and even damage the chip permanently. Every year, a tremendous amount of cooling cost has been spent in the IT industry. For example, as reported in ITRS2015 [31], the power consumption of data centers enters hundreds of Megawatts range. The global cooling power demands for the data center industry raise from 55.02 MkWh of 2017 to 482.56 MkWh around 2029, which takes averagely 55.6% total power in the data center industry in the 10-year holistic view, with the peak percentage of 72.7% at 2021.

To protect the hardware from overheating hazards, modern CPUs are featured with digital thermal sensors to monitor the temperature fluctuations. If the chip temperature exceeds the pre-defined temperature thresholds, it will trigger the automatic shut down scheme, which adversely degrades the system performances. To address the thermal crisis, some mechanical solutions have been explored, such as building heat sinks, heat spreaders, cooling fans or other advanced cooling mechanisms. For example, a 3D liquid tree-like cooling system has been proved to be favorable for minimizing the pumping power in [23]. A channel width modulation methodology has been proposed to enhance the cooling energy efficiency in [116]. The two-phase 3D liquid cooling systems has been studied in [29, 107]. However, such mechanical cooling solutions are expensive and not suitable for the mobile devices. More important, the mechanical cooling methods cannot guarantee the runtime temperature staying in a safe range.

To manage the runtime temperature and improve the thermal profile, Dynamic Thermal Management (DTM) is also developed on the system-level, which can be realized by adjusting the processing speeds using *dynamic voltage/frequency scaling* (DVFS) or turning off the unused cores using *dynamic power management* (DPM). Significant work has been done for DTM strategies, but many of them are based on simple heuristics or intuitions, such as thermal-balancing [100], "hot-and-cold" job swapping [115], allocating hot tasks to cores closer to the heat sink [84], etc. Some other works utilize reactive approaches to dynamically adjust the runtime system settings for upcoming system loads [44, 53, 46]. Although these approaches may work well for some application cases, they either lack of peak temperature guarantee or cannot ensure system performances. Thus, these methods cannot be safely utilized in the real-time systems design.

In this research, we adopt the real-time scheduling methodology and use a proactive DTM approach to guarantee the pre-defined peak temperature constraint. Meanwhile, we also aim to achieve different design optimization goals and ensure the required throughput under the peak temperature constraint at the same time. Furthermore, our rigorous analytical approach intends to have a better understanding of the interplay among different design factors/constraints, which helps to develop more effective thermal management policies.

1.3 Research Problems and Our Contributions

Due to both economic and physical challenges in IC design, power and thermal issues on multi-core platforms call for effective and cost-efficient solutions in developing next-generation computing systems. In this dissertation, we study the real-time computing system design with power/thermal-awareness. In particular, our research aims to develop a variety of design optimization algorithms (e.g. throughput maximization, energy reduction, peak temperature minimization, etc.) based on the state-of-the-art computer architecture. The research incorporates system-level DTM techniques and takes leakage-temperature dependency and multi-core thermal interference into account. Different from many existing DTM heuristics that are based on simple intuitions, we seek to address the thermal-related optimization problems through a rigorous analytical approach, which intends to understand the fundamental thermal/power-aware design principles. The significance of our research is that, our design emphasizes the guaranteed throughput performance and response time in developing different optimization strategies, which can be safely employed in the real-time system design. Meanwhile, not only our proposed algorithms can outperform existing work significantly as demonstrated in the extensive experimental results, but also the theoretical results in our research are very general and can greatly benefit other thermal-related research.

The contributions of this dissertation are summarized as follows:

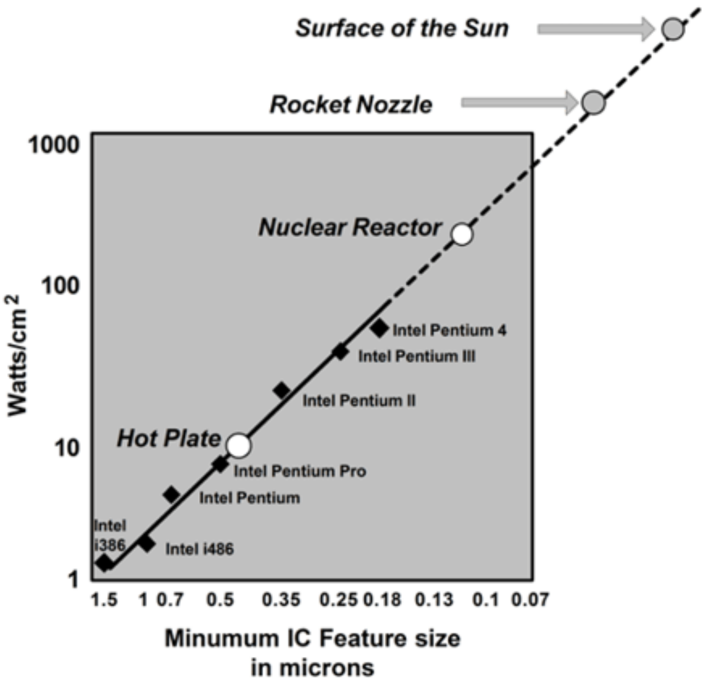
1. We analytically prove a series of fundamental principles in the forms of theorems and lemmas for thermal modeling, peak temperature identification and peak temperature reduction, which are key to thermal-constrained computer system design, that based on the well-known multi-core RC-thermal model, which accounts for the temperature-leakage dependency and multi-core heat transfer. These principles are general enough to be applied on 2D and 3D multi-core platforms, and form the theoretical basis for a more rigorous analytical study, which can be used for other thermal-related problems.

2. Based on the thermal characteristics on multi-core platforms, we analytically study the throughput maximization problem under the peak temperature constraints. To take advantage of thermal heterogeneity of different cores for performance improvement, we propose to run each core with multiple speed levels and develop a schedule based on two novel concepts, i.e. the step-up schedule and the m-Oscillating schedule, for multi-core platforms. The proposed methodology can ensure the peak temperature guarantee with a significant improvement in computing throughput, up to 89% with an average improvement of 11%. Meanwhile, the computational time reduces orders of magnitude compared to the traditional exhaustive search-based approach.

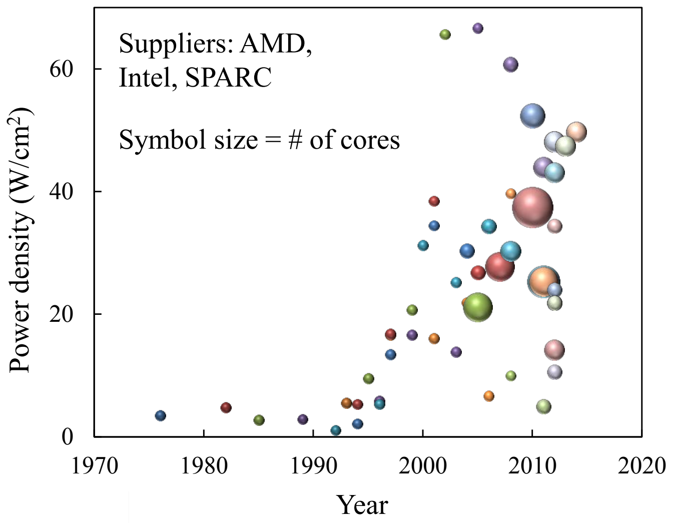
3. Although energy minimization is closely related to temperature reduction, the most energy efficient method may not be the most effective one to meet the temperature constraints, and vice versa. We then study the problem of how to partition periodic hard real-time tasks on a multi-core platform to maximize the overall energy efficiency under a peak temperature constraint. Different from the traditional load-balancing approach, we use a thermal-balancing approach to improve the overall system energy efficiency, especially when the temperature constraints are tight. We further identify the lower bound for energy consumption by this approach, and then transform the task partitioning problem to a variable sized bin packing problem. We further use an enhanced algorithm to optimize the task partitioning results. Our simulation results show that the proposed thermal-balancing approach can greatly improve the energy efficiency and task partitioning feasibility for real-time systems with high system utilizations and tight temperature constraints.

1.4 Structure of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we introduce the pertinent background to this dissertation and discuss existing works that are closely related to our research. In Chapter 3, we formally prove a series of thermal properties on multi-core platforms and study a temperature bounding method, which overturns the traditional peak temperature identification methods that use worst-case execution time to compute peak temperature directly from a given schedule. In Chapter 4, we focus on the throughput maximization problem of multi-core platforms and study a frequency oscillating methodology to fully use the "headroom" of the temperature threshold and enhance the throughput performance and service capability. In Chapter 5, we investigate how to partition periodic hard real-time tasks on a multi-core platform to maximize the overall energy efficiency under a peak temperature constraint. Finally, in Chapter 6, we conclude this dissertation and discuss possible future works.



(a)



(b)

Figure 1.2: (a) The power density increases exponentially with the IC feature size and its is comparable with a nuclear reaction. [Source: Intel Corp.] (b) Emerging new electronic devices results in power density increases chronologically [21]

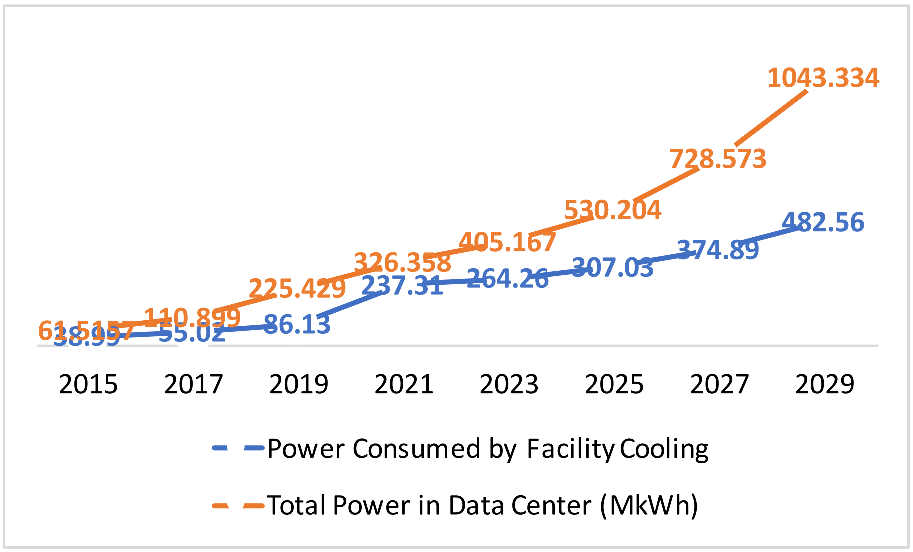


Figure 1.3: A global view of cooling power v.s. total power in the data center industry [31]

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter covers the background of this research. We first introduce several important concepts on power/energy consumption commonly used in IC design. Then, we introduce several thermal management techniques at different design stages. We further conduct a more specific survey on power, thermal and energy-aware scheduling techniques, which are closely related to our research topic.

2.1 Real-Time Systems

Real-time systems are widely used in the computing systems design, e.g. multimedia systems, embedded automotive electronics, etc.. In a real-time system, the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical instant at which these results are produced [35]. The real-time systems adjust the operating state as a function of physical time. The instant that the result is required to be delivered is called deadline.

Real-time systems can be largely categorized into *hard real-time* and *soft real-time* systems. Hard real-time's response time requirement is firm and the violation of such type of deadline can result in a catastrophe. In contrast, a *soft real-time system*, e.g. multimedia or online reserving systems, is used in non-critical situations that the deadline is met at the best effort. Missing deadline only degrades the quality of service (QoS).

Further, hard real-time scheduling can be categorized into two types: *static* and *dynamic*. The static scheduling makes a decision at the compile time according to the tasks' parameters, e.g. execution time, precedence, deadlines, etc. Since the schedule is generated off-line, static scheduling may tolerate a higher computational cost. In contrast, the dynamic scheduling makes the decision at the runtime. Although it is more flexible and adaptive to different workload scenarios, creating a runtime schedule steals the computational resources and may cause a large overhead.

There are several uniprocessor real-time scheduling policies. For example, *earliest deadline first* (EDF) policy always assign the highest priority to the task with the nearest current deadline. In addition, the EDF algorithm can achieve 100% utilization [87]. *Rate monotonic* (RM) assign the highest execution priority to the task with the shortest period. RM has been proved that a feasible schedule can always be found under the utilizations of $ln2$ (69.3%) [48].

On multi-core platform, the traditional scheduling policies for uniprocessors, e.g. EDF and RM, may not always lead to the best scheduling results. As the design space becomes larger, judiciously considering the utilization trade-offs that exists in multi-core systems are necessary. For example, according to the degree of allowed task migration, *global schedule* store all the ready tasks in a single priority-ordered queue and execute the highest priority one in sequence. In contrast, the *partitioning approach* allows each task only assign to one dedicated core, and join the task ready queue specified on this core. The research shows that some “middle approach” of a combination of global and partitioning methods may be a better choice on multi-core platforms [17].

Real-time has also been been applied in other system settings and architectures. For example, in a distributed real-time systems (DRTS), improving the schedulability by assigning local optimal end-to-end deadline has been explored in [59]. For discrete-event systems, the real-time calculus studied the temporal properties on queuing theory to provide service for incoming task requests [134].

2.2 Power Consumption

In this section, we first introduce the sources of power consumption in digital integrated circuits. Then, we discuss a number of existing power reduction techniques at the system level. The total power consumption in CMOS digital ICs consists of *dynamic power* and *static power*.

$$P = P_{dyn} + P_{leak}, \tag{2.1}$$

2.2.1 Dynamic Power Reduction

The dynamic power closely relates to the switching activities of the transistor, which is a quadratic function of supply voltage and proportional to the frequency.

$$P_{dyn} = Cv^2 f, \tag{2.2}$$

where C is the equivalent parasitic capacitance. Term v and f are the supply voltage and clock frequency (execution speed), respectively. Modern processors are usually featured with several discrete running modes and for each mode $v \propto f$. Thus, dynamic power can be simplified as

$$P_{dyn} = \gamma(v) \cdot v^3, \tag{2.3}$$

where γ is a constant for different running modes.

There are a number of system-level techniques and multi-core designs used to minimize the active power consumption [15]. For example,

Clock Gating is selectively shutting off the clock for a circuit to prevent any toggle activity of the clocks or registers to reduce the power dissipation.

Dynamic Voltage and Frequency Scaling (DVFS) is to exploit the opportunity to scale down the voltage and frequency for power saving, when performance requirements can be satisfied in a low loading condition.

Voltage Island is to realize "Multi-Supply Voltage" (MSV) techniques, that can reduce power consumption of SoCs, when not requiring all blocks to operate at maximum speeds at all times.

On-Die Voltage Regulator provides a faster response than off-chip modules for adjusting the voltage and current supply in different active states.

3D-IC utilizes Through Silicon Via (TSV) to connect several layers of processors and/or memories over a silicon interposer, which provide a low capacitance signal interconnect between die, thus reducing the I/O active power.

Since total power is a combination of dynamic power and leakage power. In what follows, we introduce the fundamentals for leakage power as well as its optimization strategies.

2.2.2 Leakage Power Reduction

The leakage power, also called *static power* consumption, is caused by a small amount of current flow from power to the ground. The leakage power can be formulated as [86]

$$P_{leak} = N_{gate} \cdot I_{leak} \cdot v_{dd}, \tag{2.4}$$

where N_{gate} represents the number of gates, v_{dd} is the voltage level, and I_{leak} is the leakage current. I_{leak} varies with both temperature and supply voltage and can be calculated by a circuit-level non-linear and high-order equation. Since leakage current depends on both supply voltage and temperature [10], for system-level analysis with a tolerable complexity, leakage power on the system-level can be approximated as

$$P_{leak} = \alpha(v) + \beta T(t), \tag{2.5}$$

where α is a constant for different running modes. $T(t)$ is the temperature at time t. β is a constant. Further, the leakage and temperature correlation can be captured by a piece-wise linear function [64], with an average of 0.3% derivation from the circuit level formulation [153, 86]. Some existing static power saving strategies are listed as below [15].

Power Gating saves leakage power by shutting off the current to the blocks of the circuit that are standby. However, power gating needs to be applied with caution, because it causes more time delay than *clock gating*.

Multi-Threshold CMOS reduces leakage power by swapping of nominal threshold voltage gates with higher threshold voltage gates. In CMOS the subthreshold leakage is inversely proportional to the threshold voltage. Careful trade-off analysis needs to be done to achieve optimal leakage savings and mitigate delay effects.

Active Back-Bias is an approach that increases the bias voltage of the substrate nodes in CMOS gates to reduce the leakage current. This biasing technique essentially increases the threshold voltage of a unit or the entire chip during standby modes, hence decreasing the leakage power.

Some other techniques are exploit for power saving purpose. For example, as the feature size continues to shrink in each technology node, the voltage scaling approaches a threshold that dynamic and leakage power has a trade-off around the threshold voltage (V_t). The optimum operating point is usually slightly above V_t and is called the near-threshold operating point.

2.3 Thermal Management

In previous section, we introduced that the total power in digital ICs is a combination of dynamic and leakage power consumption with a brief introduction of their optimization methodologies. Since high power consumption leads to the high temperature directly, thermal problem becomes one of the first class constraints in computing systems design. In this section, we first show thermal management is an indispensable part in computing systems design, followed by thermal/power-related works.

2.3.1 The Need for Thermal Management

Temperature, a long-lasting concern, is rooted in every stage of IC design and penetrates to every corner of human lives. For large-scale computing infrastructure, e.g. data centers and servers, the advancement of thermal management saves a tremendous cooling cost globally each year, and it is also an effective way to reduce the environmental impact for green computing purpose. For stationary computers, e.g. desktop or laptop computers, the application driven factors, e.g. internet surfing, video streaming and gaming, encourage IC industry to develop more aggressive thermal control methodologies to meet the application market needs as well as user satisfactory. For portable devices, e.g. mobile phones, implanted electronics and user terminals of IoT, the ultra-low-power design requires effective thermal control strategies either because a thermal-sensitive environment, or due to power/energy concern. For example, every 1°C of temperature increment of implanted devices may cause permanent tissue change. The mobile phone and tablets also need to consider the heat dissipation coming from the battery discharge along with the heat generated from mobile computing itself; meanwhile, these portable devices need to save the thermal-related leakage power to maintain the battery mission cycle.

However, the soaring power density along with the uncertainty of the workload, the temporal and spacial non-uniformity of power distribution and the large variation of power dissipations among different applications challenge the thermal controllability in modern computing systems design at the same time. In all perspectives, the research to rethink and explore different ways to improve the effectiveness of the system resources with thermal awareness is indispensable in each design stage.

2.3.2 Related Works on Power and Thermal Management

There have been extensive research efforts for thermal related optimizations on multi-core platforms, including throughput maximization (e.g. [145, 124, 40, 101]), power/energy reduction (e.g. [106, 157, 117]), peak temperature reduction (e.g. [84, 115, 156, 42]) and reliability enhancement (e.g. [138]), etc. Essentially, these works aim at optimizing the resource usage in design of high performance, low power/energy and highly reliable computing systems with chip temperature either as an optimization goal or a design constraint. Based on their approaches, the existing work can be largely categorized into the following three categories.

First, many existing researches are based on simple heuristic or intuitions. For example, for peak temperature minimization purpose, interleaving the hot/cool tasks in 3D platforms temporally and spatially is proposed in [84], properly assigning slacks to split hot tasks is proposed in [156] and assigning hot tasks to cool cores is proposed in [12]. However, in these approaches, to determine the accurate and strongly justifiable metrics to classify hot/cool tasks/cores can be difficult. In addition, without solid analytical analyses, to make other design tradeoffs in the meantime, such as task migration overhead v.s. scheduling interval length can be challenging. Although these heuristic/intuition methods may work in some application scenarios, it becomes extremely difficult, if not impossible at all, to guarantee the system performance and design constraints such as timing and peak temperature.

Second, some other approaches resort to traditional control techniques or optimization methods, such as machine learning, mathematical programming, or meta-heuristic searching methods, to deal with thermal issues. For example, using feedback control technique on multi-core platform, Fu et al. [44] proposed a framework that enforce the desired temperature and CPU utilization bounds of embedded real-time systems through DVFS. Hanumaiah et al. [53] developed a closed-loop controller to predict the desired voltage/frequency settings to achieve maximum energy efficiency without violating the thermal limitations. Xie et al. [147] developed a look-up table based DTM method on a thermal coupled processor/battery model, which considered the space limitation of mobile devices. Machine learning is also explored to learn and make predictions on temperature variations. For example, Ge et al. [46] proposed a machine learning technique to capture the correlation between temperature change and workload switching pattern, and, thus, choose the proper management policy considering performance-temperature tradeoff during runtime. These approaches help to uncover deeper rationales in temperature management better than simple intuitions. However, it is still difficult to employ these approaches to ensure strong guarantee to the temperature and other design constraints.

To this end, mathematical programming methods are also adopted to optimize resource allocation under temperature and other design constraints. For example, Wang et al. [145] proposed an integer linear programming-based approach (ILP) for throughput maximization on a temperature-constrained multi-core platform. Murali et al. [101] used a convex optimization method by a two-phase iterative approach to approximate the solution. When considering discrete processor speed levels, Hanumaiah et al. [56] formulated the task allocation and processor DVFS setting problem as a convex optimization problem to minimize the task completion time. Chantem et al. [19] proposed an optimal ILP method for thermal-aware task assignment and scheduling problem to minimize the peak temperature under a given workload. Singh et al. [129] used ILP methodology for an application-driven approach that considered the communication overhead of video streaming to minimize the peak temperature and energy contemporarily. These approaches based on mathematical programming usually can identify the optimal solution for the given problem and can guarantee that all constraints are satisfied. There are two major drawbacks of these approaches: (i) The solution itself, if it can be obtained, does not provide deep insight to the reasonings and rationales of the problems; (ii) The computational cost increases too fast and can be prohibitive as the system scale becomes larger.

To deal with the computational cost problem, many approaches used meta-heuristic searching algorithms. For example, using genetic programming approach, Saha et al. [117] proposed to minimize the energy for periodic tasks under a peak temperature constraint on heterogeneous systems. Fan et al. [40] proposed a meta-heuristic approach to boost system performance in a small interval by supplying additional power to the system without exceeding the temperature and power supply limit. For these approaches, to maintain a high quality of the result with a manageable computational cost can be a challenging issue. Also, it is difficult to employ these approaches to unveil the cause-and-effect relations within a complex system.

The third type of approaches (e.g. [42, 106, 124, 147, 130]) intend to ensure strong guarantee to thermal constraints based on formal and analytical thermal analysis, to uncover underlying correlations among different design parameters quantitatively and not qualitatively. This is particularly useful in design of real-time systems, where predictability is critical and complicated resource management policies (such as priority, preemption, resource sharing, etc) cannot be easily formulated in mathematical programming. For example, Fisher et al. [42] formulated a series of schedulability and feasibility conditions for an online thermal-aware global scheduling algorithm for sporadic task sets on homogeneous multi-core platforms. Pagani et al. [106] proposed a new multi-core power budget index, so called thermal safe power (TSP), which can guarantee peak temperature constraints and result in a safer and higher throughput capacity than traditional thermal design power (TDP). Sha et al. [124] proved a series of theorems for peak temperature identification, speed selection, oscillating frequency principles, and based on which, they presented a frequency oscillating method to maximize the throughput with a guaranteed peak temperature on a multi-core platform. To check the thermal-aware feasibility, Ahmed et al. [4] derived a series of necessary and sufficient conditions on a temperature-constrained platform, which considered the performance/temperature trade-off based on different topologies. Assisted with rigorous mathematical analysis, these approaches usually can achieve the goal of strong thermal guarantee without suffering from prohibitive computational cost in mathematical programming approach. Also they help to uncover fundamental principles for more efficient and effective thermal-aware design, which would be otherwise unavailable.

2.4 Summary

In this section, we present the essential pertinent of our research and review some closely related works in the literature. We first introduce the basic concepts and different source of power consumption. Existing power reduction techniques are discussed. Then, we present the need for thermal management on multi-core platforms with an extensive literature review of current technologies. Based on the above discussions, we can see that thermal-aware scheduling under a variety of constraints still poses a tremendous challenge for both academia and industry. Studying the interplay of different design constraints in a comprehensive and systematic way is becoming more and more critical.

In this dissertation, the goal of our research is to develop effective and efficient scheduling methods on multi-core platform to provide deterministic guarantees of thermal constraints under different design objectives, e.g. energy reduction, peak temperature reduction and throughput maximization, etc. In the following chapters, i.e. Chapter 3, 4 and 5, we present our contributions on this subject. We then conclude this dissertation in Chapter 6.

CHAPTER 3

FUNDAMENTALS ON MULTI-CORE THERMAL-AWARE REAL-TIME SCHEDULING

To study the power/thermal management on multi-core platform, the first priority is to build a better understanding of the thermal models, which helps to develop more effective thermal management policies. However, when considering the interdependency between the leakage power and temperature and core-to-core heat transfer, the thermal analysis on multi-core platform becomes substantially complex. To facilitate rigorous analytical thermal analysis, it is our intention to develop some general and provable principles/fundamentals on characteristics of heat dissipation for ease of formal verification and analysis in real-time system design.

The rest of this chapter is organized as follows. Section 3.1 introduces the preliminaries on system model and thermal model used in this research. Section 3.2 utilizes a series of provable lemmas and theorems to unveil the characteristics of the well-known RC-thermal model. Section 3.3 shows that directly using tasks' worst-case execution time to capture the peak temperature can be misleading. To this end, we introduce the new concept of worst-case execution time-based "step-up schedule" and show it can bound the peak temperature not only for an arbitrary real-time schedule with given worst-case execution time, but it is also effective when the schedule's actual execution time varies. Section 3.4 shows the experimental results and Section 3.5 concludes this chapter.

3.1 Preliminaries

We present the models for our multi-core systems. The **bold characters** represent the vectors and matrices and non-bold characters are used for ordinary variables and coefficients. All the matrices/vectors/values are in the real number domain. The notations in Table 3.1 are used in the dissertation.

Table 3.1: Summary of Notations

Symbol	**Meaning**
$\mathbb{S}(t)$	A periodic multi-core schedule;
$\mathbb{I}_q$	The q_{th} state interval in $\mathbb{S}(t)$ with time interval $[t_{q-1}, t_q]$;
l_q	The interval length of $\mathbb{I}_q$, i.e. $l_q = t_q - t_{q-1}$;
$\mathbf{T_0}$	The starting temperatures;
$\mathbf{T}_{ss}(t)$	The stable status temperatures at time t;
$\mathbf{1}_{N\times 1}$	An $(N \times 1)$ matrix with all elements being 1;
$\mathbf{0}_{N\times 1}$	An $(N \times 1)$ matrix with all elements being 0;
$max(\mathbf{X})$	Find the maximum scalar value from matrix/vector $\mathbf{X}$;
Given two matrices $\mathbf{X}$ and $\mathbf{Y}$ with the same dimensions (e.g. $N_1 \times N_2$), operators $>$, $<$, $\geq$ and $\leq$ are defined as element-wise scalar comparisons. For example, $\mathbf{X} \leq \mathbf{Y}$ means that $X_{i,j} \leq Y_{i,j}$, $\forall i \in [1, N_1]$ and $\forall j \in [1, N_2]$.	

3.1.1 System Model

We consider a multi-core platform $\mathfrak{N}$ contains N_c number of cores, $\mathfrak{N} = \{core_\kappa : \kappa = 1, \cdots, N_c\}$. Each core is DVFS-independent. Also, each core has different running modes and each running mode is characterized by a pair of parameters (v, f), where v is the supply voltage and f is the working frequency ($v \propto f$). For an inactive core, we assume $v = f = 0$. In this paper, for ease of presentation, we use supply voltage v to denote the processing speed (amount of work performed within a unit time) when there is no confusion.

As different cores may execute in different running modes at different times, a multi-core platform can be regarded as running on a sequence of scheduling intervals, in each of which each core runs only in a unique mode. We call such an interval, e.g. $[t_{q-1}, t_q]$, as a **state interval**.

Consider a multi-core periodic schedule $\mathbb{S}(t) = \{\mathbb{I}_1, \cdots, \mathbb{I}_z\}$, where $\mathbb{I}_q = [t_{q-1}, t_q]$, the performance of the multi-core platform can be represented by the average completed workload on each core divided by the length of one hyper-period. The performance (THR) is

$$THR = \frac{\sum_{q=1}^{z} THR_q}{N \sum_{q=1}^{z} l_q} = \frac{\sum_{q=1}^{z} \sum_{i=1}^{N} f_{i,q} \cdot l_q}{N \sum_{q=1}^{z} l_q}, \tag{3.1}$$

where $f_{i,q}$ is the running frequency of the i th core within the q th state interval. l_q is the length of the q th state interval.

3.1.2 Thermal Model

The thermal model, similar to that in [138, 144, 52], is built upon the duality between heat transfer and electrical phenomena as an RC-lumped circuit. Specifically, the RC-model consists of three vertical, conductive layers for the die, heat spreader, and heat sink, and a fourth vertical, convective layer for the sink-to-air interface. Heat generated from the active silicon device layer is conducted through the silicon die to the thermal interface material, heat spreader and heat sink, then convectively removed to the ambient air [131].

The thermal nodes on die layers are *active* nodes, which represent the processing cores with non-zero power consumptions. In contrast, thermal nodes on other layers are called *inactive* nodes, since they do not consume power. Assume the thermal nodes in the system are $\mathbf{\Pi} = \{\Pi_i, i = 1, \cdots, N\}$, in which the first N_c elements represent the active nodes. Let $\Pi_i \in \mathbf{\Pi_{HSK}}$ if the thermal nodes lay on the heat sink layer and R_{conv} represents the thermal resistance from the heat sink to ambient air.

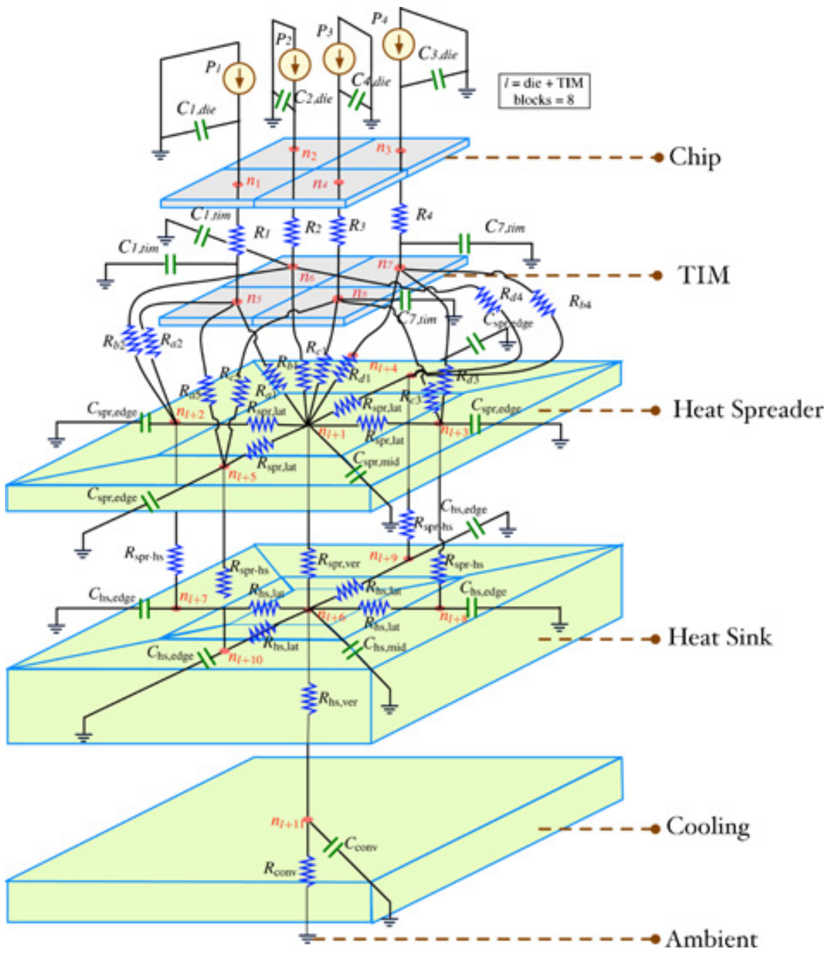


Figure 3.1: A HotSpot Thermal Model for a 4-core platform [55]. (Our model adds lateral thermal resistors on the chip level for core-to-core heat transfer.)

The thermal behavior of a multi-core platform within a state interval can be formulated as

$$\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\mathbf{T}(t) + \mathbf{B}(\mathbf{v}), \tag{3.2}$$

where $\mathbf{T}(t)$ vector represents node temperatures at time t. Coefficient matrix $\mathbf{A} = [A_{i,j}]_{N\times N}$ is an architectural-related constant, thus the system is time invariant. $\mathbf{A}$ depends only on the thermal capacitance matrix $\mathbf{C} = diag\{C_1, \cdots, C_N\}$ and thermal resistance matrix $\mathbf{G} = [G_{i,j}]_{N\times N}$ as $\mathbf{A} = -\mathbf{C}^{-1}\mathbf{G}$, where C_i is the thermal capacitance of the i-th thermal node, $C_i > 0$ and

$$G_{i,j} = \begin{cases} \sum_{\theta \neq i} \frac{1}{R_{i,\theta}} + \xi_i \frac{1}{R_{conv}}, & \text{if } i = j, \\ -\frac{1}{R_{i,j}}, & \text{otherwise,} \end{cases} \tag{3.3}$$

in which $\xi_i = 1$ when $\Pi_i \in \mathbf{\Pi_{HSK}}$; otherwise, $\xi_i = 0$. $R_{i,i}$ (or $R_{i,j}$) denotes the thermal resistance of the i-th thermal node to itself (or the j-th thermal node). The upper left $N_c \times N_c$ sub-matrix of $\mathbf{A}$ and $\mathbf{G}$ contribute to the cores. Existing studies show that matrix $\mathbf{G}$ has following properties:

Property 3.1.1. *Matrix* $\mathbf{G}$ *has following properties:*

1. $\mathbf{G}$ *is a quasi-positive matrix with all of its entries being non-negative except for those on the main diagonal [52];*
2. $\mathbf{G}$ *is strictly diagonal dominant, real symmetric and nonsingular (Lemma 1 in [144]);*

Both $\mathbf{C}$ and $\mathbf{G}$ are $N \times N$ square matrices. Since $\mathbf{C}$ only contains non-zero elements on the diagonal, it is invertible. Moreover, $\mathbf{G}$ is also invertible, because it is *nonsingular*. Then, since $\mathbf{A} \cdot \mathbf{A}^{-1} = -\mathbf{C}^{-1}\mathbf{G} \cdot (-\mathbf{C}^{-1}\mathbf{G})^{-1} = \mathbf{C}^{-1}\mathbf{G}\mathbf{G}^{-1}\mathbf{C} = \mathbf{I}$, $\mathbf{A}$ is invertible. $\mathbf{A}$ is neither symmetric nor diagonal dominant.

Coefficient vector $\mathbf{B} = [B_i]_{N\times 1}$, a power-related vector, depends on not only the thermal capacitances of the multi-core platform but also the running mode of each core. Assume $\forall \mathbf{v}_1 \geq \mathbf{v}_2$ leads to $\mathbf{B}(\mathbf{v}_1) \geq \mathbf{B}(\mathbf{v}_2)$.

When running a multi-core processor under a constant supply voltage $\mathbf{v}$ long enough (i.e. $t \to \infty$), it will eventually reach a constant temperature $\mathbf{T}^{\infty}(\mathbf{v}) = -\mathbf{A}^{-1}\mathbf{B}(\mathbf{v})$ as $d\mathbf{T}(\infty)/dt = \mathbf{0}$. For schedules that consist of multiple state intervals, the state intervals may not be long enough for the temperature to be constant. As shown in [52], the transient temperature at time t within a state interval (e.g. the

q-th interval $[t_{q-1}, t_q]$) can be formulated as

$$\mathbf{T}(t) = e^{\mathbf{A}(t-t_{q-1})}\mathbf{T}(t_{q-1}) + (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})\mathbf{T}_q^{\infty}, \tag{3.4}$$

where $t_{q-1} \leq t \leq t_q$ and $\mathbf{T}(t_{q-1})$ is the temperature vectors at the beginning of the q-th interval. $\mathbf{T}_q^{\infty}$ is the constant temperature when running processor using supply voltage $\mathbf{v}_q$ long enough and $\mathbf{I}$ is an identity matrix.

For a periodic schedule $\mathbb{S}(t)$ with z state intervals and period t_p, let t_{q-1} and t_q be the starting time and ending time of the q-th state interval, respectively. Let $l_q = t_q - t_{q-1}$, and from (3.4), we can derive the temperature at t_p through the temperature at each consecutive scheduling point in the first period as

$$\begin{aligned}
\mathbf{T}(t_1) &= e^{\mathbf{A}l_1}\mathbf{T_0} + (\mathbf{I} - e^{\mathbf{A}l_1})\mathbf{T}_1^{\infty} = (\mathbf{I} - e^{\mathbf{A}l_1})\mathbf{T}_1^{\infty} + e^{\mathbf{A}l_1}\mathbf{T_0};\\
\mathbf{T}(t_2) &= e^{\mathbf{A}l_2}\mathbf{T}(t_1) + (\mathbf{I} - e^{\mathbf{A}l_2})\mathbf{T}_2^{\infty}\\
&= \sum_{q=1}^{2} e^{\mathbf{A}\sum_{\theta=q+1}^{2} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty} + e^{\mathbf{A}\sum_{\theta=1}^{2} l_\theta}\mathbf{T_0};\\
&\dots\\
\mathbf{T}(t_h) &= e^{\mathbf{A}l_h}\mathbf{T}(t_{h-1}) + (\mathbf{I} - e^{\mathbf{A}l_h})\mathbf{T}_h^{\infty}\\
&= \sum_{q=1}^{h} e^{\mathbf{A}\sum_{\theta=q+1}^{h} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty} + e^{\mathbf{A}\sum_{\theta=1}^{h} l_\theta}\mathbf{T_0};\\
&\dots\\
\mathbf{T}(t_p) &= \sum_{q=1}^{z} e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty} + e^{\mathbf{A}\sum_{\theta=1}^{z} l_\theta}\mathbf{T_0}.
\end{aligned} \tag{3.5}$$

When repeating a periodic schedule with multiple state intervals long enough, the temperature eventually enters the *thermal stable status*, in which the temperature trace exhibits a repeat pattern. Specifically, for a periodic schedule $\mathbb{S}(t)$ with z state intervals and period t_p, let t_{q-1} and t_q be the starting time and ending time of the q-th state interval, respectively. The transient temperature in the stable status can be formulated as [52]

$$\mathbf{T}_{ss}(t_q) = \mathbf{T}(t_q) + \mathbf{K}_q(\mathbf{I} - \mathbf{K})^{-1}(\mathbf{T}(t_p) - \mathbf{T}(0)), \tag{3.6}$$

in which $\mathbf{T}(t_q)$ and $\mathbf{T}_{ss}(t_q)$ are the temperature at time t_q in the first period and in the *thermal stable status*, respectively. $\mathbf{T}(0)$ is the starting temperature for the first period and equals to $\mathbf{T_0}$. The θ-th state interval size $l_\theta = t_\theta - t_{\theta-1}$, $\mathbf{K}_q = e^{\mathbf{A}\sum_{\theta=1}^{q} l_\theta}$ and $\mathbf{K} = e^{\mathbf{A}\sum_{\theta=1}^{z} l_\theta} = e^{\mathbf{A}t_p}$.

3.2 The Properties of the Thermal Model

In this section, we focus on some inherent properties related to the multi-core RC thermal model itself. The thermal model in (3.2) is a *linear time-invariant* (LTI) system, which captures the thermal dynamics by N first-order differential equations involving N state variables. The system matrix $\mathbf{A}$ plays an important role in temperature dynamics before and when a multi-core platform reaches its temperature stable status, because $\mathbf{A}$ relates how the current temperature affects the temperature change $d\mathbf{T}(t)/dt$ [52] and $\mathbf{T}^{\infty}$. Moreover, the property of $\mathbf{A}$ determines the system stability [13], and its transformations, such as $-\mathbf{A}^{-1}$, $e^{\mathbf{A}l}$ or $(\mathbf{I} - e^{\mathbf{A}l})^{-1}$ etc, are closely related to other properties of a system. In this section, we first present some properties related to matrix $\mathbf{A}$.

Lemma 3.2.1. *Matrix* $\mathbf{A}$ *has all negative real eigenvalues.*[1]

Proof. Since $\mathbf{C} = diag\{C_1, \cdots, C_N\}$ and $C_i > 0$, we have $\mathbf{C}^{1/2} = diag\{\sqrt{C_1}, \cdots, \sqrt{C_N}\}$ and $\mathbf{C}^{-1/2} = diag\{1/\sqrt{C_1}, \cdots, 1/\sqrt{C_N}\}$ and they are nonsingular. The transpose of $\mathbf{C}^{1/2}$ and $\mathbf{C}^{-1/2}$ equal to themselves, respectively.

[1]Similar conclusion was mentioned in [13].

$\mathbf{G}$ is positive definite, because a symmetric diagonally dominant matrix with real non-negative diagonal entries is positive definite [92]. Thus, there exists a $\mathbf{Y} \neq \mathbf{0}$ such that $\mathbf{Y}^T\mathbf{G}\mathbf{Y} > \mathbf{0}$. Let $\mathbf{Y} = \mathbf{C}^{-1/2}\mathbf{X}$, $\mathbf{X} \neq \mathbf{0}$ and $\mathbf{X}^T$ denotes the transpose of $\mathbf{X}$. Then, we have $\mathbf{Y}^T\mathbf{G}\mathbf{Y} = (\mathbf{C}^{-1/2}\mathbf{X})^T\mathbf{G}\mathbf{C}^{-1/2}\mathbf{X} = \mathbf{X}^T\mathbf{C}^{-1/2}\mathbf{G}\mathbf{C}^{-1/2}\mathbf{X} = \mathbf{X}^T(\mathbf{C}^{-1/2}\mathbf{G}\mathbf{C}^{-1/2})\mathbf{X} = \mathbf{X}^T\mathbf{\Omega}\mathbf{X} > \mathbf{0}$, where $\mathbf{\Omega} = \mathbf{C}^{-1/2}\mathbf{G}\mathbf{C}^{-1/2}$ and it is symmetric. Thus, $\mathbf{\Omega}$ is positive definite, and its eigenvalue must be positive real numbers (Theorem 7.2.2 and Theorem 7.3.2 in [7]).

Since there exists a nonsingular matrix $\mathbf{C}^{1/2}$ such that the *similarity transformation* (page 506 in [95]) of $-\mathbf{A} = \mathbf{C}^{-1}\mathbf{G} = \mathbf{C}^{-1/2}\mathbf{C}^{-1/2}\mathbf{G} = \mathbf{C}^{-1/2}(\mathbf{C}^{-1/2}\mathbf{G}\mathbf{C}^{-1/2})\mathbf{C}^{1/2} = (\mathbf{C}^{1/2})^{-1}\mathbf{\Omega}\mathbf{C}^{1/2}$, $-\mathbf{A}$ is similar to $\mathbf{\Omega}$ and sharing all the eigenvalues (page 508 in [95]). Thus, all the eigenvalues of $\mathbf{A}$ are negative real numbers. $\square$

In control theory, since all the eigenvalues of $\mathbf{A}$ are strictly negative real values, it is *asymptotically stable* [16]. Moreover, all the *asymptotically stable* systems are also bounded-input, bounded-output (BIBO) stable, which means the output will be bounded for every input to the system that is bounded. In other words, there always exists a peak temperature for any schedule executed on a given platform, with its power supply stay below the maximal threshold.

Lemma 3.2.2. *Matrix* $\mathbf{A}$ *is diagonalizable.*

Proof. Let $\tilde{\mathbf{A}} = \mathbf{C}^{1/2}\mathbf{A}\mathbf{C}^{-1/2}$ and $\tilde{\mathbf{A}}^T$ be the transpose of $\tilde{\mathbf{A}}$. Then, we have $\tilde{\mathbf{A}}^T = (\mathbf{C}^{1/2}\mathbf{A}\mathbf{C}^{-1/2})^T = -(\mathbf{C}^{-1/2}\mathbf{G}\mathbf{C}^{-1/2})^T = -(\mathbf{C}^{-1/2})^T\mathbf{G}^T(\mathbf{C}^{-1/2})^T = -\mathbf{C}^{-1/2}\mathbf{G}\mathbf{C}^{-1/2} = \tilde{\mathbf{A}}$, which means $\tilde{\mathbf{A}}$ is symmetric.

Since $\tilde{\mathbf{A}}$ is real and symmetric, it is diagonalizable (Theorem 7.2.1 in [7]). Thus, there exists an invertible matrix $\mathbf{Q}$ such that $\mathbf{Q}\tilde{\mathbf{A}}\mathbf{Q}^{-1} = \mathbf{\Gamma}$, in which $\mathbf{\Gamma}$ is a diagonal matrix. We can see $\mathbf{A} = \mathbf{C}^{-1/2}\tilde{\mathbf{A}}\mathbf{C}^{1/2} = \mathbf{C}^{-1/2}\mathbf{Q}^{-1}\mathbf{\Gamma}\mathbf{Q}\mathbf{C}^{1/2} = (\mathbf{Q}\mathbf{C}^{1/2})^{-1}\mathbf{\Gamma}\mathbf{Q}\mathbf{C}^{1/2}$.

There exists an invertible matrix $\mathbf{QC}^{1/2}$ such that $(\mathbf{QC}^{1/2})\mathbf{A}(\mathbf{QC}^{1/2})^{-1} = \mathbf{\Gamma}$ is a diagonal matrix, so $\mathbf{A}$ is diagonalizable (Page 303 Definition 2 in [7]). $\square$

Since $\mathbf{A}$ is diagonalizable and all of its eigenvalues are negative (Lemma 3.2.1), we can easily calculate its eigenvalues. Let $-\lambda_i$ be the *i-th* eigenvalue of $\mathbf{A}$ and $\lambda_i > 0$, we have $\mathbf{A} = \mathbf{WDW}^{-1}$, where $\mathbf{D} = diag\{-\lambda_1, \cdots, -\lambda_N\}$ and $\mathbf{W} = [\vec{w_1}, \cdots, \vec{w_N}]$. $\vec{w_i}$ is the independent eigenvectors associated with $-\lambda_i$. The matrix exponential of $e^{\mathbf{A}l}$ can be diagonalized as

$$e^{\mathbf{A}l} = \sum_{h=0}^{\infty} \frac{l^h(\mathbf{WDW}^{-1})^h}{h!} = \mathbf{W}(\sum_{h=0}^{\infty} \frac{l^h\mathbf{D}^h}{h!})\mathbf{W}^{-1} = \mathbf{W}e^{\mathbf{D}l}\mathbf{W}^{-1}, \tag{3.7}$$

where $e^{\mathbf{D}l} = diag\{e^{-\lambda_1 l}, \cdots, e^{-\lambda_N l}\}$ and $e^{-\lambda_i l}$ is the i-th eigenvalue of $e^{\mathbf{A}l}$.

Lemma 3.2.3. *Matrix* $\mathbf{A}$ *is constant and all the entries of* $-\mathbf{A}^{-1} = [\mathscr{A}_{i,j}]_{N\times N}$ *are positive real numbers and* $\mathscr{A}_{i,j} > 0$*.*

Proof. Let B_i be the i-th element of vector $\mathbf{B}$. Since $\mathbf{T}^{\infty} = -\mathbf{A}^{-1}\mathbf{B}(\mathbf{v})$, the stable-state temperature of the i-th thermal node is $T_i^{\infty} = \sum_{j=1}^{N} \mathscr{A}_{i,j}B_j$. If the μ-th node non-decreasingly changes its power and remain all other nodes' power unchanged, we have $\widetilde{B}_\mu \geq B_\mu$. Let $\widetilde{T}_i^{\infty}$ be the stable state temperature of the i-th node after changing the power; then, we have

$$\begin{aligned}\widetilde{T}_i^{\infty} - T_i^{\infty} &= \sum_{j=1}^{N} \mathscr{A}_{i,j}(\widetilde{B}_j - B_j) \\ &= \sum_{\substack{j=1\\ j\neq\mu}}^{N} \mathscr{A}_{i,j}(\widetilde{B}_j - B_j) + \mathscr{A}_{i,\mu}(\widetilde{B}_\mu - B_\mu).\end{aligned} \tag{3.8}$$

Since $\widetilde{B}_j = B_j$ when $j \neq \mu$, we have $\sum_{\substack{j=1\\ j\neq\mu}}^{N} \mathscr{A}_{i,j}(\widetilde{B}_j - B_j) = \mathbf{0}$.

By contradiction, assume $\mathscr{A}_{i,j} \leq 0$, because $\widetilde{B}_\mu - B_\mu \geq 0$, we can infer that $\widetilde{T}_i^{\infty} < T_i^{\infty}$, which means by non-decreasingly changing the μ-th node's power consumption, while other nodes remain unchanged, results in a non-increasing stable state temperature on the i-th node, which is not realistic. Thus, we can conclude $\mathscr{A}_{i,j} > 0$. $\square$

With Lemma 3.2.3, we can easily prove the following lemma.

Lemma 3.2.4. *Given two constant supply voltage profiles* $\mathbf{v}_1 \geq \mathbf{v}_2$ *running infinitely long, we have* $\mathbf{T}^{\infty}(\mathbf{v}_1) \geq \mathbf{T}^{\infty}(\mathbf{v}_2)$.

Proof. From (3.2), we have $\mathbf{T}^{\infty}(\mathbf{v}_1) - \mathbf{T}^{\infty}(\mathbf{v}_2) = -\mathbf{A}^{-1}[\mathbf{B}(\mathbf{v}_1) - \mathbf{B}(\mathbf{v}_2)]$. Since $\mathbf{B}(\mathbf{v}_1) - \mathbf{B}(\mathbf{v}_2) \geq \mathbf{0}$ when $\mathbf{v}_1 \geq \mathbf{v}_2$, and $-\mathbf{A}^{-1}$ contains all positive entries (Lemma 3.2.3), we have $\mathbf{T}^{\infty}(\mathbf{v}_1) \geq \mathbf{T}^{\infty}(\mathbf{v}_2)$. $\square$

As shown in (3.6), matrix $\mathbf{K}$ plays an important role in determining the stable status temperature of a periodic schedule. Specifically, for matrix $\mathbf{K}$, we have the following lemma and theorem, which are keys for late proofs.

Lemma 3.2.5. *All the elements in matrix* $(\mathbf{I} - \mathbf{K})^{-1}$ *are positive and each entry monotonically decreases with* l, *where* $\mathbf{K} = e^{\mathbf{A}l}$, $l > 0$.

Proof. [**Part 1**]: Prove $(\mathbf{I} - \mathbf{K})^{-1} > \mathbf{0}$.

Let $\rho(e^{\mathbf{A}l})$ denote the *spectral radius* of $e^{\mathbf{A}l}$, we have $\rho(e^{\mathbf{A}l}) = max|e^{\lambda_i l}|$ (page 497 in [95]). Because for all $\lambda_i > 0$, we have for all $0 < e^{-\lambda_i l} < 1$ and $|e^{\lambda_i l}| < 1$, so $\rho(e^{\mathbf{A}l}) < 1$. Since $\rho(e^{\mathbf{A}l}) < 1$, we have $\lim_{H\to\infty}(e^{\mathbf{A}l})^H = \mathbf{0}$.

We adopt the *Neumann Series* (page 618 in [95]) by geometric series formula for matrices version, which can be proved similarly as the geometric series formula for numbers, i.e. $\sum_{h=0}^{H} \mathbf{K}^h = (\mathbf{I} - \mathbf{K}^{H+1})(\mathbf{I} - \mathbf{K})^{-1}$. Thus, we have $(\mathbf{I} - \mathbf{K})^{-1} = \sum_{h=0}^{\infty} \mathbf{K}^h$.

Since all the elements of $e^{\mathbf{A}l}$ are positive (Lemma 1 in [52]), we can conclude $(\mathbf{I} - e^{\mathbf{A}l})^{-1}$ only contains positive elements.

[**Part 2**]: Prove each element in $(\mathbf{I} - \mathbf{K})^{-1}$ monotonically decreases with l.

Let $\mathscr{K}_{k,j}$, $\mu_{k,j}$ and $\phi_{k,j}$ be the element on the k-th row and j-th column of $(\mathbf{I}-\mathbf{K})^{-1}$, $\mathbf{W}$ and $\mathbf{W}^{-1}$, respectively, $k,j \in \{1,\cdots,N\}$. Diagonalize $(\mathbf{I}-\mathbf{K})^{-1}$ by (3.7), we have

$$\begin{aligned}
&(\mathbf{I}-e^{\mathbf{A}l})^{-1} = (\mathbf{I}-\mathbf{W}\cdot e^{\mathbf{D}l}\cdot \mathbf{W}^{-1})^{-1}\\
=&(\mathbf{W}\cdot\mathbf{W}^{-1}-\mathbf{W}\cdot e^{\mathbf{D}l}\cdot\mathbf{W}^{-1})^{-1}\\
=&(\mathbf{W}\cdot(\mathbf{I}-e^{\mathbf{D}l})\cdot\mathbf{W}^{-1})^{-1} = \mathbf{W}\cdot e^{(\mathbf{I}-\mathbf{D}l)^{-1}}\cdot\mathbf{W}^{-1}\\
=&\mathbf{W}\cdot diag\{(1-e^{-\lambda_1 l})^{-1},\cdots,(1-e^{-\lambda_N l})^{-1}\}\cdot\mathbf{W}^{-1}.
\end{aligned} \tag{3.9}$$

Thus, we have $\mathscr{K}_{k,j} = \sum_{i=1}^{N}\mu_{k,i}\cdot\phi_{i,j}\cdot(1-e^{-\lambda_i l})^{-1}$. To prove $\mathscr{K}_{k,j}$ monotonically decreases with l while $\mu_{k,j}$ and $\phi_{k,j}$ are constants, since $\mathscr{K}_{k,j} > 0$, we need to prove each eigenvalue $(1-e^{-\lambda_i l})^{-1}$ monotonically decreases with l.

Since $-\lambda_i < 0$ and $l > 0$, $e^{-\lambda_i l}$ monotonically decreases with l and $0 < e^{-\lambda_i l} < 1$. Then, $(1-e^{-\lambda_i l})$ monotonically increases with l and $0 < 1-e^{-\lambda_i l} < 1$. Thus, $(1-e^{-\lambda_i l})^{-1} > 0$ and it monotonically decreases with l. $\square$

Theorem 3.2.6. *Let $l > 0$ and $\mathbf{0} \le \mathbf{T} \le (\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v_{min}}))$, then $(\mathbf{I}-\mathbf{K})\mathbf{T} \ge \mathbf{0}$, where $\mathbf{K} = e^{\mathbf{A}l}$, $\mathbf{v_{max}} = [v_{max,i}]_{N\times 1}$ and $\mathbf{v_{min}} = [v_{min,i}]_{N\times 1}$. $v_{max,i}$ and $v_{min,i}$ denote the maximum and minimum available supply voltage on the i-th node, respectively.*

Proof. Consider a state interval $\mathbb{I}_q = [t_{q-1}, t_q]$ starts at $\mathbf{T_0} = \mathbf{T}^{\infty}(\mathbf{v_{max}})$ and runs at the mode with supply voltage $\mathbf{v}$. Since $\mathbf{v} \le \mathbf{v_{max}}$, we have $\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v}) \ge \mathbf{0}$ (Lemma 3.2.4). Because given a multi-core platform and a state interval, the temperature on each core must monotonically decrease if all the cores' starting temperature is higher than the running mode's stable state temperature (Theorem 5 in [52]), we have $\forall t \in [t_{q-1}, t_q]$, $\mathbf{T}(t) \le \mathbf{T}^{\infty}(\mathbf{v_{max}})$. Thus, from (3.4), $\mathbf{T}(t)$ can be expressed as

$$\begin{aligned} & e^{\mathbf{A}(t-t_{q-1})}\mathbf{T}^{\infty}(\mathbf{v_{max}}) + (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})\mathbf{T}^{\infty}(\mathbf{v}) \leq \mathbf{T}^{\infty}(\mathbf{v_{max}}) \\ & \Rightarrow (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})(\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v})) \geq \mathbf{0} \end{aligned} \tag{3.10}$$

In contrary, if $\mathbf{T_0} = \mathbf{T}^{\infty}(\mathbf{v_{min}})$, and the system executes at $\mathbf{v}$ with $\mathbf{v} \geq \mathbf{v_{min}}$, the temperature must monotonically increase (Theorem 5 in [52]). Thus, we have

$$\begin{aligned} & e^{\mathbf{A}(t-t_{q-1})}\mathbf{T}^{\infty}(\mathbf{v_{min}}) + (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})\mathbf{T}^{\infty}(\mathbf{v}) \geq \mathbf{T}^{\infty}(\mathbf{v_{min}}) \\ & \Rightarrow (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})(\mathbf{T}^{\infty}(\mathbf{v}) - \mathbf{T}^{\infty}(\mathbf{v_{min}})) \geq \mathbf{0} \end{aligned} \tag{3.11}$$

Since $\mathbf{v_{min}} \leq \mathbf{v} \leq \mathbf{v_{max}}$, we have $\mathbf{T}^{\infty}(\mathbf{v_{min}}) \leq \mathbf{T}^{\infty}(\mathbf{v}) \leq \mathbf{T}^{\infty}(\mathbf{v_{max}})$, which both satisfy (3.10) and (3.11) contemporarily. Thus, $(\mathbf{I} - \mathbf{K})\mathbf{T} \geq \mathbf{0}$ holds throughout our problem and $\mathbf{0} \leq \mathbf{T} \leq (\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v_{min}}))$. $\square$

According to Theorem 3.2.6, as long as the temperatures of all thermal nodes (i.e. $\mathbf{T}$) stay within the feasible range for the given supply voltages (not by external factors), we always have $(\mathbf{I} - \mathbf{K})\mathbf{T} > \mathbf{0}$ for any arbitrary $\mathbf{T}$. With the knowledge of these properties, we are ready to introduce the peak temperature bounding method.

3.3 Peak Temperature Identification and Bounding

Peak temperature is usually a primary concern when designing a real-time computing system. On single-core platforms, it is straightforward to understand that the peak temperature always occurs at the end of the high-speed interval in two-speed schedules [20]. However, on multi-core platforms, peak temperature identification and bounding become more complicated, because different components may follow different speed schedules, and the power densities vary intricately in one chip, which results in that the peak temperature may not always occurs at a scheduling point.

3.3.1 Related Works

There are a few approaches proposed to identify the peak temperature for a multi-core platform. For example, one approach is to search the peak temperature by splitting the execution interval into smaller ones, and assuming each interval has the same power consumptions (e.g. [131, 113, 126]). This kind of approach is computationally expensive and its accuracy heavily depends on the checking granularity. To reduce the computational cost, some approximations are applied, e.g. without considering the lateral heat transfer [101] or simply assuming thermal nodes can immediately reach the stable state temperature [156]. However, these approximations can lead to large error margins, which either caused thermal violation or wasted precious thermal resources. Schor et al. [122] proposed a peak temperature bounding method, which considered all possible scenarios of task arrivals for the critical set of cumulative workload trace. However, this method significantly overestimated the peak value and its computational complexity for profiling the workload demand trace can be high. Since the peak temperature does not necessarily occur at the scheduling point, several analytical solutions are also proposed for interval-wise peak temperature checking in [104] and [52] with a tighter bound for maximal temperature and less computational cost than [122]. All the approaches introduced above assume that task execution times are given. When task execution times can be variable, it is a common practice to bound the worst-case scenarios by adopting the worst-case execution time (WCET) of each task. However, the thermal analysis has its uniqueness and using WCET directly in thermal analysis can be misleading in bounding the peak temperature.

Next, we use a motivation example to show how using WCET directly to capture the peak temperature cannot guarantee the thermal constraints.

Table 3.2: Motivation Example Task Sets

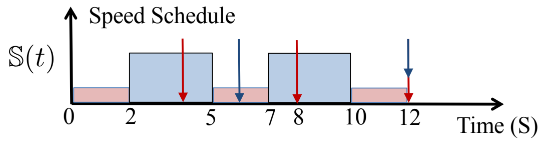
$\mathbb{S}(t)$ **based on WCET**	$\widetilde{\mathbb{S}}(t)$ **based on Varied-ET**
$\tau_a = \{6ms,\ 3ms,\ 1.3V\}$	$\widetilde{\tau}_a = \{6ms,\ 3ms,\ 1.3V\}$
$\tau_b = \{4ms,\ 2ms,\ 0.6V\}$	$\widetilde{\tau}_{b,1} = \{4ms,\ 2ms,\ 0.6V\}$
	$\widetilde{\tau}_{b,2} = \{4ms,\ 0.1ms,\ 0.6V\}$
	$\widetilde{\tau}_{b,3} = \{4ms,\ 2ms,\ 0.6V\}$

* $\widetilde{\tau}_{b,k}$ denotes the k-th instance of task $\widetilde{\tau}_b$.
* τ =$\{$*period, execution time (ET), supply voltage*$\}$.

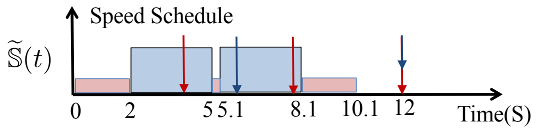
3.3.2 Motivation Example

While it is a common practice to use WCET-based test to bound the worst-case scenarios in a real-time system, the peak temperature for the WCET-based schedule may not be able to bound the worst-case peak temperature. Consider a 3-core platform with each core executing a task set, with their periods equal to the deadlines, as shown in Table 3.2. Let the system use a non-preemptive earliest deadline first (EDF) policy and assume all the tasks/jobs arrive at the beginning of the hyperperiod.

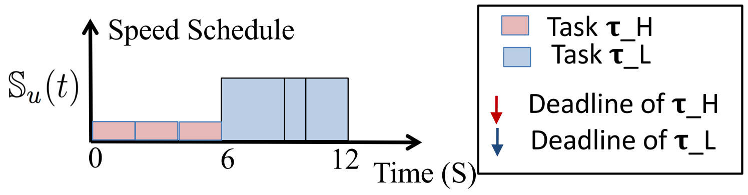
We simulated this motivation example on HotSpot-5.02 [131] at the $65nm$ technology node. The total power consumption (P) is composed of dynamic power and leakage power [52]. Dynamic power is proportional to the cubic of supply voltage and leakage power depends linearly on temperature T as $P_{leak} = \alpha(v) + \beta T(t)$. The total power of the κ-th core is $P_\kappa(t) = \alpha(v_\kappa) + \beta T_\kappa(t) + \gamma(v_\kappa)v_\kappa^3$, where α and γ are positive constants within the interval that $core_\kappa$ runs at supply voltage v_κ. β is a constant. Power parameters were abstracted from the McPAT simulator [82] (Details can be seen in [124] and [52]). Figure 3.2(a) shows the schedule with all the tasks/jobs execute their WCETs; its peak temperature is 82.519°C at $t = 10s$, as shown in Figure 3.2(d). However, when task τ_b runs with shorter execution time



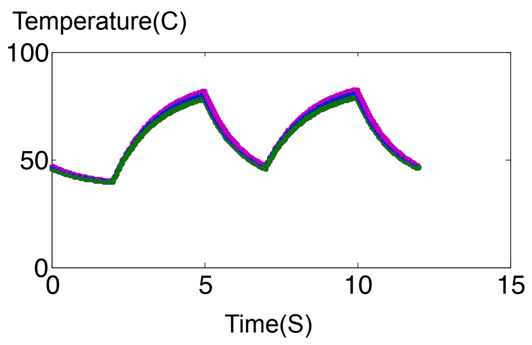
(a) Worst-case execution time schedule $\mathbb{S}(t)$



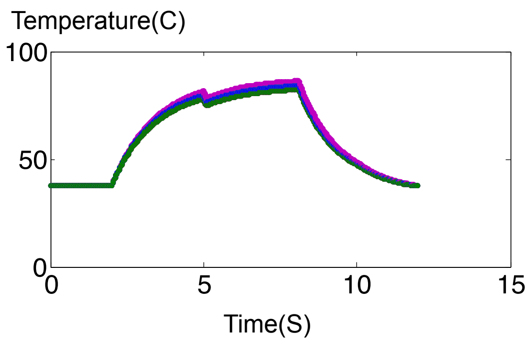
(b) Varied-execution-time schedule $\widetilde{\mathbb{S}}(t)$



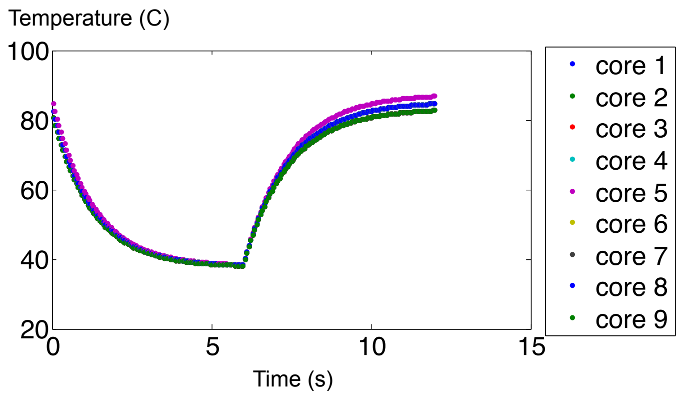
(c) Step-up schedule $\mathbb{S}_u(t)$



(d) $Period = 12s$, $T_{peak}(\mathbb{S}(t)) = 82.5190°\text{C}$, $t_{peak}(\mathbb{S}(t)) = 10s$



(e) $Period = 12s$, $T_{peak}(\widetilde{\mathbb{S}}(t)) = 86.2872°\text{C}$, $t_{peak}(\widetilde{\mathbb{S}}(t)) = 8.1s$



(f) $Period = 12s$, $T_{peak}(\mathbb{S}_u(t)) = 86.6975\circ\text{C}$, $t_{peak}(\mathbb{S}_u(t)) = 12s$

Figure 3.2: Temperature trace of different schedules on a 9-core platform.

than its WCET as Figure 3.2(b), it exhibits a 3.7682°C higher peak temperature as 86.2872°C at $t = 8.1s$ in Figure 3.2(e).

The motivation example shows that the schedule built upon WCETs may not be able to successfully bound the peak temperature. This is because the peak temperature depends more on power density rather than the overall energy consumption (i.e. the integration of overall power consumption).

3.3.3 Bounding the Peak Temperature

In practical scenarios, since the actual execution time deviates from WCET, it is impossible to know the actual execution time until the task has been completed. Lacking the information of the actual execution time becomes a major obstacle for thermal guarantee. As shown in our motivation example, simply plugging in the WCETs in a schedule cannot guarantee the peak temperature constraint. To this end, in this section, we introduce a new concept of "step-up schedule", which can be used to effectively bound the peak temperature in terms of: (1) The multi-core peak temperature may not always occur at a scheduling point; (2) The actual execution time may be missing.

Specifically, in this section, we introduce a new concept of WCET-based *step-up schdule* (Definition 3.3.1) and show that its peak temperature can be easily identified in Theorem 3.3.3. Then, we prove that *step-up schedules* can effectively bound the peak temperature in Theorem 3.3.2-3.3.7.

Definition 3.3.1. *Let multi-core voltage schedule $\mathbb{S}(t)$ contain z state intervals, with $\mathbf{v}_q$ being the voltage vector for the q_{th} state interval $\mathbb{I}_q$. Then $\mathbb{S}(t)$ is called a* **step-up schedule** *if $\mathbf{v}_q \leq \mathbf{v}_{q+1}$, $\forall q \in \{1, \cdots, z-1\}$.*

According to Definition 3.3.1, the voltage for each core is monotonically non-decreasing from the first to the last state interval in a step-up schedule. Note that, the concept of "step-up schedule" was first introduced in our earlier work [20] on single core platforms. The characteristics of a step-up schedule for a multi-core platform become substantially more complicated due to the heat transfer problems as shown below.

First, we prove that the starting temperature of a multi-core schedule does not influence its stable status temperature.

Theorem 3.3.2. *Let $\mathbb{S}(t) = \{\mathbb{I}_q : q = 1 \cdots z\}$ be a periodic multi-core schedule. Then, the stable-state temperature $\mathbf{T}_{ss}(t)$ is independent of the initial temperature $\mathbf{T_0}$.*

Proof. Let $t_0, t_1, \cdots, t_{(z-1)}$ be the starting time for interval $\mathbb{I}_1, \mathbb{I}_2, \cdots, \mathbb{I}_z$, respectively. In addition, let $l_q = t_q - t_{q-1}$ and assume that processor runs voltage profile $\mathbf{v}_q$ within interval $\mathbb{I}_q$, where $q \in \{1, 2, \cdots, z\}$. Based on (3.6), we have

$$\begin{aligned}\mathbf{T}_{ss}(t_0) &= \mathbf{T_0} + (\mathbf{I} - \mathbf{K})^{-1}(\mathbf{T}(t_p) - \mathbf{T_0}) \\ &= \mathbf{T_0} + (\mathbf{I} - \mathbf{K})^{-1}[\mathbf{K} \cdot \mathbf{T_0} + \mathbf{F} - \mathbf{T_0}] \\ &= (\mathbf{I} - \mathbf{K})^{-1} \cdot \mathbf{F}\end{aligned} \tag{3.12}$$

in which $\mathbf{T}(t_p)$ is the ending temperature of the first execution period; $\mathbf{F} = e^{\mathbf{A}\sum_{\theta=2}^{z} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_1})\mathbf{T}_1^\infty + \cdots + (\mathbf{I} - e^{\mathbf{A}l_z})\mathbf{T}_z^\infty$; $\mathbf{K} = e^{\mathbf{A}(\sum_{\theta=1}^{z} l_\theta)} = e^{\mathbf{A}t_p}$. Since neither $\mathbf{K}$ nor $\mathbf{F}$ depends on $\mathbf{T_0}$, $\mathbf{T}_{ss}(t_0)$ does not depend on the starting temperature $\mathbf{T_0}$. □

Theorem 3.3.2 shows that when identifying peak temperature in the stable status, we can assume any starting temperature, e.g. $\mathbf{T_0} = \mathbf{0}$, to ease the presentation. For example, when $\mathbf{T_0} = \mathbf{0}$, for schedule $\mathbb{S}(t) = \{\mathbb{I}_1, \cdots, \mathbb{I}_z\}$ contains z state intervals and $\mathbb{I}_q = [t_{q-1}, t_q]$, the temperature at each consecutive scheduling point from (3.4) can be expressed as

$$\begin{aligned}
\mathbf{T}(t_1) &= e^{\mathbf{A}l_1}\mathbf{T_0} + (\mathbf{I} - e^{\mathbf{A}l_1})\mathbf{T}_1^{\infty} = (\mathbf{I} - e^{\mathbf{A}l_1})\mathbf{T}_1^{\infty}; \\
\mathbf{T}(t_2) &= e^{\mathbf{A}l_2}\mathbf{T}(t_1) + (\mathbf{I} - e^{\mathbf{A}l_2})\mathbf{T}_2^{\infty} \\
&= \sum_{q=1}^{2} e^{\mathbf{A}\sum_{\theta=q+1}^{2} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty}; \\
&\cdots \\
\mathbf{T}(t_h) &= e^{\mathbf{A}l_h}\mathbf{T}(t_{h-1}) + (\mathbf{I} - e^{\mathbf{A}l_h})\mathbf{T}_h^{\infty} \\
&= \sum_{q=1}^{h} e^{\mathbf{A}\sum_{\theta=q+1}^{h} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty}; \\
&\cdots \\
\mathbf{T}(t_p) &= \sum_{q=1}^{z} e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty}.
\end{aligned} \tag{3.13}$$

In this paper, to ease the presentation, we assume temperature starts from $\mathbf{T_0} = \mathbf{0}$ otherwise specified.

In addition, for a step-up schedule, its peak temperature always occurs at the end of the period, as stated in the following theorem.

Theorem 3.3.3. *The peak temperature when repeating a step-up schedule $\mathbb{S}(t)$ periodically from the ambient temperature occurs at the end of the schedule when the temperature reaches the stable status.*

Proof. Assume step-up schedule $\mathbb{S}(t)$ is of period t_p and contains z state intervals with scheduling points $t_0, t_1, \cdots, t_z$. Let $l_q = t_q - t_{q-1}$. Also, let t_x be an arbitrary time instant within the h_{th} interval, i.e. $t_x \in [t_{h-1}, t_h]$ and $\Delta t_x = t_x - t_{h-1}$. Based on (3.6), we have

$$\begin{cases}
\mathbf{T}_{ss}(t_p) & = \mathbf{T}(t_p) + \mathbf{K}(\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(t_p) = (\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(t_p) \\
\mathbf{T}_{ss}(t_x) & = \mathbf{T}(t_x) + \mathbf{K}_x(\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(t_p) \\
& = (\mathbf{I} - \mathbf{K})^{-1}[(\mathbf{I} - \mathbf{K})\mathbf{T}(t_x) + \mathbf{K}_x\mathbf{T}(t_p)],
\end{cases} \tag{3.14}$$

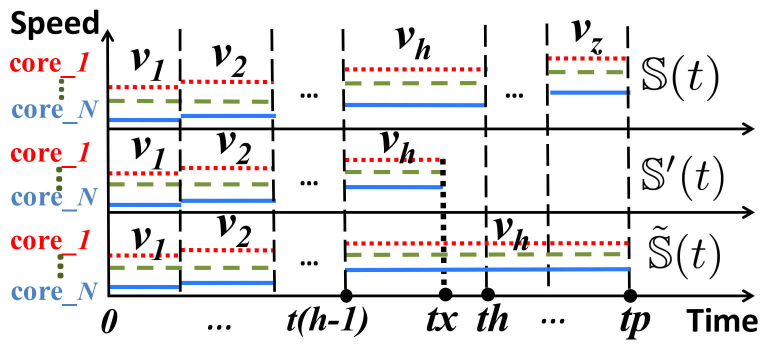


Figure 3.3: Step up schedule proof illustration for Theorem 3.3.3

in which $\mathbf{K} = e^{\mathbf{A}t_p}$ and $\mathbf{K}_x = e^{\mathbf{A}(\Delta t_x + \sum_{\theta=1}^{q-1} l_\theta)}$. Since $(\mathbf{I} - \mathbf{K})^{-1}$ contains all positive elements (Lemma 3.2.5), to prove $\mathbf{T}_{ss}(t_p) \geq \mathbf{T}_{ss}(t_x)$, we want to prove $\mathbf{T}(t_p) - [(\mathbf{I} - \mathbf{K})\mathbf{T}(t_x) + \mathbf{K}_x\mathbf{T}(t_p)] \geq \mathbf{0}$, which is equivalent to prove

$$\begin{aligned} &\mathbf{T}(t_p) - [(\mathbf{I} - \mathbf{K})\mathbf{T}(t_x) + \mathbf{K}_x\mathbf{T}(t_p)] \\ =&(\mathbf{I} - \mathbf{K}_x)(\mathbf{I} - \mathbf{K})[(\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(t_p) - (\mathbf{I} - \mathbf{K}_x)^{-1}\mathbf{T}(t_x)] \geq \mathbf{0}. \end{aligned} \tag{3.15}$$

Since $(\mathbf{I} - \mathbf{K}_x)(\mathbf{I} - \mathbf{K})\mathbf{T} \geq \mathbf{0}$, if $\mathbf{T} \geq \mathbf{0}$ (Theorem 3.2.6), we need to prove that $(\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(t_p) - (\mathbf{I} - \mathbf{K}_x)^{-1}\mathbf{T}(t_x) \geq \mathbf{0}$. From (3.6), $(\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(t_p)$ is the stable status temperature at time t_p of schedule $\mathbb{S}(t)$, and $(\mathbf{I} - \mathbf{K}_x)^{-1}\mathbf{T}(t_x)$ is the stable status temperature at time t_x for the $\mathbb{S}'(t)$ that consists of all state intervals of $\mathbb{S}(t)$ within interval $[0, t_x]$.

To prove $\mathbb{S}(t)$ has a higher peak temperature than $\mathbb{S}'(t)$, we define an intermediate schedule $\widetilde{\mathbb{S}}(t)$ with period t_p, which consists of all the state intervals of $\mathbb{S}'(t)$ within interval $[0, t_{h-1}]$, and keeps constant voltage $\mathbf{v}_h$ within $[t_{h-1}, t_p]$, as shown in Figure 3.3. Then we need to prove $\mathbf{T}_{ss}(\mathbb{S}(t_p)) \geq \mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'(t_x))$.

First, we prove $\mathbf{T}_{ss}(\mathbb{S}(t_p)) \geq \mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p))$, in which $\mathbf{T}_{ss}(\mathbb{S}(t_p)) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(\mathbb{S}(t_p))$ and $\mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p)) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(\widetilde{\mathbb{S}}(t_p))$ from (3.6). $\mathbf{K}$ is identical for both schedule $\mathbb{S}(t)$ and $\widetilde{\mathbb{S}}(t)$ because their periods are the same. Then, we need to prove $\mathbf{T}(\mathbb{S}(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_p))$, which are the ending temperatures of the first period.

Let $\mathbf{v}_q$ and $\widetilde{\mathbf{v}}_q$ be the supply voltage of the q-th interval of schedule $\mathbb{S}(t)$ and $\widetilde{\mathbb{S}}(t)$, respectively. Let $\mathbf{T}_q^\infty = \mathbf{T}^\infty(\mathbf{v}_q)$ and $\widetilde{\mathbf{T}}_q^\infty = \widetilde{\mathbf{T}}^\infty(\widetilde{\mathbf{v}}_q)$. We know $\mathbf{v}_q = \widetilde{\mathbf{v}}_q$ and $\mathbf{T}_q^\infty = \widetilde{\mathbf{T}}_q^\infty$ when $q \in \{1, \cdots, h\}$; $\mathbf{v}_q \geq \widetilde{\mathbf{v}}_q$ and $\mathbf{T}_q^\infty \geq \widetilde{\mathbf{T}}_q^\infty$ when $q \in \{h+1, \cdots, z\}$, since $\mathbb{S}(t)$ is a step-up schedule (Definition 3.3.1). Then, from (3.13), we have

$$\begin{aligned}
\mathbf{T}(\mathbb{S}(t_p)) - \mathbf{T}(\widetilde{\mathbb{S}}(t_p)) &= \sum_{q=1}^{h} \left(e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})(\mathbf{T}_q^\infty - \widetilde{\mathbf{T}}_q^\infty)\right) \\
&\quad + \sum_{q=h+1}^{z} \left(e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})(\mathbf{T}_q^\infty - \widetilde{\mathbf{T}}_q^\infty)\right) \qquad (3.16) \\
&= \mathbf{0} + \sum_{q=h+1}^{z} \left(e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})(\mathbf{T}_q^\infty - \widetilde{\mathbf{T}}_q^\infty)\right).
\end{aligned}$$

Since $e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}$ contains all positive elements (Lemma 1 in [52]), we need to prove $(\mathbf{I} - e^{\mathbf{A}l_q})(\mathbf{T}_q^\infty - \widetilde{\mathbf{T}}_q^\infty) \geq \mathbf{0}$ when $q \in \{h+1, \cdots, z\}$. Since $\mathbf{T}_q^\infty \geq \widetilde{\mathbf{T}}_q^\infty$ when $q \in \{h+1, \cdots, z\}$, the conclusion is proved (Theorem 3.2.6).

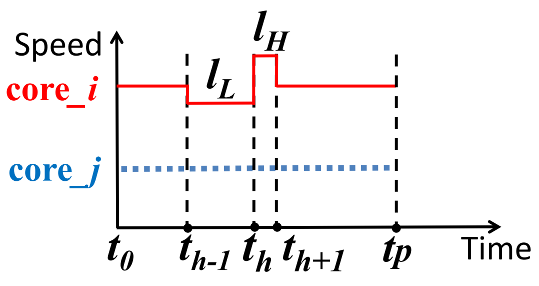
Next, we prove $\mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'(t_x))$. Starting from the same temperature, if we can prove that for each consecutive period the ending temperature of $\widetilde{\mathbb{S}}(t)$ is greater than the one of $\mathbb{S}'(t)$, we are able to claim that in the stable status $\mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'(t_x))$. To prove $\mathbf{T}(\widetilde{\mathbb{S}}(t_p)) \geq \mathbf{T}(\mathbb{S}'(t_x))$ for the first execution period, we know in $[0, t_x]$, $\widetilde{\mathbb{S}}(t)$ and $\mathbb{S}'(t)$ are the same, so $\mathbf{T}(\widetilde{\mathbb{S}}(t_x)) = \mathbf{T}(\mathbb{S}'(t_x))$. Then, within $[t_x, t_p]$ the temperature of schedule $\widetilde{\mathbb{S}}(t)$ is monotonically non-decrease because it is a step-up schedule, so we have $\mathbf{T}(\widetilde{\mathbb{S}}(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_x))$. Then, for the second period, the starting temperature of $\widetilde{\mathbb{S}}(t)$ is greater than $\mathbb{S}'(t)$, so we have $\mathbf{T}(\widetilde{\mathbb{S}}(t_x)) \geq \mathbf{T}'(\mathbb{S}(t_x))$. Therefore for any time within $[0, t_x]$ of the second period, $\mathbf{T}(\widetilde{\mathbb{S}}(t)) \geq \mathbf{T}(\mathbb{S}(t))$. Similarly, since in $[t_x, t_p]$ the temperature of schedule $\widetilde{\mathbb{S}}(t)$ is monotonically non-decrease, we have $\mathbf{T}(\widetilde{\mathbb{S}}(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_x))$. So on and so forth, we prove $\mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'(t_x))$. $\square$

Based on (3.4) and (3.6), we can quickly identify the peak temperature with linear complexity. Furthermore, the peak temperature of a step-up schedule can be used to bound the peak temperature of an *arbitrary schedule*. Before we introduce this conclusion, we first introduce the following definition.

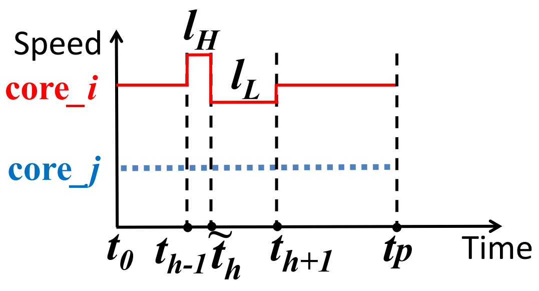
Definition 3.3.4. *Given an* ***arbitrary periodic schedule*** $\mathbb{S}(t)$*, the corresponding step-up schedule (denoted as* $\mathbb{S}_u(t)$*) is the periodic schedule that, for each core, the schedule consists of the same scheduling intervals as that in* $\mathbb{S}(t)$*, but these intervals are ordered according to a non-decreasing order of their supply voltages.*

To prove that a step-up schedule can help to bound the peak temperatures, we first introduce the following lemma.

Lemma 3.3.5. *Let* $\mathbb{S}(t)$ *and* $\widetilde{\mathbb{S}}(t)$ *be two periodic schedules, with the same period* t_p*, and all cores run with the same constant supply voltages/frequencies, except for core*$_i$ *during* h*-th and* $(h+1)$*-th state interval. For* $\mathbb{S}(t)$*, core*$_i$ *uses the mode with voltage* v_L *(*v_H*, resp.) for the* h*-th (*$(h+1)$*-th, resp.) state interval and* $v_H \geq v_L$*. In* $\widetilde{\mathbb{S}}(t)$*, core*$_i$ *exchanges the* h*-th and* $(h+1)$*-th state intervals of* $\mathbb{S}(t)$*. Let* $\mathbf{T}_{ss}(\mathbb{S}(t))$ *(*$\mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t))$*, resp.) denote the temperature at* t *when running schedule* $\mathbb{S}(t)$ *(*$\widetilde{\mathbb{S}}(t)$*), resp.) in the stable status. Then, we have* $\mathbf{T}_{ss}(\mathbb{S}(t_p)) \geq \mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p))$*.*



(a) $\mathbb{S}(t)$



(b) $\widetilde{\mathbb{S}}(t)$

Figure 3.4: Lemma 3.3.5 illustration.

Proof. To prove $\mathbf{T}_{ss}(\mathbb{S}(t_p)) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(\mathbb{S}(t_p))$ is greater than $\mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p)) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(\widetilde{\mathbb{S}}(t_p))$, where $\mathbf{K} = e^{\mathbf{A}t_p}$ are the same, we need to prove for the first period

$\mathbf{T}(\mathbb{S}(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_p))$. Since the supply voltage in $t \in [t_{h+1}, t_p]$ are the same for $\mathbb{S}(t)$ and $\widetilde{\mathbb{S}}(t)$, we then need to prove $\mathbf{T}(\mathbb{S}(t_{h+1})) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_{h+1}))$.

In the first period of $\mathbb{S}(t)$, based on (3.4), we have

$$\begin{cases} \mathbf{T}(\mathbb{S}(t_h)) & = e^{\mathbf{A}\cdot l_L}\mathbf{T}(\mathbb{S}(t_{h-1})) + (\mathbf{I} - e^{\mathbf{A}\cdot l_L})\mathbf{T}_L^{\infty} \\ \mathbf{T}(\mathbb{S}(t_{h+1})) & = e^{\mathbf{A}\cdot l_H}\mathbf{T}(\mathbb{S}(t_h)) + (\mathbf{I} - e^{\mathbf{A}\cdot l_H})\mathbf{T}_H^{\infty}, \end{cases} \tag{3.17}$$

where $\mathbf{T}_L^{\infty}$ and $\mathbf{T}_H^{\infty}$ are the constant temperatures when $core_i$ runs low-speed mode v_L and v_H long enough, respectively, while other cores keep constant mode. Then, combine $\mathbf{T}(\mathbb{S}(t_h))$ and $\mathbf{T}(\mathbb{S}(t_{h+1}))$ in (3.17), for $\mathbb{S}(t)$ we have

$$\begin{aligned} \mathbf{T}(\mathbb{S}(t_{h+1})) =& e^{\mathbf{A}\cdot l_H} e^{\mathbf{A}\cdot l_L}\mathbf{T}(\mathbb{S}(t_{h-1})) + e^{\mathbf{A}\cdot l_H}(\mathbf{I} - e^{\mathbf{A} l_L})\mathbf{T}_L^{\infty} \\ &+ (\mathbf{I} - e^{\mathbf{A}\cdot l_H})\mathbf{T}_H^{\infty}. \end{aligned} \tag{3.18}$$

Similarly, for $\widetilde{\mathbb{S}}(t)$ we have

$$\begin{aligned} \mathbf{T}(\widetilde{\mathbb{S}}(t_{h+1})) =& e^{\mathbf{A}\cdot l_H} e^{\mathbf{A}\cdot l_L}\mathbf{T}(\mathbb{S}(t_{h-1})) + e^{\mathbf{A}\cdot l_L}(\mathbf{I} - e^{\mathbf{A} l_H})\mathbf{T}_H^{\infty} \\ &+ (\mathbf{I} - e^{\mathbf{A}\cdot l_L})\mathbf{T}_L^{\infty}. \end{aligned} \tag{3.19}$$

Since $\mathbb{S}(t)$ and $\widetilde{\mathbb{S}}(t)$ start from the same temperature and run the same schedule in $[0, t_{h-1}]$, we can infer $\mathbf{T}(\mathbb{S}(t_{h-1})) = \mathbf{T}(\widetilde{\mathbb{S}}(t_{h-1}))$. Thus, to prove $\mathbf{T}(\mathbb{S}(t_{h+1})) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_{h+1}))$, we need to prove

$$\mathbf{T}(\mathbb{S}(t_{h+1})) - \mathbf{T}(\widetilde{\mathbb{S}}(t_{h+1})) = (\mathbf{I} - e^{\mathbf{A} l_H})(\mathbf{I} - e^{\mathbf{A} l_L})(\mathbf{T}_H^{\infty} - \mathbf{T}_L^{\infty}) \geq \mathbf{0} \tag{3.20}$$

Since $v_H \geq v_L$, we have $\mathbf{T}_H^{\infty} - \mathbf{T}_L^{\infty} \geq \mathbf{0}$, and, thus $\mathbf{T}(\mathbb{S}(t_{h+1})) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_{h+1}))$ (Theorem 3.2.6). Then, since $\mathbb{S}(t)$ and $\widetilde{\mathbb{S}}(t)$ run at same speeds within $t \in [t_{h+1}, t_p]$, we can infer $\mathbf{T}(\mathbb{S}(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_p))$.

□

Lemma 3.3.5 indicates that, as a high-speed interval moves toward the end of a periodic schedule, it tends to increase the temperature at the end of the schedule during the stable status. With the help of the lemma, we are now ready to introduce the following theorem.

Theorem 3.3.6. *Given an arbitrary periodic schedule $\mathbb{S}(t)$ and its corresponding step-up schedule $\mathbb{S}_u(t)$ with period of t_p, let $T_{peak}(\mathbb{S}(t))$ and $T_{peak}(\mathbb{S}_u(t))$ be the peak temperature during the stable status. Then, $T_{peak}(\mathbb{S}(t)) \leq T_{peak}(\mathbb{S}_u(t))$.*

Proof. This theorem can be proved based on the facts that both $\mathbb{S}(t)$ and $\mathbb{S}_u(t)$ are periodic and the multi-core thermal model presented in (3.2) is a linear time-invariant system [122, 4], following the superposition principle: (1) The thermal impact at one time instant is the sum of the thermal impact by each core; (2) The thermal impact of each core is the sum of the impact by each state interval in the schedule. With the assistance of Lemma 3.3.5, Theorem 3.3.6 can therefore be proved. $\square$

For periodic tasks with variable execution times, we can then bound the worst-case peak temperature by constructing the *WCET-based* step-up schedule, and its peak temperature is guaranteed to be no lower than the peak temperature in any run-time scenarios. This conclusion is stated in the following Theorem.

Theorem 3.3.7. *Given a periodic schedule $\mathbb{S}(t)$ for a task set with fixed periods, deadlines but variable execution times, the corresponding WCET-based step-up schedule bounds the peak temperature for $\mathbb{S}(t)$ at different run-time scenarios.*

Proof. Consider a WCET-based step-up schedule $\widetilde{\mathbb{S}}(t)$ and its corresponding varied execution time schedule $\mathbb{S}(t)$, assume $\widetilde{\mathbb{S}}(t)$ processing the workload for $\widetilde{l}_h$ length in the h-th interval, while $\mathbb{S}(t)$ processing the workload for the length of l_a and idle for the length of l_b ($l_a + l_b = \widetilde{l}_h$ and $l_a, l_b \geq 0$). Let $\widetilde{\mathbf{T}}(t)$ and $\mathbf{T}(t)$ denote the temperature at time t for $\widetilde{\mathbb{S}}(t)$ and $\mathbb{S}(t)$, respectively. Then, since $\widetilde{\mathbb{S}}(t)$ and $\mathbb{S}(t)$ are the same within $[0, t_{h-1} + l_a]$, we have $\widetilde{\mathbf{T}}(t_{h-1} + l_a) = \mathbf{T}(t_{h-1} + l_a)$.

Since the processor consumes less power between $[t_{h-1} + l_a, t_h]$ than $[t_{h-1}, t_{h-1} + l_a]$, we have $\mathbf{T}_b^\infty \le \mathbf{T}_h^\infty$, where $\mathbf{T}_b^\infty$ denotes the stable state temperature as consuming the same power of interval $[t_{h-1} + l_a, t_h]$. Aaccording to (3.4), we have

$$\begin{cases} \widetilde{\mathbf{T}}(t_h) & = e^{\mathbf{A}l_b}\widetilde{\mathbf{T}}(t_{h-1} + l_a) + (\mathbf{I} - e^{\mathbf{A}l_b})\mathbf{T}_h^\infty \\ \mathbf{T}(t_h) & = e^{\mathbf{A}l_b}\mathbf{T}(t_{h-1} + l_a) + (\mathbf{I} - e^{\mathbf{A}l_b})\mathbf{T}_b^\infty \end{cases} \tag{3.21}$$

Then, we have

$$\widetilde{\mathbf{T}}(t_h) - \mathbf{T}(t_h) = (\mathbf{I} - e^{\mathbf{A}l_b})(\mathbf{T}_h^\infty - \mathbf{T}_b^\infty) \tag{3.22}$$

Since $\mathbf{T}_h^\infty \ge \mathbf{T}_b^\infty$, we can infer $\widetilde{\mathbf{T}}(t_h) \ge \mathbf{T}(t_h)$ (Theorem 3.2.6). Then, for the next state interval $[t_h, t_{h+1}]$, we have

$$\begin{cases} \widetilde{\mathbf{T}}(t_{h+1}) & = e^{\mathbf{A}l_{h+1}}\widetilde{\mathbf{T}}(t_h) + (\mathbf{I} - e^{\mathbf{A}l_{h+1}})\mathbf{T}_{h+1}^\infty \\ \mathbf{T}(t_{h+1}) & = e^{\mathbf{A}l_{h+1}}\mathbf{T}(t_h) + (\mathbf{I} - e^{\mathbf{A}l_{h+1}})\mathbf{T}_{h+1}^\infty \end{cases} \tag{3.23}$$

and

$$\widetilde{\mathbf{T}}(t_{h+1}) - \mathbf{T}(t_{h+1}) = e^{\mathbf{A}l_{h+1}}(\widetilde{\mathbf{T}}(t_h) - \mathbf{T}(t_h)) \tag{3.24}$$

Since all the elements of $e^{\mathbf{A}l_{h+1}}$ are positive (Lemma 1 in [52]), we can conclude $\widetilde{\mathbf{T}}(t_{h+1}) \ge \mathbf{T}(t_{h+1})$. So on and so forth, for all the later consecutive intervals, the WCET-based step-up schedule's temperature is no lower than that of the varied execution time schedule, so the theorem is proved. $\square$

With Theorem 3.3.2-3.3.7, we can finally bound the peak temperature when scheduling a periodic task sets with variable execution times, as formulated below.

Corollary 3.3.8. *Given a periodic schedule $\mathbb{S}(t)$, if the corresponding WCET-based step-up schedule $\mathbb{S}_u(t)$ satisfies the T_{max} constraint, then, $\mathbb{S}(t)$ must satisfy the T_{max} constraint, when tasks do not take their worst-case execution time.*

In other words, a feasible periodic schedule—with regard to deadline constraints—is also thermally feasible if its corresponding WCET-based step-up schedule can satisfy the peak temperature constraint.

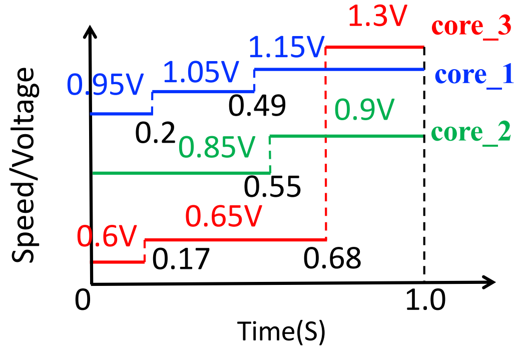
3.4 Experimental Results

In this section, we validate a series of theorems and simulate the characteristics of the proposed step-up schedules.

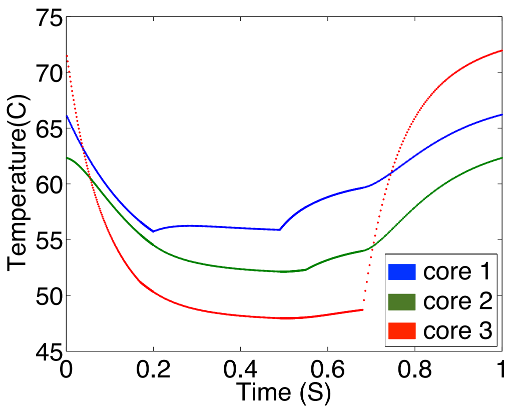
The proposed algorithm is tested on hypothetical multi-core configurations of Alpha 21264 processors, with topologies of 2×1, 3×1, 3×2 and 3×3 layout, of $4mm \times 4mm$ core size. The processing cores consist of several conductive layers, e.g. die layer, heat spreader, heat sink and heat-to-air interface, etc. Since we study the system-level temperature-related problems, we, therefore, simplify the floor-plan to the core-level. In particular, when calculating the temperature, the power of each core in the simplified floor-plan is equal to the sum of the power of all the blocks of the original Alpha 21264 floor-plan that is constrained by this core. The thermal-related parameters, such as thermal capacitance and resistance are abstracted from HotSpot-5.02 [131] at $65nm$ technology node. The power consumption of each core is computed by the models shown in (2.1) and the power parameters are abstracted from the McPAT simulator [82]. We assumed that the available supply voltages for each core are in the range of $[0.6V, 1.3V]$ with a $0.05V$ step size. The ambient temperature was set to be $T_{amb} = 35\,°\mathrm{C}$, unless otherwise specified.

3.4.1 Properties of Step-Up Schedules

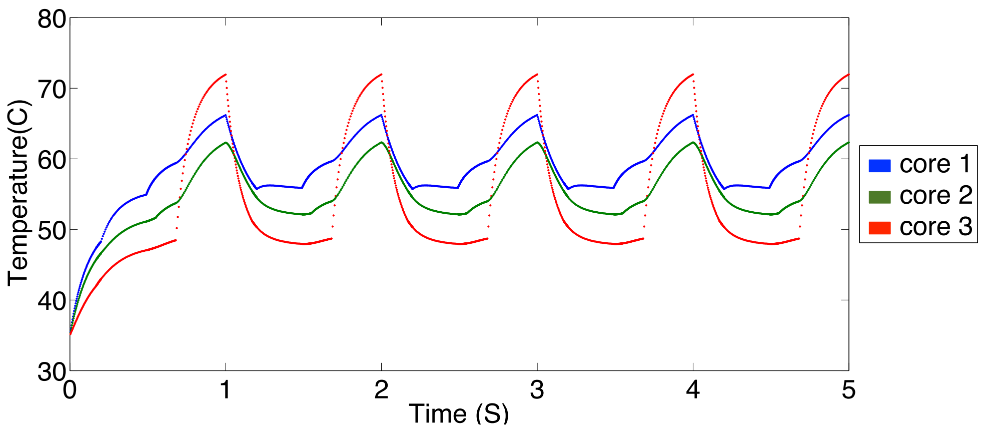
By randomly selecting periods and creating non-decreasing speed levels within one period, we generate a large set of random step-up schedules and collect the temper-



(a)

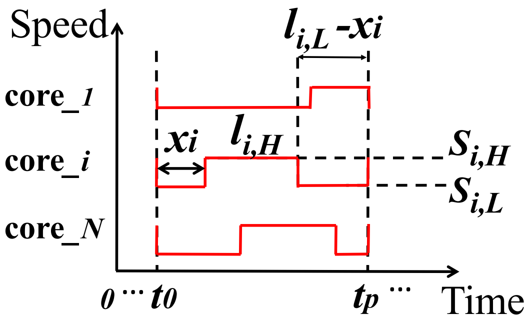


(b)

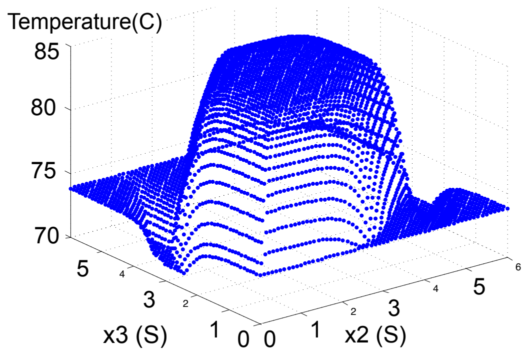


(c)

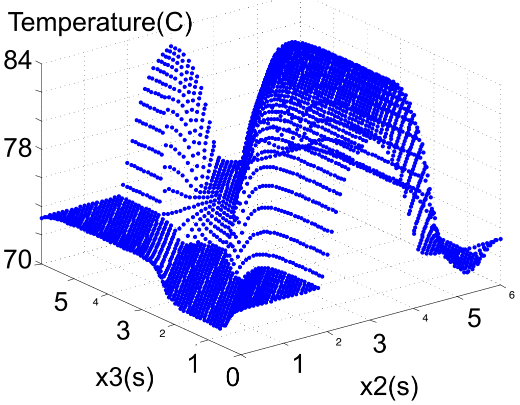
Figure 3.5: (a) Speed schedule on a 3-core platform. (b) Temperature trace in the stable status. (c) Temperature trace starting from $T_{amb} = 35°C$.



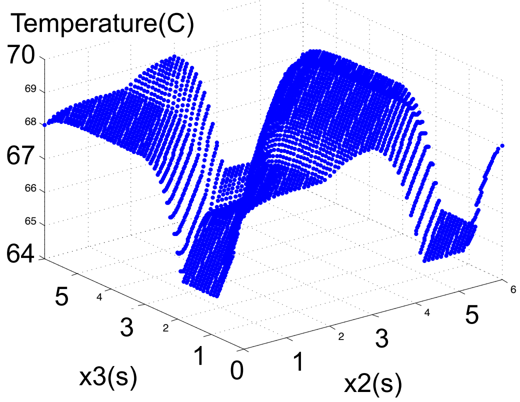
(a)



(b) case 1



(c) case 2



(d) case 3

Figure 3.6: (a) An N-core schedule with shifting *phase*. (b,c,d) Peak temperature changes differently according to the *phase* (x_i) with settings in Table 3.3.

ature traces using HotSpot. Fig. 3.5(a) shows a sample step-up schedule on a 3-core platform. The period of the schedule is set to be 1 second and each core has up to 3 different intervals. As we can see from Fig. 3.5(c) and 3.5(b), when starting from the ambient temperature, the temperature of each core monotonically increases and reaches its peak value at the end of the period, which conforms to Theorem 3.3.3.

3.4.2 Bounding Peak Temperature using Step-Up Schedules

Note that, in our approach, we use the corresponding step-up schedule to bound the peak temperature of a schedule. The question is how tight the peak temperature

Table 3.3: Different settings for testing peak temperature variations by different phases ($t_p = 6s$) on a 3-core platform.

Case	**coreID**	$l_{i,H}(s)$	$S_{i,H}$	$S_{i,L}$	T_{peak}^{max}	T_{peak}^{min}(°C)
	core 1	3.0	1.3	0.6		
case 1	core 2	3.0	1.3	0.6	84.1299	71.2224
	core 3	3.0	1.3	0.6		
	core 1	3.0	1.3	0.6		
case 2	core 2	1.2	1.3	0.6	83.4173	71.0797
	core 3	4.2	1.3	0.6		
	core 1	3.0	0.8	0.6		
case 3	core 2	1.2	1.15	0.8	69.0676	64.7403
	core 3	4.2	1.3	0.9		

* The processing speed $S_{i,H}$ and $S_{i,L}$ are denoted by their corresponding supply voltages.

bound can be, since an overly pessimistic bound can compromise the throughput maximization goal. We use experiments to study the potential impacts by comparing the peak temperature of a step-up schedule with those from different non-step-up schedules.

In our experiment, we randomly constructed a large number of test cases for schedules on a 3-core platform with the settings shown in Table 3.3. We constructed the test cases such that they have different initial starting times for the high/low-speed intervals (which we called *phases*), as shown in Fig. 3.6(a). Specifically, we let $core_1$'s x_1 be fixed at the length of its low-voltage mode, but vary x_2 of $core_2$ and x_3 of $core_3$ by a 0.1 second step size from 0 to the lengths of their low-voltage intervals. In the last column of Table 3.3, T_{peak}^{max} and T_{peak}^{min} denote the maximum/minimum peak temperature among different x_2 and x_3 selections, respectively. Fig. 3.6(b), 3.6(c), 3.6(d) also show the peak temperature variations of different x_2 and x_3 selections. We can observe that in Fig. 3.6, due to the variation of the high/low-speed ratios and speed levels, the peak temperature changes in different patterns and the temperature differences can be significant, e.g.

$T_{peak}^{max} - T_{peak}^{min} = 12.9075$°C in case 1, which takes 26.27% of the temperature margin between T_{amb} and T_{peak}^{max}. This difference also varies with the hyperperiod of the schedule. Table 4.2 shows the maximum possible peak temperature differences when the time scaling is applied to the schedule. As we can see from Table 3.4, as t_p decrease with out change the running mode within each interval, the hyperperiod of the schedule decreases, and the maximum possible peak temperature differences decrease significantly. Note that, when $t_p = 0.5s$, the maximum difference is only 1.8314°C. This shows that using step-up schedules to bound the peak temperature, coupled with the m-Oscillating scheduling scheme in our approach, is not only highly efficient, but also very effective.

Table 3.4: Peak temperature variations (in °C) by different m for case 1 of Table 3.3.

$\mathbf{t_p}$	**6s**	**2s**	**1s**	**0.5s**
T_{peak}^{max}	84.1299	80.9767	73.8299	63.2497
T_{peak}^{min}	71.2224	69.2141	67.8739	61.4183

3.5 Summary

In this chapter, based on the well-known multi-core RC-thermal model, we analytically prove a series of fundamental and provable principles for thermal model, peak temperature identification and bounding methods, which are key to thermal-constrained computer system design. These conclusions emphasize thermal guarantees and they are general enough to be applied on 2D, 3D multi-core platforms and other linear-time-invariant (LTI) systems that may be of interest from a temperature-aware standpoint.

CHAPTER 4

PEAK TEMPERATURE MINIMIZATION AND THROUGHPUT MAXIMIZATION ON MULTI-CORE PLATFORMS

In the previous chapter, we introduced the well-known RC-thermal model on multi-core platform, and studied the a series of fundamental principles based on the characteristics of the system matrix. Then, we proved a WCET-based "step-up schedule" can effectively bound the peak temperature on any arbitrary case, which provide deterministic guarantee of thermal constraints. Based on these thermal-aware design guidelines, in this chapter, we investigate how to apply real-time scheduling techniques to minimize the peak temperature and, therefore, maximize the system throughput under a temperature threshold on multi-core processors.

The rest of this chapter is organized as follows. Section 4.1 discusses the related work. Section 4.2 utilizes a searching algorithm for multi-core throughput maximization. Section 4.2.2 formally proves a serious of peak temperature minimization on multi-core platforms. Section 4.3 presents an *M-Oscillating* schedule with non-negligible overhead to maximize the system throughput. Experimental results are presented in Section 4.4, and Section 4.5 concludes this chapter.

4.1 Related Work

Since temperature closely relates to power consumption, power metrics are commonly used as temperature control indexes. Some studies maximize the performance under a power cap, such as [151], but cannot guarantee the temperature constraint. To this end, thermal safe power (TSP) is proposed as a novel power budget index for a safer and higher throughput under a peak temperature constraint than thermal design power (TDP) [106]. Later, Khdr et al. [74] transform the thermal constraint to maximally allowed power density, by which the system performance can be maximized without thermal violation on tiled heterogeneous multi-core processors. However, due to the non-linear correlations between power and temperature and the possible occurrence of spatial and temporal power/thermal unbalancing, power-indexed thermal management is overly pessimistic for multi-core processors.

To mitigate the thermal crisis, some packaging-aware thermal control methodologies have been explored, such as building heat sinks, heat spreaders, cooling fans or other advanced cooling mechanisms (e.g. embedded micro-channel liquid cooling on 3D processors or using phase-changing cooling materials). However, designing such a heat dissipation package is uneconomical if not infeasible [131], and it is unsuitable for hand-held devices [119].

Alternatively, operating system level mechanisms, e.g. *dynamic thermal management* (DTM), is exploited by adjusting the processing speeds using *dynamic voltage/frequency scaling* (DVFS) or turning off the unused cores using *dynamic power management* (DPM). For example, "hot-and-cold" job swapping [115], the feedback control scheme [151], and other techniques such as those in [119, 111, 126] belong to this category. Based on different design stages, the proactive approaches, such as [126], develop their solutions during the design time, which can tolerate a higher computational overhead and use more accurate power or thermal models. On the contrary, the reactive approaches, such as [74, 119, 111], make decisions online. Although online approaches can be flexible and adaptive, the results are often degraded or cannot guarantee the given thermal constraints due to large uncertainty of program execution and the simplified models adopted for cost reason.

In this chapter, we develop a proactive DTM scheme to optimize the throughput while ensuring the peak temperature constraint. Different from existing work, such as sprinting the speed to boost a transient performance [114, 40], our work focuses on the periodic schedules that can deliver a steady and sustainable performance in the stable status. There are a few papers published [101, 56, 71, 145, 18] with research closely related to our work. Under peak temperature constraints, Mutapcic et al. [101] applied a convex optimization method to solve the throughput maximization problem. However, there exist two problems with this approach. First, this work assumed that the working frequency can be instantaneously and continuously varied, which may not be realistic in practice. Second, the heat transfer among different processing units has been ignored, which may render a suboptimistic solution and violate the temperature constraint when it is applied. Hanumaiah et al. [56] solved a multi-core task-mapping problem with speed control on different cores to minimize the latest completion time. However, this work simply assumed the peak temperature always occurred at a scheduling point (the time instant when at least one core changes its running mode), which may not always be the case [124, 104]. Other approaches are also proposed, e.g. machine learning approach in [71], integer linear programming approach (ILP) in [145] or analytical study in [18]. However, these works either ignore temperature dynamics, e.g. [71, 18], or the computational overhead does not scale well with the problem size, e.g. [145].

In this chapter, we extend the concept of the "m-Oscillating schedule," originally defined for single-core schedule [63], to multi-core platforms, to maximize computing performance without violating the peak temperature constraint.

Based on the system and thermal models presented in Chapter 3, the problem to solve in this chapter can be formulated as follows.

Problem 4.1.1. *Given a multi-core platform $\mathfrak{N}$ and its peak temperature threshold T_{max}, set a running mode to each core and repeat the settings periodically to maximize the chip-wide throughput with the peak temperature below T_{max} all the time.*

4.2 Peak Temperature Minimization and Throughput Maximization

Before presenting our approach, we first show a motivation example. The studies of [101, 56] solved the multi-core throughput maximization problem by assuming the speed of each core can be continuously and instantaneously varied, which is not always possible in practice. In our research, we adopt a more realistic model that each processor features discrete running modes (supply voltage/frequency).

Given the existing work, an intuition is to round down the speed to the available discrete one (as an extension of Section VI-D in [56]) to maintain the peak temperature constraint. We call this approach as the lower neighboring speed (**LNS**) method. This approach can guarantee the peak temperature constraint since if the running mode is not higher than the speed profile that leads to T_{max}, the temperature will never exceed T_{max} (Theorem 9 and Definition 3 in [52]). However, the results might become overly pessimistic when the available speed levels are limited. To improve the results, we can exhaustively search all the speed combinations on different cores that can maximize the performance without exceeding the temperature threshold. We call this approach the exhaustive search (**EXS**), as shown in Algorithm 1.

Algorithm 1 assumes each core runs at one unique discrete mode and, thus, the temperature eventually reaches the constant value $\mathbf{T}^{\infty}$. However, its complexity increases exponentially with the problem size as $O(size(\mathbf{f})^N)$, where $size(\mathbf{f})$ denotes the total number of discrete speed levels. Moreover, since each core in **LNS** and **EXS** can only execute one single speed, the temperature "headroom" cannot be filled by raising the speed of any core to the next higher level due to the possible

Algorithm 1 Exhaustive Search Method (EXS).

```
1: Input: multi-core platform $\mathfrak{N} = \{core_i | i = 1 \cdots N\}$ and $T_{max}$
2: Output: $THR_{max}$; $\boldsymbol{f}_{optimal}$

3: $\boldsymbol{f} = [f_1, \cdots, f_N]$ and $THR_{max} = 0$;
4: for $f_1 = f_{lowest}$ to $f_{highest}$ do
5:   $\cdots$
6:   for $f_N = f_{lowest}$ to $f_{highest}$ do
7:     if $(max(\mathbf{T}^{\infty}) \leq T_{max})\&\&(sum(\boldsymbol{f}) \geq THR_{max})$ then
8:       $THR_{max} \leftarrow sum(\boldsymbol{f})$;
9:       $\boldsymbol{f}_{optimal} = \boldsymbol{f}$;
10:     end if
11:   end for
12:   $\cdots$
13: end for
14: return $THR_{max}$ and $\boldsymbol{f}_{optimal}$
```

violation of T_{max}. Is it possible to use more than one speed on each core to achieve a better performance with temperatures staying below T_{max}?

Table 4.1: Performance of different approaches

Cores	Continuous speed (V)	Discrete speed schedule settings (0.6V:1.3V)				
		LNS	EXS	$t_p = 20ms$	$10ms$	$5ms$
core 1	1.2085	1: 0	1: 0	0.83: 0.17	0.77: 0.23	0.73: 0.27
core 2	1.1748	1: 0	1: 0	0.18: 0.82	0.18: 0.82	0.18: 0.82
core 3	1.2085	1: 0	0: 1	0.83: 0.17	0.77: 0.23	0.73: 0.27
THR	1.1972	0.6	0.83	0.87	0.90	0.92

* Assume discrete speeds with supply-voltage of 0.6V and 1.3V are available. The settings for continuous speed show the supply voltage for each core. The settings with two available discrete speed levels show the ratio of interval lengths when these two modes (0.6V and 1.3V) are applied to each core.

Consider a 3-core processor with $T_{max} = 65°C$, whose power and thermal models are further detailed in Chapter 3. We can observe from Table 4.1 that when continuous speeds are available, the single-speed performance (short for Perf.) can be as high as 1.1972 in the second column. In practical scenarios, if only discrete speeds are available, e.g. speeds with supply voltages of 0.6V and 1.3V, **LNS** and **EXS** methods exhibit pessimistic performances as 0.6 and 0.83, respectively. Alternatively, if we use two speeds interchangeably as shown in column 5 to 7 in Table 4.1 and Fig. 4.1, it utilizes the temperature "headroom" more efficiently to improve the performance. In addition, as the period of two-speed schedules become smaller, i.e. when t_p changes from $20ms$ to $5ms$ in Table 4.1, the overall performance improves.

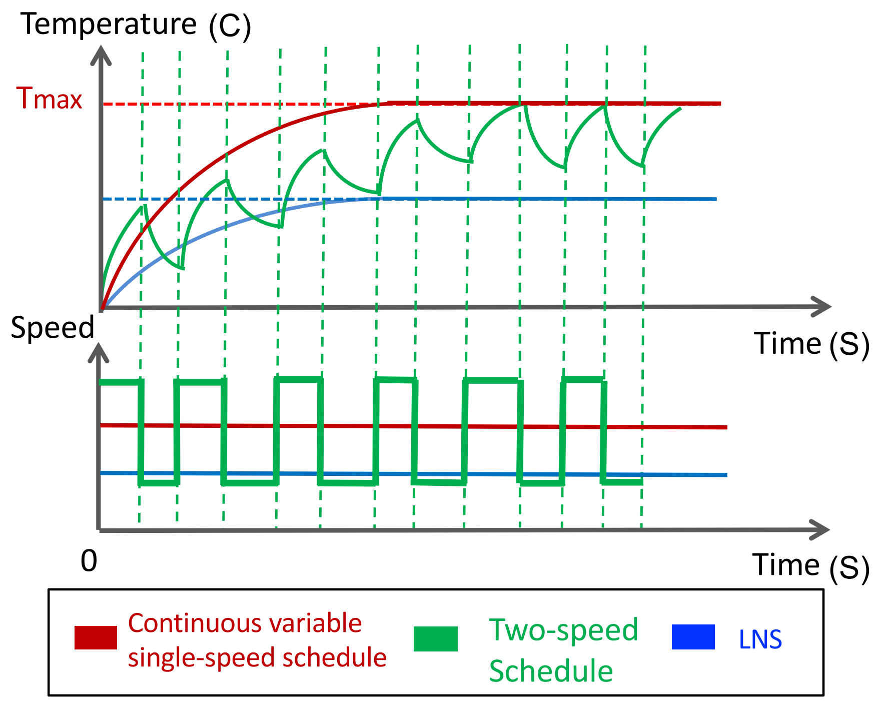


Figure 4.1: Illustration of temperature traces of different approaches.

4.2.1 Choose Two Neighboring Running Modes

Theorem 3.3.6 can be employed to bound the peak temperature for arbitrary periodic multi-core schedules, which helps to ensure that the peak temperature constraint is not violated. To maximize the performance without exceeding the given peak temperature constraint, it is desirable that a periodic schedule can lead to the peak temperature as low as possible while maintaining the same throughput. Specifically, we have the following theorem.

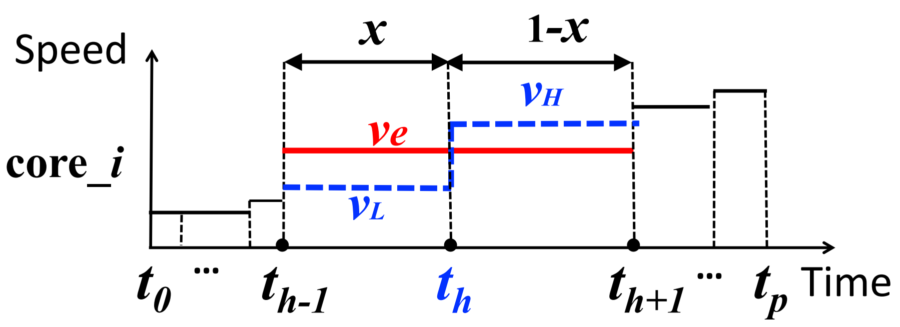


Figure 4.2: Illustration for Theorem 4.2.1.

Theorem 4.2.1. *Let $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$ be two periodic step-up schedules with period t_p, that are exactly the same except for core$_i$. For $\mathbb{S}_{u1}(t)$, core$_i$ uses a constant mode with voltage v_e throughout the period, but for $\mathbb{S}_{u2}(t)$, core$_i$ uses the mode with voltage v_L for l_L seconds followed by v_H for l_H seconds ($l_L + l_H = t_p$) such that*

$$(l_L + l_H) \cdot v_e = l_L \cdot v_L + l_H \cdot v_H. \tag{4.1}$$

Let $T_{peak}(\mathbb{S}_{u1}(t))$ denote the peak temperature when running schedule $\mathbb{S}_{u1}(t)$ periodically. Then, we have $T_{peak}(\mathbb{S}_{u1}(t)) \leq T_{peak}(\mathbb{S}_{u2}(t))$.

Proof. **[Part 1]** According to Theorem 3.3.3, we have $T_{peak}(\mathbb{S}_{u1}(t_p)) = max\big(\mathbf{T}_{ss}(\mathbb{S}_{u1}(t_p))\big)$ and $T_{peak}(\mathbb{S}_{u2}(t_p)) = max\big(\mathbf{T}_{ss}(\mathbb{S}_{u2}(t_p))\big)$. Thus, we want to prove $\mathbf{T}_{ss}(\mathbb{S}_{u1}(t_p)) \leq \mathbf{T}_{ss}(\mathbb{S}_{u2}(t_p))$. Assume the schedule starts from $\mathbf{T_0} = \mathbf{T}(0) = \mathbf{0}$, from (3.6), we have

$$\begin{cases} \mathbf{T}_{ss}(\mathbb{S}_{u1}(t_p)) = (\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(\mathbb{S}_{u1}(t_p)) \\ \mathbf{T}_{ss}(\mathbb{S}_{u2}(t_p)) = (\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(\mathbb{S}_{u2}(t_p)), \end{cases} \tag{4.2}$$

where $\mathbf{K} = e^{\mathbf{A}t_p}$. Since $(\mathbf{I} - \mathbf{K})^{-1}$ contains all positive elements [109], we only need to prove in the first period $\mathbf{T}(\mathbb{S}_{u1}(t_p)) \leq \mathbf{T}(\mathbb{S}_{u2}(t_p))$. Since $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$ are completely the same within $[t_0, t_{h-1}]$ and $[t_{h+1}, t_p]$, we only need to prove $\mathbf{T}(\mathbb{S}_{u1}(t_{h+1})) \leq \mathbf{T}(\mathbb{S}_{u2}(t_{h+1}))$.

Using the *superposition* principle, without loss of generality, we assume all cores, except for *core*$_i$, have no power consumptions. To ease the presentation, let $t_{h+1} -$

$t_{h-1} = 1$, $x = t_h - t_{h-1}$ and $1 - x = t_{h+1} - t_h$ ($0 \le x \le 1$), as shown in Fig. 4.2. From (3.4), then, we have

$$\begin{aligned}
\mathbf{T}(\mathbb{S}_{u1}(t_{h+1})) =& e^{\mathbf{A}}\mathbf{T}(\mathbb{S}_{u1}(t_{h-1})) + (\mathbf{I} - e^{\mathbf{A}})\mathbf{T}_e^\infty \\
\mathbf{T}(\mathbb{S}_{u2}(t_{h+1})) =& e^{\mathbf{A}}\mathbf{T}(\mathbb{S}_{u2}(t_{h-1})) + e^{\mathbf{A}(1-x)}(\mathbf{I} - e^{\mathbf{A}x})\mathbf{T}_L^\infty \\
& + (\mathbf{I} - e^{\mathbf{A}(1-x)})\mathbf{T}_H^\infty \\
=& e^{\mathbf{A}}\mathbf{T}(\mathbb{S}_{u2}(t_{h-1})) + (e^{\mathbf{A}(1-x)} - e^{\mathbf{A}})\mathbf{T}_L^\infty \\
& + (\mathbf{I} - e^{\mathbf{A}(1-x)})\mathbf{T}_H^\infty,
\end{aligned} \tag{4.3}$$

where $\mathbf{T}_e^\infty$, $\mathbf{T}_L^\infty$ and $\mathbf{T}_H^\infty$ are the constant temperature when $core_i$ runs in the mode with v_e, v_L and v_H long enough while all the other cores keep idle, respectively. Since $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$ are completely the same with $[t_0, t_{h-1}]$, we have $\mathbf{T}(\mathbb{S}_{u1}(t_{h-1})) = \mathbf{T}(\mathbb{S}_{u2}(t_{h-1}))$. Then, compare $\mathbf{T}(\mathbb{S}_{u2}(t_{h+1}))$ to $\mathbf{T}(\mathbb{S}_{u1}(t_{h+1}))$, we have

$$\begin{aligned}
& \mathbf{T}(\mathbb{S}_{u2}(t_{h+1})) - \mathbf{T}(\mathbb{S}_{u1}(t_{h+1})) \\
=& (\mathbf{I} - e^{\mathbf{A}}) \cdot \left[(\mathbf{I} - e^{\mathbf{A}})^{-1}(e^{\mathbf{A}(1-x)} - e^{\mathbf{A}})\mathbf{T}_L^\infty \right. \\
& \left. + (\mathbf{I} - e^{\mathbf{A}})^{-1}(\mathbf{I} - e^{\mathbf{A}(1-x)})\mathbf{T}_H^\infty - \mathbf{T}_e^\infty\right] \\
=& (\mathbf{I} - e^{\mathbf{A}}) \cdot [\boldsymbol{\rho}\mathbf{T}_L^\infty + (\mathbf{I} - \boldsymbol{\rho})\mathbf{T}_H^\infty - \mathbf{T}_e^\infty],
\end{aligned} \tag{4.4}$$

where $\boldsymbol{\rho} = (\mathbf{I} - e^{\mathbf{A}})^{-1}(e^{\mathbf{A}(1-x)} - e^{\mathbf{A}})$ and $\mathbf{I} - \boldsymbol{\rho} = (\mathbf{I} - e^{\mathbf{A}})^{-1}(\mathbf{I} - e^{\mathbf{A}(1-x)})$. According to (Lemma 3 in [52]), to prove $\mathbf{T}(\mathbb{S}_{u2}(t_{h+1})) - \mathbf{T}(\mathbb{S}_{u1}(t_{h+1})) \ge \mathbf{0}$, we need to prove $[\boldsymbol{\rho}\mathbf{T}_L^\infty + (\mathbf{I} - \boldsymbol{\rho})\mathbf{T}_H^\infty - \mathbf{T}_e^\infty] \ge \mathbf{0}$.

[Part 2] Consider an intermediate function $x\mathbf{T}_L^\infty + (1-x)\mathbf{T}_H^\infty$, we need to prove

$$\mathbf{T}_e^\infty \le x\mathbf{T}_L^\infty + (1-x)\mathbf{T}_H^\infty \le \boldsymbol{\rho}\mathbf{T}_L^\infty + (\mathbf{I} - \boldsymbol{\rho})\mathbf{T}_H^\infty \tag{4.5}$$

First, we prove $\mathbf{T}_e^\infty \le x\mathbf{T}_L^\infty + (1-x)\mathbf{T}_H^\infty$, in which $\mathbf{T}^\infty(\mathbf{v}) = -\mathbf{A}^{-1}\mathbf{C}^{-1}(\boldsymbol{\Psi}(\mathbf{v}) + \boldsymbol{\delta})$. Matrix $-\mathbf{A}^{-1}$ is constant, which contains all positive elements, because in practical scenarios, without changing any factor, increasing the power (voltage) of one node cannot decrease the temperature of other nodes. Moreover, since $\boldsymbol{\Psi}(\mathbf{v}) = [\psi(v_i)]_{N\times 1}$

and for each element $\psi_i(v_i) = \alpha + \gamma v_i^3$ is a convex function (α and γ are constants for a fixed v_i), $\mathbf{T}^{\infty}(\mathbf{v})$ is a convex function [14]. Therefore, given the condition in (4.1), we have $v_e^3 \leq x \cdot v_L^3 + (1-x) \cdot v_H^3$ and $\mathbf{T}_e^{\infty} \leq x \cdot \mathbf{T}_L^{\infty} + (1-x) \cdot \mathbf{T}_H^{\infty}$.

Second, we need to prove $x\mathbf{T}_L^{\infty} + (1-x)\mathbf{T}_H^{\infty} \leq \boldsymbol{\rho}\mathbf{T}_L^{\infty} + (\mathbf{I}-\boldsymbol{\rho})\mathbf{T}_H^{\infty}$, which can be sufficiently proved by

$$x \cdot \mathbf{I} \geq \boldsymbol{\rho}. \tag{4.6}$$

Since $\mathbf{A}$ is diagonalizable and all of its eigenvalues are negative [104], we can easily calculate its eigenvalues. Let $-\lambda_i$ be the *i-th* eigenvalue of $\mathbf{A}$ and $\lambda_i > 0$, we have $\mathbf{A} = \mathbf{W}\mathbf{D}\mathbf{W}^{-1}$, where $\mathbf{D} = diag\{-\lambda_1, \cdots, -\lambda_N\}$ and $\mathbf{W} = [\vec{w_1}, \cdots, \vec{w_N}]$. $\vec{w_i}$ is the independent eigenvectors associated with $-\lambda_i$. The matrix exponential of $e^{\mathbf{A}l}$ can be diagonalized as $e^{\mathbf{A}l} = \sum_{h=0}^{\infty} \frac{l^h(\mathbf{W}\mathbf{D}\mathbf{W}^{-1})^h}{h!} = \mathbf{W}\big(\sum_{h=0}^{\infty} \frac{l^h\mathbf{D}^h}{h!}\big)\mathbf{W}^{-1} = \mathbf{W}e^{\mathbf{D}l}\mathbf{W}^{-1}$, where $e^{\mathbf{D}l} = diag\{e^{-\lambda_1 l}, \cdots, e^{-\lambda_N l}\}$ and $e^{-\lambda_i l}$ is the i-th eigenvalue of $e^{\mathbf{A}l}$. Then, $\boldsymbol{\rho}$ can be diagonalized as

$$\boldsymbol{\rho} = (\mathbf{I} - e^{\mathbf{A}})^{-1}(e^{\mathbf{A}(1-x)} - e^{\mathbf{A}}) = \mathbf{W}diag\{\frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1 - e^{-\lambda_i}}\}\mathbf{W}^{-1} \tag{4.7}$$

To prove (4.6), that is to prove

$$\begin{aligned}
& x \cdot \mathbf{I} \geq \mathbf{W}diag\{\frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1 - e^{-\lambda_i}}\}\mathbf{W}^{-1} \\
\Longrightarrow & \mathbf{W}^{-1}x \cdot \mathbf{W} \geq \mathbf{W}^{-1}\mathbf{W}diag\{\frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1 - e^{-\lambda_i}}\}\mathbf{W}^{-1}\mathbf{W} \\
\Longrightarrow & x \cdot \mathbf{I} \geq diag\{\frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1 - e^{-\lambda_i}}\} \\
\Longrightarrow & x \geq \frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1 - e^{-\lambda_i}} \\
\Longrightarrow & \frac{1 - e^{-\lambda_i(1-x)}}{1 - e^{-\lambda_i}} - (1-x) \geq 0.
\end{aligned} \tag{4.8}$$

Consider function $\Upsilon(\varpi) = (1 - e^{-\lambda_i \varpi})(1 - e^{-\lambda_i})^{-1} - \varpi$, where $0 \leq \varpi \leq 1$ and $\lambda_i \geq 0$. Function $\Upsilon(\varpi)$ is a concave function because $\Upsilon''(\varpi) \leq 0$. In addition, function $\Upsilon(\varpi)$ passes two points, i.e. $(0,0)$ and $(1,0)$ when $\Upsilon(0) = 0$ and $\Upsilon(1) = 0$. Therefore, we have $\Upsilon(\varpi) \geq 0$ when $0 \leq \varpi \leq 1$. $\square$

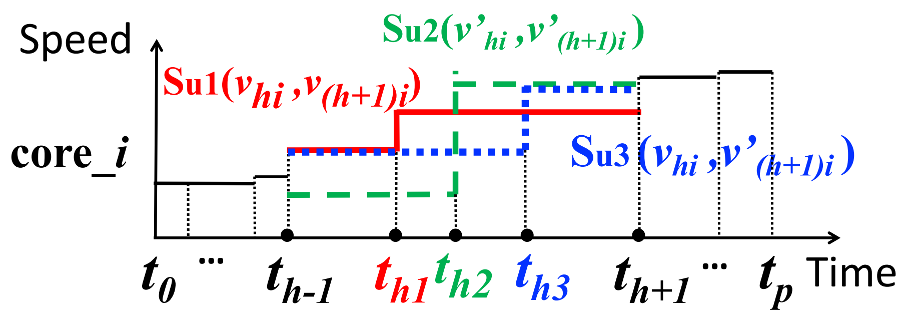

Speed
core_i
Su1(v_{hi}, $v_{(h+1)i}$)
Su2(v'_{hi}, $v'_{(h+1)i}$)
Su3(v_{hi}, $v'_{(h+1)i}$)
t_0
…
t_{h-1}
t_{h1}
t_{h2}
t_{h3}
t_{h+1}
…
t_p
Time

Figure 4.3: Illustration for Theorem 4.2.2.

Theorem 4.2.1 indicates that using a constant speed is more desirable and can result in a lower peak temperature than using two different speeds in a step-up schedule. Furthermore, we show that if we have to use two different speeds, then using two neighboring speeds is a better choice for lowing the peak temperature for a step-up schedule, as stated in the following theorem.

Theorem 4.2.2. *Let $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$ be two periodic step-up schedules that are exactly the same except for $core_i$ during interval $[t_{h-1}, t_{h+1}]$. Assume that in $\mathbb{S}_{u1}(t)$, $core_i$ uses two modes with voltages $v_{i,h}$ and $v_{i,(h+1)}$, while in $\mathbb{S}_{u2}(t)$, $core_i$ uses $v'_{i,h}$ and $v'_{i,(h+1)}$ such that (i) $core_i$ completes the same workload in both $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$; (ii) $v'_{i,h} \leq v_{i,h} \leq v_{i,(h+1)} \leq v'_{i,(h+1)}$. Then we have $T_{peak}(\mathbb{S}_{u1}(t)) \leq T_{peak}(\mathbb{S}_{u2}(t))$.*

Proof. As shown in Fig. 4.3, within interval $[t_{h-1}, t_{h+1}]$ on $core_i$ we define a third same throughput schedules $\mathbb{S}_{u3}(v_{i,h}, v'_{i,(h+1)})$ and let the h_{th} interval of $\mathbb{S}_{u1}(t)$, $\mathbb{S}_{u2}(t)$ and $\mathbb{S}_{u3}(t)$ change their modes at t_{h1}, t_{h2} and t_{h3}, respectively. Then, we have $t_{h1} \leq t_{h2} \leq t_{h3}$. Note that $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u3}(t)$ are of the same modes within interval $[0, t_{h1}]$; however, $\mathbb{S}_{u3}(t)$ uses two modes to complete the tasks within $[t_{h1}, t_{h+1}]$, while $\mathbb{S}_{u1}(t)$ use a constant mode. From Theorem 4.2.1, we can conclude that $\mathbf{T}(\mathbb{S}_{u3}(t_{h+1})) \geq \mathbf{T}(\mathbb{S}_{u1}(t_{h+1}))$. Then, in the following intervals, the temperature of $\mathbb{S}_{u3}(t)$ will always be higher than $\mathbb{S}_{u1}(t)$. Thus, we can conclude $T_{peak}(\mathbb{S}_{u3}(t)) \geq T_{peak}(\mathbb{S}_{u1}(t))$. Similar method can also be applied to prove $T_{peak}(\mathbb{S}_{u2}(t)) \geq T_{peak}(\mathbb{S}_{u3}(t))$. Therefore, $T_{peak}(\mathbb{S}_{u2}(t)) \geq T_{peak}(\mathbb{S}_{u1}(t))$. $\square$

Theorem 4.2.2 indicates that choosing two neighboring modes can achieve higher computational performances if the single mode with a constant supply voltage is not available when constructing a multi-core schedule.

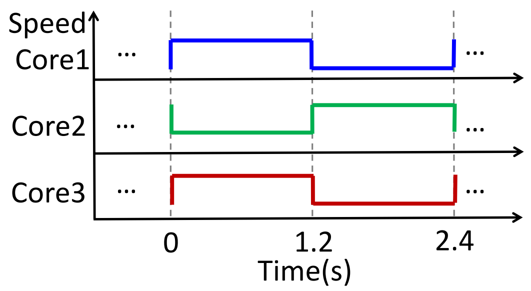
4.2.2 M-Oscillating Schedule on Multi-Core Platforms

With the method to bound the peak temperature for a periodic schedule and also with the guidelines to choose the running modes in such a schedule as introduced above, our next decision to make is to determine the length of the period for the periodic schedule. Recall that the motivation example seems to indicate that for a periodic two-speed step-up schedule, the smaller the period is, the higher the throughputs it can achieve.

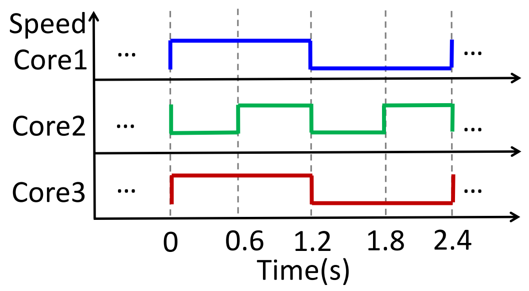
The m-Oscillating scheduling method, as introduced in [63] for single processor platform, is the scheduling method that frequently changes processor running modes between high and low voltage settings, while keeping the same workload within the same period to reduce the peak temperature. In what follows, we show that simply applying m-Oscillating scheduling method for each individual core on a multicore platform cannot always reduce the peak temperature.

Under the similar platform settings as shown in Section 3.3.2, we set up a schedule on a 3-core platform with the period of $2.4s$; each core runs at equal times in two processing modes, with high-voltage $v_H = 1.3V$ and low-voltage $v_L = 0.6V$, as shown in Fig. 4.4(a). Fig. 4.4(c) shows the stable status temperature trace within one period, with a peak temperature of 70.83°C. Next, we let core 2 double its oscillating frequency and core 1 and core 3 keep the same schedule, as shown in

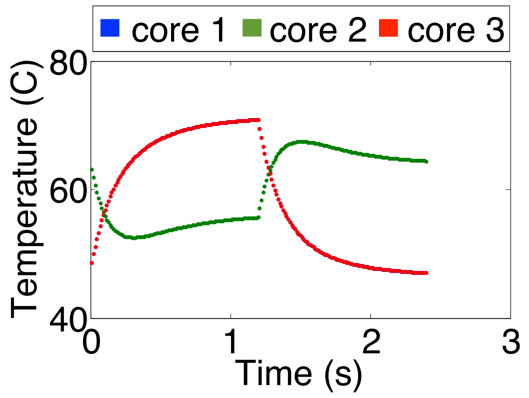
Fig. 4.4(b). In the stable status, as shown in Fig. 4.4(d), the peak temperature becomes 79.86°C, which is higher than the previous one. This example clearly shows



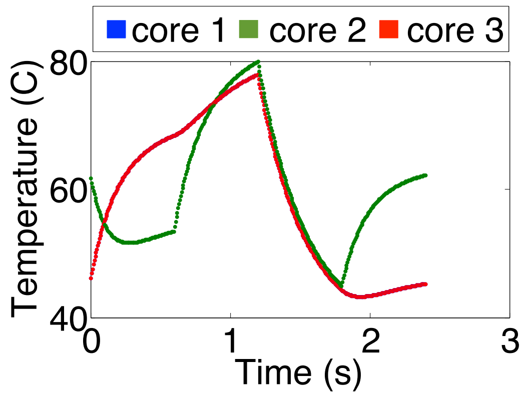
(a)



(b)



(c)



(d)

Figure 4.4: (a) Core 1 and core 3 run at $1.3V$ within $[0, 1.2]s$ and $0.6V$ within $[1.2, 2.4]s$; Core 2 runs at $0.6V$ within $[0, 1.2]s$ and $1.3V$ within $[1.2, 2.4]s$. (b) Core 2 doubles its oscillating frequency from schedule in Fig.4.4(a). (c) Stable status temperature trace for schedule in Fig. 4.4(a). (d) Stable status temperature trace for schedule in Fig. 4.4(b).

that the frequency oscillation scheme performed only on one core (asynchronized oscillation) does not necessarily reduce the peak temperature in a multi-core platform.

When all the cores oscillates their schedules in Figure 4.4(a) under a synchronized manner, as defined in Definition 4.2.3, the peak temperature monotonically decreases with the increase of the scaling factor m, as shown in Figure 4.5. In regard to this, we formally define the *m-Oscillating schedule for a multi-core platform* as follows.

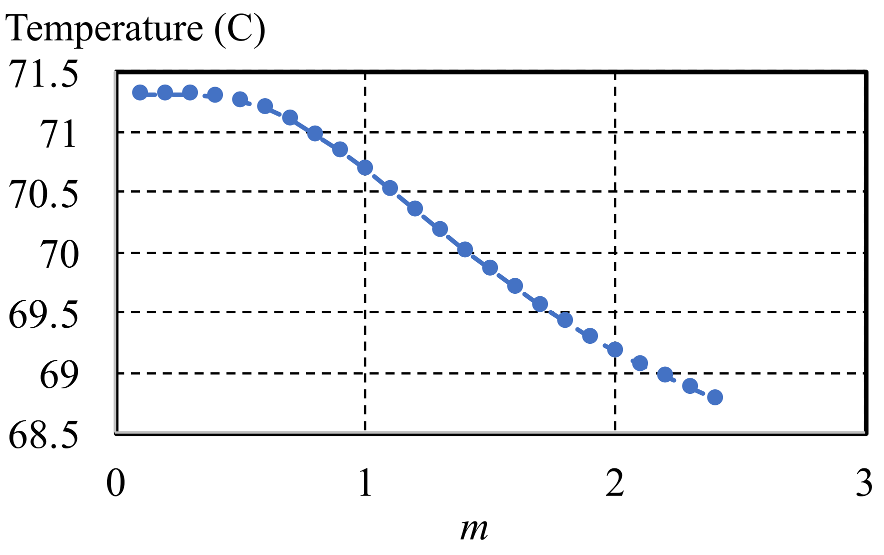


Figure 4.5: The peak temperature monotonically decreases with m.

Definition 4.2.3. *Let $\mathbb{S}(t)$ be a periodic schedule on a multi-core platform. The corresponding* **m-Oscillating schedule***, denoted as $\mathbb{S}(m,t)$, is the one that scales down the length of each state interval by m times without changing its running modes.*

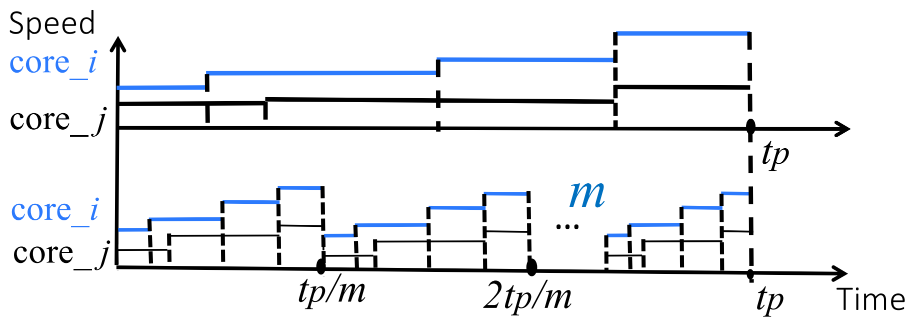


Figure 4.6: Illustration of m-Oscillating schedule.

Figure 4.6 shows that $\mathbb{S}(m,t)$ is derived from $\mathbb{S}(t)$ by scaling down each interval length by m times without changing the running modes.

For an m-Oscillating schedule with different m, the total dynamic energy consumption remains constant and the average temperatures within one period of m-Oscillating schedules do not change, as formulated in Lemma 4.2.4.

Lemma 4.2.4. *Let* $\mathbf{T}_{\mathbf{avg}}(\mathbb{S}(t))$ *and* $\mathbf{T}_{\mathbf{avg}}(\mathbb{S}(m,t))$ *denote the average temperature vector within one period of* $\mathbb{S}(t)$ *and* $\mathbb{S}(m,t)$*, respectively. Then,* $\mathbf{T}_{\mathbf{avg}}(\mathbb{S}(t)) = \mathbf{T}_{\mathbf{avg}}(\mathbb{S}(m,t))$.

Proof. Let $\mathbb{S}(t) = \{\mathbb{I}_q | q = 1 \cdots z\}$ be an arbitrary schedule contains z state intervals with period t_p and $\mathbb{S}(m,t)$ be the corresponding m-Oscillating schedule. According to [38], the average temperature of a state interval $\mathbb{I}_q = [t_{q-1}, t_q]$ is

$$\begin{aligned}\mathbf{T}_{\mathbf{avg}}(\mathbb{I}_q) &= \frac{1}{l_q}\int_{t_{q-1}}^{t_q}\mathbf{T}(t)dt \\ &= -\frac{1}{l_q}\mathbf{A}^{-1}[l_q\mathbf{C}^{-1}(\mathbf{\Psi}_q + \boldsymbol{\delta}) - (\mathbf{T}(t_q) - \mathbf{T}(t_{q-1}))]\end{aligned} \tag{4.9}$$

where $l_q = t_q - t_{q-1}$. Thus, the average temperature within one period of $\mathbb{S}(t)$ in the stable status is

$$\begin{aligned}\mathbf{T}_{\mathbf{avg}}(\mathbb{S}(t)) &= \frac{1}{t_p}\sum_{q=1}^{z}\int_{t_{q-1}}^{t_q}\mathbf{T}(t)dt \\ &= -\frac{1}{t_p}\sum_{q=1}^{z}\mathbf{A}^{-1}[l_q\mathbf{C}^{-1}(\mathbf{\Psi}_q + \boldsymbol{\delta}) - (\mathbf{T}(t_q) - \mathbf{T}(t_{q-1}))] \\ &= -\frac{1}{t_p}\mathbf{A}^{-1}[\sum_{q=1}^{z} l_q\mathbf{C}^{-1}(\mathbf{\Psi}_q + \boldsymbol{\delta}) - \sum_{q=1}^{z}(\mathbf{T}(t_q) - \mathbf{T}(t_{q-1}))]\end{aligned} \tag{4.10}$$

Since the former interval's ending temperature equals to the next interval's starting temperature, and the starting temperature equals to its ending temperature in the thermal stable status for one period. Thus, we have $\sum_{q=1}^{z}(\mathbf{T}(t_q) - \mathbf{T}(t_{q-1})) = \mathbf{0}$, so $\mathbf{T}_{\mathbf{avg}}(\mathbb{S}(t)) = -\frac{1}{t_p}\mathbf{A}^{-1}\mathbf{C}^{-1}\sum_{q=1}^{z} l_q(\mathbf{\Psi}_q + \boldsymbol{\delta}) = \frac{1}{t_p}\sum_{q=1}^{z} l_q * \mathbf{T}_q^{\infty}$.

In addition, the average temperature for schedule $\mathbb{S}(m,t)$ can be expressed as

$$\begin{aligned}\mathbf{T_{avg}}(\mathbb{S}(m,t)) &= \frac{1}{t_p/m}\sum_{q=1}^{z}\int_{t_{q-1}/m}^{t_q/m}\mathbf{T}(t)dt \\ &= -\frac{1}{t_p/m}\mathbf{A}^{-1}\mathbf{C}^{-1}\sum_{q=1}^{z}\frac{l_q}{m}(\boldsymbol{\Psi}_q+\boldsymbol{\delta}) \qquad (4.11)\\ &= -\frac{1}{t_p}\mathbf{A}^{-1}\mathbf{C}^{-1}\sum_{q=1}^{z}l_q(\boldsymbol{\Psi}_q+\boldsymbol{\delta}) = \frac{1}{t_p}\sum_{q=1}^{z}l_q * \mathbf{T}_q^{\infty}\end{aligned}$$

Thus, $\mathbf{T_{avg}}(\mathbb{S}(t)) = \mathbf{T_{avg}}(\mathbb{S}(m,t))$ and m-Oscillating does not change the average temperature in the stable status. □

With the help of Lemma 4.2.4, we can readily prove that as m increases, the peak temperature of an m-Oscillating schedule monotonically decreases.

Theorem 4.2.5. *Let $\mathbb{S}(t) = \{\mathbb{I}_q : q = 1\cdots z\}$ be a step-up schedule contains z state intervals and $\mathbb{S}(m,t)$ be the corresponding m-Oscillating schedule. Then $T_{peak}(\mathbb{S}(m,t)) \geq T_{peak}(\mathbb{S}(m+1,t))$.*

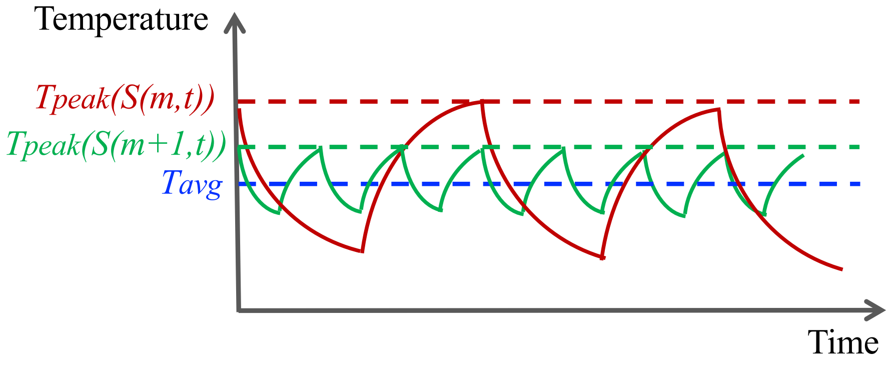


Figure 4.7: Illustration for Theorem 4.2.5 Proof.

Proof. Consider an arbitrary schedule $\mathbb{S}(t)$ with period t_p. According to Definition 4.2.3, $\mathbb{S}(m,t)$ is the corresponding m-oscillating schedule with period t_p/m. For each time point t/m of $\mathbb{S}(m,t)$, there exists a corresponding $t/(m+1)$ in schedule $\mathbb{S}(m+1,t)$.

Let $\mathbf{T_m}(t_p/m)$ and $\mathbf{T^{ss}_m}(t_p/m)$ be the ending temperature of $\mathbb{S}(m,t)$ in the first period and in the stable status, respectively. Assume the schedule starts at temperature $\mathbf{T_0} = \mathbf{0}$. Then, according to (3.5), for the first period of $\mathbb{S}(m,t)$, we have

$$\mathbf{T_m}(t_p/m) = \sum_{q=1}^{z} e^{\mathbf{A}\varpi/m}(\mathbf{I} - e^{\mathbf{A}l_q/m})\mathbf{T}^{\infty}_q, \tag{4.12}$$

where $\varpi = \sum_{\theta=q+1}^{z} l_\theta < t_p$. According to (3.6), in the stable status of $\mathbb{S}(m,t)$, we have

$$\begin{aligned}\mathbf{T^{ss}_m}(t_p/m) &= (\mathbf{I} - e^{\mathbf{A}t_p/m})^{-1}\mathbf{T_m}(t_p/m)\\ &= (\mathbf{I} - e^{\mathbf{A}t_p/m})^{-1}\sum_{q=1}^{z} e^{\mathbf{A}\varpi/m}(\mathbf{I} - e^{\mathbf{A}l_q/m})\mathbf{T}^{\infty}_q\\ &= \sum_{q=1}^{z} \dot{\mathbf{T}}^{\mathbf{ss}}_{\mathbf{m}}(t_p/m),\end{aligned} \tag{4.13}$$

where $\dot{\mathbf{T}}^{\mathbf{ss}}_{\mathbf{m}}(t_p/m) = (\mathbf{I} - e^{\mathbf{A}t_p/m})^{-1}e^{\mathbf{A}\varpi/m}(\mathbf{I} - e^{\mathbf{A}l_q/m})\mathbf{T}^{\infty}_q$.

To prove the peak temperature of $\mathbb{S}(m,t)$ is higher than the peak temperature of $\mathbb{S}(m+1,t)$, since the average temperature remains the same for different m value (Lemma 4.2.4), it is to prove the temperature varying range for $\mathbb{S}(m,t)$ is no smaller than $\mathbb{S}(m+1,t)$ in the stable status as shown in Fig. 4.7. Specifically, there are two cases as: (1) $\mathbf{T^{ss}_m}(t/m) \geq \mathbf{T^{ss}_{m+1}}(t/(m+1)) \geq \mathbf{T_{avg}}(\mathbb{S}(t))$, and (2) $\mathbf{T^{ss}_m}(t/m) \leq \mathbf{T^{ss}_{m+1}}(t/(m+1)) \leq \mathbf{T_{avg}}(\mathbb{S}(t))$, where $\mathbf{T_{avg}}(\mathbb{S}(t)) = \mathbf{T_{avg}}(\mathbb{S}(m,t)) = \mathbf{T_{avg}}(\mathbb{S}(m+1,t))$ and from Lemma 4.2.4,

$$\mathbf{T_{avg}}(\mathbb{S}(t)) = 1/t_p\sum_{q=1}^{z} l_q\mathbf{T}^{\infty}_q = \sum_{q=1}^{z}\dot{\mathbf{T}}_{\mathbf{avg}}(\mathbb{S}(t)), \tag{4.14}$$

where $\dot{\mathbf{T}}_{\mathbf{avg}}(\mathbb{S}(t)) = l_q/t_p \cdot \mathbf{T}^{\infty}_q$.

Essentially, it is to prove that the absolutely value of

$$|\mathbf{T^{ss}_m}(t_p/m) - \mathbf{T_{avg}}(\mathbb{S}(t))| = \sum_{q=1}^{z} |\dot{\mathbf{T}}^{\mathbf{ss}}_{\mathbf{m}}(t_p/m) - \dot{\mathbf{T}}_{\mathbf{avg}}(\mathbb{S}(t))| \tag{4.15}$$

monotonically decreases with m, which can be proved by showing for each q in (4.15), $|\dot{\mathbf{T}}^{\mathbf{ss}}_{\mathbf{m}}(t_p/m) - \dot{\mathbf{T}}_{\mathbf{avg}}(\mathbb{S}(t))|$ monotonically decreases with m.

Similar to the proof of Theorem 4.2.1, $\dot{\mathbf{T}}^{\mathbf{ss}}_{\mathbf{m}}(t_p/m)$ and $\dot{\mathbf{T}}_{\mathbf{avg}}(\mathbb{S}(t))$ can be diagonalized as

$$\begin{cases} \dot{\mathbf{T}}^{\mathbf{ss}}_{\mathbf{m}}(t_p/m) = \mathbf{W} diag\{\Upsilon_1(m), \cdots, \Upsilon_N(m)\}\mathbf{W}^{-1}\mathbf{T}^{\infty}_q, \\ \dot{\mathbf{T}}_{\mathbf{avg}}(\mathbb{S}(t)) = l_q/t_p * \mathbf{W}\ \mathbf{I}\ \mathbf{W}^{-1}\mathbf{T}^{\infty}_q \end{cases} \tag{4.16}$$

in which

$$\Upsilon_i(m) = \frac{e^{-\lambda_i \frac{\varpi}{m}}(1 - e^{-\lambda_i \frac{l_q}{m}})}{1 - e^{-\lambda_i \frac{t_p}{m}}}. \tag{4.17}$$

To prove $|\dot{\mathbf{T}}^{\mathbf{ss}}_{\mathbf{m}}(t_p/m) - \dot{\mathbf{T}}_{\mathbf{avg}}(\mathbb{S}(t))| = |\mathbf{W} diag\{|\Upsilon_i(m) - l_q/t_p|\}\mathbf{W}^{-1}\mathbf{T}^{\infty}_q|$ monotonically decreases with m, it is to prove $|\Upsilon_i(m) - l_q/t_p|$ monotonically decreases with m, in which $\mathbf{W}$, $\mathbf{W}^{-1}$ and $\mathbf{T}^{\infty}_q$ are constants.

We first prove $\Upsilon_i(m)$ monotonically decreases with m. Note that λ_i, ϖ l_q and t_p are constants. Since $e^{-\lambda_i \varpi/m}$ monotonically decreases with m, we then need to prove $\frac{1-e^{-\lambda_i l_q/m}}{1-e^{-\lambda_i t_p/m}}$ monotonically decreases with m too.

Let $F(\varphi) = \frac{1-e^{-\lambda_i l_q/m}}{1-e^{-\lambda_i t_p/m}} = \frac{1-e^{-\sigma_1 \varphi}}{1-e^{-\sigma_2 \varphi}}$, where $\sigma_1 = \lambda_i l_q$, $\sigma_2 = \lambda_i t_p$ and $\varphi = 1/m$. To prove $F(m)$ monotonically decrease with m, we need to prove $F(\varphi) = \frac{1-e^{-\varphi\sigma_1}}{1-e^{-\varphi\sigma_2}}$ monotonically increase with φ, where $\varphi > 0$ and $\sigma_2 > \sigma_1 > 0$, which is equivalent to prove

$$F'(\varphi) = \frac{(\sigma_2 - \sigma_1)e^{-(\sigma_1+\sigma_2)\varphi} + \sigma_1 e^{-\sigma_1\varphi} - \sigma_2 e^{-\sigma_2\varphi}}{(1 - e^{-\sigma_2\varphi})^2} > 0. \tag{4.18}$$

Let

$$\frac{F'(\varphi)}{F(\varphi)} = \frac{\sigma_1 e^{-\sigma_1\varphi}}{1 - e^{-\sigma_1\varphi}} - \frac{\sigma_2 e^{-\sigma_2\varphi}}{1 - e^{-\sigma_2\varphi}} = \xi(\sigma_1) - \xi(\sigma_2) \tag{4.19}$$

in which $\xi(\sigma) = \frac{\sigma e^{-\sigma\varphi}}{1-e^{-\sigma\varphi}}$. Since $F(\varphi) > 0$, we need to prove $\xi(\sigma)$ monotonically decreases with σ, i.e. $\xi'(\sigma) = \frac{e^{-\sigma\varphi}(1-\sigma\varphi-e^{-\sigma\varphi})}{(1-e^{-\sigma\varphi})^2} < 0$. Let $H(\varphi) = 1 - \sigma\varphi - e^{-\sigma\varphi}$, we can see $H(0) = 0$ and $H'(\varphi) = \sigma(-1 + e^{-\sigma\varphi}) < 0$ when $\varphi > 0$. Thus, we can infer $\xi'(\sigma) < 0$.

Then, since $\Upsilon_i(m)$ monotonically decreases with m and $\lim_{m\to\infty} \Upsilon_i(m) = 1$, we can infer that when $m \le \infty$, $\Upsilon_i(m) \ge 1$. In addition, since $l_q/t_p \le 1$ as defined in the schedule, $|\Upsilon_i(m) - l_q/t_p| = \Upsilon_i(m) - l_q/t_p \geq 0$ always holds. Thus, as $\Upsilon_i(m)$ monotonically decreases with m, $|\Upsilon_i(m) - l_q/t_p|$ also monotonically decreases. $\square$

As m continues to grow and eventually approaches infinity, the resulting peak temperature converges to a stable state temperature similar to that of using a single constant speed for each core. This observation is formulated in the following theorem.

Theorem 4.2.6. *Given a step-up schedule $\mathbb{S}(t) = \{\mathbb{I}_q | q = 1, \cdots, z\}$ contains z state intervals with period t_p, let state interval $\mathbb{I}_q = [t_{q-1}, t_q]$ and interval length $l_q = t_q - t_{q-1}$. Let $\mathbb{S}(m,t)$ be the m-Oscillating schedule of $\mathbb{S}(t)$. When $m \rightarrow \infty$, the peak temperature converge to the temperature of that running a constant speed profile $\mathbf{v_{eq}} = [v_{eq,i}]_{N\times 1}$, where $v_{eq,i} = \sqrt[3]{\sum_{q=1}^{z} \frac{l_q}{t_p} \cdot v_{q,i}^3}$.*

Proof. Since a *step-up schedule* is able to bound the peak temperature for any *arbitrary schedule* (Theorem 3.3.6), we prove that the peak temperature for the step-up schedule converges as $m \rightarrow \infty$. Based on (3.6), assume $\mathbf{T_0} = \mathbf{0}$, we have

$$\mathbf{T}_{ss}(t_p, m) = (\mathbf{I} - e^{\mathbf{A}t_p/m})^{-1}\mathbf{T}(t_p, m) \tag{4.20}$$

in which $\mathbf{T}(t_p, m) = \sum_{q=1}^{z} e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta/m}(\mathbf{I} - e^{\mathbf{A}l_q/m})\mathbf{T}_q^\infty$ from (3.5). Let $\mathbf{\Theta} = \mathbf{I} - e^{\mathbf{A}t_p/m}$. According to the L'Hospital's Rule [76], we have

$$\lim_{m\rightarrow\infty} \mathbf{T}_{ss}(t_p, m) = \lim_{m\rightarrow\infty} \mathbf{\Theta}^{-1}\mathbf{T}(t_p, m) = \lim_{m\rightarrow\infty} (\frac{d\mathbf{\Theta}}{dm})^{-1}\frac{d\mathbf{T}(t_p, m)}{dm} \tag{4.21}$$

in which

$$\begin{aligned} \frac{d\mathbf{T}(t_p, m)}{dm} &= \frac{1}{m^2}\mathbf{A}\sum_{q=1}^{z}\Big(\sum_{\theta=q}^{z} l_\theta \cdot e^{\mathbf{A}\sum_{\theta=q}^{z}\frac{l_\theta}{m}} - \sum_{\theta=q+1}^{z} l_\theta \cdot e^{\mathbf{A}\sum_{\theta=q+1}^{z}\frac{l_\theta}{m}}\Big)\mathbf{T}_q^\infty \\ \frac{d\mathbf{\Theta}}{dm} &= \frac{1}{m^2}\mathbf{A}t_p e^{\mathbf{A}\frac{t_p}{m}} \end{aligned} \tag{4.22}$$

Because $\lim_{m\to\infty} e^{\mathbf{A}l_\theta/m} = \mathbf{I}$, we have

$$\begin{aligned}
&\lim_{m\to\infty} \mathbf{T}_{ss}(t_p, m)\\
=&\lim_{m\to\infty} \frac{\sum_{q=1}^{z}(\sum_{\theta=q}^{z} l_\theta \cdot e^{\mathbf{A}\sum_{\theta=q}^{z}\frac{l_\theta}{m}} - \sum_{\theta=q+1}^{z} l_\theta \cdot e^{\mathbf{A}\sum_{\theta=q+1}^{z}\frac{l_\theta}{m}})\mathbf{T}_q^\infty}{t_p e^{\mathbf{A}\frac{t_p}{m}}} \qquad (4.23)\\
=&\frac{\sum_{q=1}^{z}(\sum_{\theta=q}^{z} l_\theta - \sum_{\theta=q+1}^{z} l_\theta)\mathbf{T}_q^\infty}{t_p} = \sum_{q=1}^{z}\frac{l_q}{t_p}\mathbf{T}_q^\infty
\end{aligned}$$

Thus, as $m \to \infty$ the temperature of m-Oscillating schedules converges to the temperature that equals the average temperature in Lemma 4.2.4.

Assume running a constant speed vector $\mathbf{v_{eq}}$ can reach the same temperature profile when $m \to \infty$ in (4.23). Then, from (3.2), we can solve $\mathbf{v_{eq}} = [v_{eq,i}]_{N\times 1}$ as

$$\begin{aligned}
&\mathbf{T}^\infty(\mathbf{v_{eq}}) = -\mathbf{A}^{-1}\mathbf{C}^{-1}(\boldsymbol{\Psi}(\mathbf{v_{eq}}) + \boldsymbol{\eta}) = \sum_{q=1}^{z}\frac{l_q}{t_p}\mathbf{T}_q^\infty\\
&\Rightarrow -\mathbf{A}^{-1}\mathbf{C}^{-1}(\boldsymbol{\Psi}(\mathbf{v_{eq}}) + \boldsymbol{\eta}) = -\mathbf{A}^{-1}\mathbf{C}^{-1}\sum_{q=1}^{z}\frac{l_q}{t_p}(\boldsymbol{\Psi}(\mathbf{v_q}) + \boldsymbol{\eta})\\
&\Rightarrow \boldsymbol{\Psi}(\mathbf{v_{eq}}) = \sum_{q=1}^{z}\frac{l_q}{t_p}\boldsymbol{\Psi}(\mathbf{v_q}) \qquad (4.24)\\
&\Rightarrow \alpha + \gamma v_{eq,i}^3 = \sum_{q=1}^{z}\frac{l_q}{t_p}(\alpha + \gamma v_{q,i}^3)\\
&\Rightarrow v_{eq,i} = \sqrt[3]{\sum_{q=1}^{z}\frac{l_q}{t_p}\cdot v_{q,i}^3}
\end{aligned}$$

$\square$

Theorem 4.2.5 indicates that oscillating the processing speeds can effectively reduce the peak temperature while completing the same throughput. Besides, the m-Oscillating schedule also helps to improve the real-time service capability of a schedule as formulated in the following theorem.

Theorem 4.2.7. *For a periodic schedule $\mathbb{S}(t)$, assuming $m_2 \geq m_1 > 0$, then the corresponding m-oscillating schedule $\mathbb{S}(m_2, t)$ is able to provide the service capability that is no lower than $\mathbb{S}(m_1, t)$.*

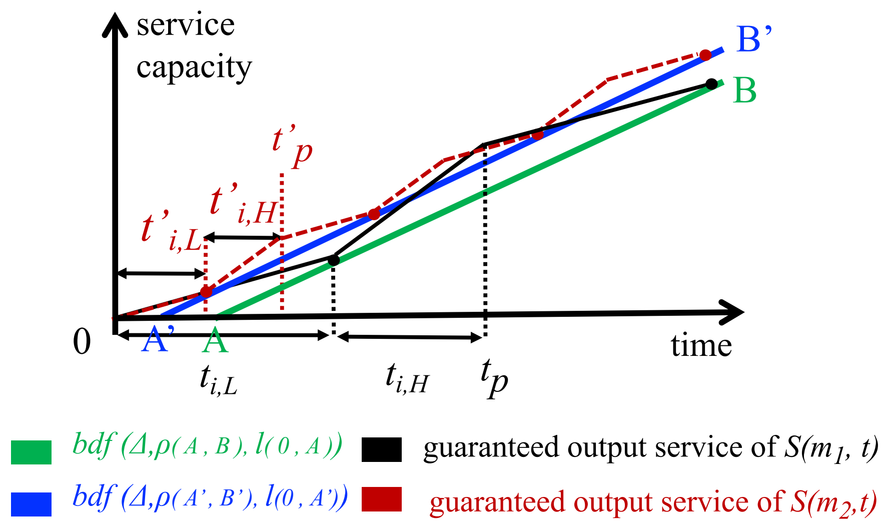

service capacity
B'
B
t'p
t'i,H
t'i,L
0
A'
A
time
ti,L
ti,H
tp
bdf (Δ,ρ(A, B), l(0, A))
bdf (Δ,ρ(A', B'), l(0, A'))
guaranteed output service of S(m1, t)
guaranteed output service of S(m2,t)

Figure 4.8: Real-time calculus illustration of Theorem 4.2.7: the bounded-delay approximation shows a higher service capacity of $\mathbb{S}(m_2, t)$ than $\mathbb{S}(m_1, t)$, if $m_2 \geq m_1 > 0$.

Proof. As shown in Figure 4.8, without losing generality, $\mathbb{S}(t)$ is assumed to be a two-speed schedule. Let $t_{i,H}$ and $t_{i,L}$ denotes the high-speed and low-speed length on the i-th node within one period t_p (as $t_{i,H} + t_{i,L} = t_p$) of $\mathbb{S}(t)$, respectively. Then, the bounded-delay function of the i-th node of $\mathbb{S}(m_1, t)$ is $bdf(\Delta, \rho(A, B), l(0, A))$ [65], which is defined by the slope $\rho(A, B)$ and the bounded-delay $l(0, A)$ (the distance between 0 and point A) for interval length Δ. Similarly, $bdf(\Delta, \rho(A', B'), l(0, A'))$ is the bounded-delay function of the i-th node for $\mathbb{S}(m_2, t)$.

To prove that $\mathbb{S}(m_2, t)$ is able to provide the service capability no lower than that of $\mathbb{S}(m_1, t)$, we aim to prove $l(0, A') \leq l(0, A)$ and $\rho(A', B') \nless \rho(A, B)$. Let the slopes for the high and low-speed interval of the service output trace $\mathfrak{B}^G(\Delta)$ be ρ_H and ρ_L, respectively. Then, for $\mathbb{S}(m_1, t)$, we can express the slope factor $\rho(A, B)$ and delay factor $l(0, A)$ as

$$\rho(A, B) = \frac{\rho_L t_{i,L}/m_1 + \rho_H t_{i,H}/m_1}{(t_{i,L} + t_{i,H})/m_1} = \frac{\rho_L t_{i,L} + \rho_H t_{i,H}}{t_{i,L} + t_{i,H}} \tag{4.25}$$

$$l(0, A) = \frac{(\rho_L - \rho_H) \cdot t_{i,H}/m_1 \cdot t_{i,L}/m_1}{(t_{i,H} + t_{i,L})/m_1} = \frac{(\rho_L - \rho_H) \cdot t_{i,H} \cdot t_{i,L}}{(t_{i,H} + t_{i,L})/m_1} \tag{4.26}$$

For $\mathbb{S}(m_2, t)$, we have

$$\rho(A', B') = \frac{\rho_L t_{i,L} + \rho_H t_{i,H}}{t_{i,L} + t_{i,H}} \tag{4.27}$$

$$l(0, A') = \frac{(\rho_L - \rho_H) \cdot t_{i,H} \cdot t_{i,L}}{(t_{i,H} + t_{i,L})/m_2} \tag{4.28}$$

It is not hard to see that $\rho(A, B) = \rho(A', B')$ and since $m_2 \geq m_1 > 0$, we have $l(0, A') \leq l(0, A)$. $\square$

Note that both Theorem 4.2.7 is established with the assumption that there is no energy/timing overhead when switching the running mode. In practical scenarios, the appropriate value of m needs to be judiciously selected, based on the actual systems' timing and energy overheads.

4.3 Throughput Maximization Using Frequency Oscillation

With the design principles and thermal characteristics presented above, we are now ready to introduce our approach to solve the multi-core throughput maximization problem.

The proposed approach contains three steps. First, we determine the ideal (continuously varied) speed for each core for throughput maximization under the peak temperature constraint. Then, the corresponding two neighboring discrete speeds are used to form a *step-up* schedule (Theorem 4.2.2), if the ideal speeds are not available. Then, we develop the m-Oscillating schedule accordingly to reduce the peak temperature (Theorem 4.2.5) with transition overhead taken into consideration. Finally, we adjust the high/low-speed execution time ratio to satisfy the peak temperature constraint. The detailed algorithm is illustrated in Algorithm 2.

As the starting point of our approach, we use a method similar to Hanumaiah et al. [56] to find the single constant mode (with $\mathbf{v}_{const}$) on each core to maximize the throughput. Specifically, we assume the stable state temperature for each core equals to T_{max}, i.e. $\mathbf{T}^{\infty}(\mathbf{v}_{const}) = [T_{max}]_{N\times 1}$. The power consumption, therefore, can be calculated by letting $\frac{d\mathbf{T}}{dt} = \mathbf{0}$ and $\mathbf{T} = [T_{max}]_{N\times 1}$ in (3.2), and the optimal voltage for each core can be calculated as $v_i = \sqrt[3]{(P_i - \alpha(v_i) - \beta T_{max})/\gamma(v_i)}$. With the knowledge of the single constant mode of v_i defined for the i_{th} core, the available high-voltage $v_{i,H}$ and low-voltage $v_{i,L}$ and their execution time ratios $r_{i,H}$ and $r_{i,L}$ that maintain the same throughput can be obtained by solving: (i) $v_{i,H} \cdot r_{i,H} + v_{i,L} \cdot r_{i,L} = v_i$; (ii) $r_{i,H} + r_{i,L} = 1$.

Next, to construct the m-Oscillating schedule, we need to find a proper m value to interleave the high/low-speed intervals to reduce the peak temperature (Theorem 4.2.5). However, in practical scenarios, each DVFS transition stalls the program execution for a small interval, which is unfavorable for the throughput maximization. Assume the program execution halted τ during the DVFS transition, then, each DVFS causes $(v_{i,H}+v_{i,L})\tau$ performance loss on $core_i$. To compensate the performance loss, in general, we shift a small interval of $\delta_i = \frac{(v_{i,H}+v_{i,L})\tau}{v_{i,H}-v_{i,L}}$ from low-speed to high-speed, as shown in Fig. 4.9(a). Meanwhile, the transition overhead introduces an upper bound $M_i = m_{i,max}(\tau)$, since the low-speed interval $t_{i,L}$ of $core_i$ should be long enough to cover the DVFS transition. The upper bound of m is $M_i = \lfloor \frac{t_{i,L}}{\delta_i+\tau} \rfloor$ on $core_i$. Since the peak temperature of a step-up schedule can be calculated with a linear complexity, the computational cost for searching m is affordable.

As we use two running modes instead of one in the m-Oscillating schedule, it may violate the peak temperature constraint, so it is necessary to adjust the high/low-speed ratio to lower down the peak temperature. First, we order the cores by their peak temperature, and the core with the highest peak temperature is selected to reduce its temperature. Note that, due to the heat transfer among cores, reducing the high-peed interval on any core can help to reduce its peak temperature. To find the core that can most effectively reduce the peak temperature (e.g. $core_i$), with the minimum throughput loss, we define a metric called *temperature performance trade-off index* for $core_i$, denoted as $\mathbf{TPT}_{core_i}$. Specifically, $\mathbf{TPT}_{core_i}(j) = \frac{\Delta T_i}{|v_{j,H} - v_{j,L}| \times t_{unit}}$ is the ratio of temperature reduction at $core_i$ to the throughput loss at $core_j$ when changing the high-voltage interval to the low-voltage interval for one unit of time, i.e. t_{unit}, on $core_j$. We iteratively modify the schedule for the core with the highest $\mathbf{TPT}_{core_i}$ until the temperature constraint is satisfied. The computational complexity of Algorithm 2 is $O(M + \frac{t_p}{t_{unit}} N)$.

Algorithm 2 Algorithm of m-Oscillating for throughput maximization under peak temperature constraints (**AO**).

```
 1: Input: Multi-core platform $\mathfrak{N} = \{core_i | i = 1 \cdots N\}$;
           Transition overhead parameters: $\tau$;
           $T_{max}$ and $T_{amb}$;
           Unit time: $t_{unit}$
 2: Output: The m-Oscillating schedule $\mathbb{S}(m^{opt}, t)$ and throughput $THR$ (equation 3.1)

 3: $m^{opt} = 1; M = m_{max}(\tau)$ // the largest possible value of $m$ for a given $\tau$
 4: Set $\mathbf{T}^{\infty}(\mathbf{v}) = [T_{max}]_{N \times 1}$ to find the constant voltage for each core, e.g. $v_i$ for $core_i$;
 5: for $1 \leq m \leq M$ do
 6:     Find modes (voltages) as well as their execution time ratios for each core, e.g. $v_{i,H}$,
        $r_{i,H}$, $v_{i,L}$ and $r_{i,L}$ for $core_i$ based on $v_i$ and $\tau$;
 7:     if ($T_{peak}(\mathbb{S}(m,t)) > T_{peak}(\mathbb{S}(m+1,t))$) then
 8:         $m^{opt} = m + 1$;
 9:     end if
10: end for
11: while ($T_{peak}(\mathbb{S}(m^{opt}, t) > T_{max}$) do
12:     Select $core_i$ = the core with the highest peak temperature;
13:     for $core_j \in \mathfrak{N}$ do
14:         $\mathbf{TPT}_{core_i}(j) = \frac{\Delta T_i}{|v_{j,H} - v_{j,L}| \times t_{unit}}$
15:     end for
16:     Select core $k$ = the core with the highest $\mathbf{TPT}_{core_i}(j)$;
17:     Reduce $v_{k,H}$ interval by one $t_{unit}$ and increase $v_{k,L}$ interval by one $t_{unit}$;
18: end while
```

Note that, in Algorithm 2, we require that each m-Oscillating schedule be a step-up schedule. This decision is really a double-edged sword. A step-up schedule allows us to quickly determine the highest temperature in a schedule to ensure that peak temperature constraint is guaranteed. In the meantime, however, since temperature varies with power density, the schedules that can interleave the intervals with high and low-voltage modes, temporally and spatially, lead to peak temperature lower than a step-up schedule. Accordingly we can design another algorithm that intends to distribute the workload more evenly temporally, as illustrated in Algorithm 3.

Algorithm 3 Phase-Conscious Oscillating (**PCO**).

```
1: Input: Multi-core platform $\mathfrak{N} = \{core_i | i = 1 \cdots N\}$;
          The m-Oscillating schedule $\mathbb{S}_{AO} = \mathbb{S}(m^{opt}, t)$ from Alg. 2;
          $T_{max}$ and $T_{amb}$;
          Unit time: $t_{unit}$;
2: Output: Phase-Conscious Oscillating $\mathbb{S}_{PCO}$ and $THR$.

3: Initialize $\mathbf{X} = \mathbf{0}$; // The phase vector $\mathbf{X} = [x_i]_{N\times 1}$.
4: for all possible $\mathbf{X} = [x_i]$ in which $x_i \in [0, t_p]$ do
5:     while $\exists\ T_{peak}(core_i) < T_{max}$, $core_i \in \mathfrak{N}$ do
6:         Select core $i$ = the core with the lowest peak temperature;
7:         Compute $\mathbf{TPT}_{core_i}(j)$ as Line 14 in Algorithm 2;
8:         Select core $k$ = the core with the lowest $\mathbf{TPT}_{core_i}(j)$;
9:         Reduce $v_{k,L}$ interval by one $t_{unit}$ and increase $v_{k,H}$ interval by one $t_{unit}$;
10:    end while
11:    Repeat Line 11 to Line 18 in Algorithm 2;
12:    Compute $THR$ for current schedule;
13: end for
14: Output $\mathbb{S}_{PCO}$ with the largest $THR$;
```

Specifically, Algorithm 3 iteratively constructs possible schedules that each core has a different starting instant for the high-speed intervals (namely *phase*), based on the resulted schedule from Algorithm 2. Then, for a fixed phase vector **X**, to maximize the throughput under a peak temperature constraint, Algorithm 3 first chooses the core with the lowest **TPT** index to extend its high-speed interval length, with the minimum overall peak temperature increment, as shown in Line 5 to Line 10. Then, to ensure the peak temperature, we have to judiciously cut off the high-speed with the minimum throughput loss, which is the same as Line 11 to 18 in Algorithm 2. To measure the peak temperature, it requires to check each time instant with step size of t_{unit}, with the complexity of $O(\frac{t_p}{t_{unit}})$. At last, we output the schedule with the largest *THR* as the resulted $\mathbb{S}_{PCO}$. The overall complexity is $O((\frac{t_p}{t_{unit}})^N \cdot \frac{t_p}{t_{unit}} \cdot \frac{t_p}{t_{unit}} N)$.

4.4 Experimental Results

In this section, we use experiments to test the effectiveness of the proposed *M-Oscillating* methodology in peak temperature minimization and throughput maximization. The power and thermal models adopted in this experiment are the same as Chapter 3.

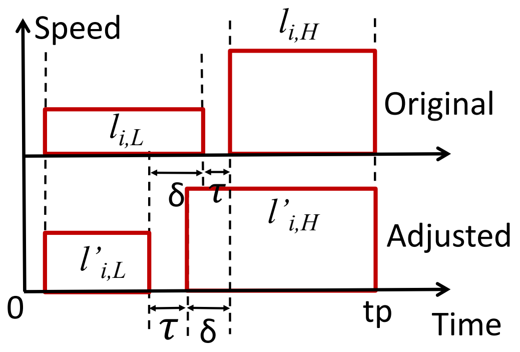
4.4.1 Peak Temperature Minimization for m-Oscillating Schedule

We verified Theorem 4.2.5 by using the schedule depicted in Fig. 3.5(a) as the original schedule. Then, we adopted Definition 4.2.3 to construct the corresponding m-Oscillating schedule for each m and profile the peak temperature for each case. Table 4.2 shows that the peak temperature monotonically decreases when m increases, exactly as predicted by Theorem 4.2.5.

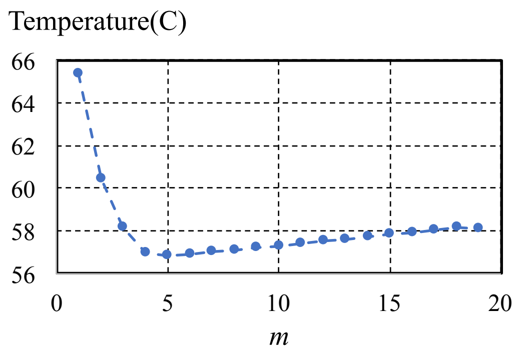
Table 4.2: Peak temperature T_{peak} (°C) monotonically decreases as m

m	**1**	**2**	**3**	**4**	**5**	**6**
T_{peak}	71.96	68.08	65.19	63.22	61.84	60.97
m	**7**	**8**	**9**	**10**	**11**	**12**
T_{peak}	60.79	60.65	60.53	60.44	60.36	60.30

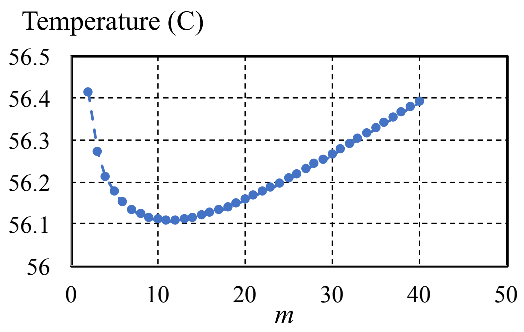
When considering the speed transition overhead as depicted in Section 4.3, we conducted another experiment by assuming different transition overheads. As shown in Fig. 4.9(b), Fig. 4.9(c) and Fig. 4.9(d), the transition overhead becomes smaller,



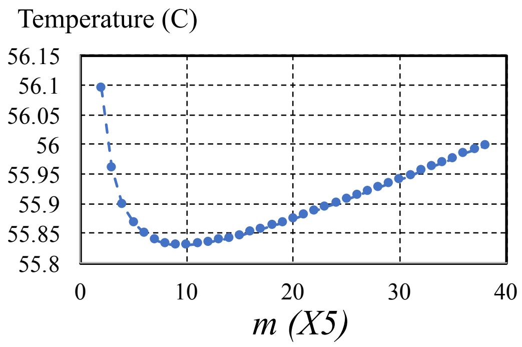
(a)



(b) $\tau = 5 \times 10^{-4}s$



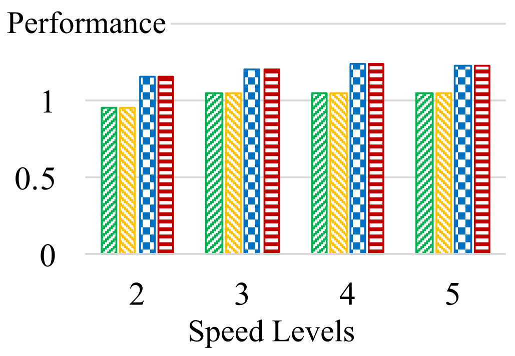
(c) $\tau = 5 \times 10^{-5}s$



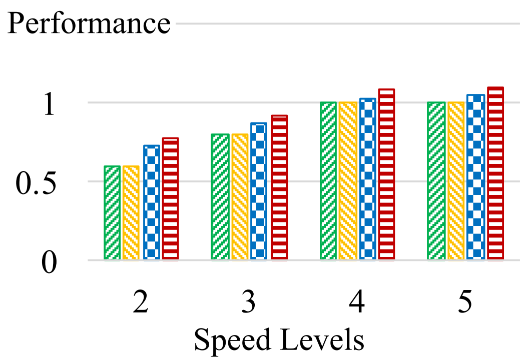
(d) $\tau = 5 \times 10^{-6}s$

Figure 4.9: (a) Speed adjustment on $core_i$ when consider speed transition overhead. (b)(c)(d) Peak temperature varies differently when transition overhead τs are different.

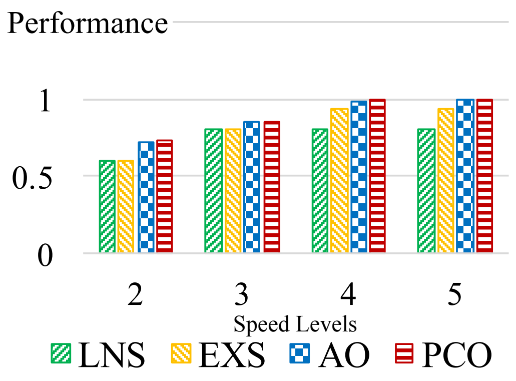
and the best m value to minimize the peak temperature is 5, 12 and 51, respectively. In general, a small transition overhead leads to a lower maximal temperature, because a small transition overhead results in a larger upper bound of m, by which it is able to fully exploit the speed transition for peak temperature minimization, and vice versa.



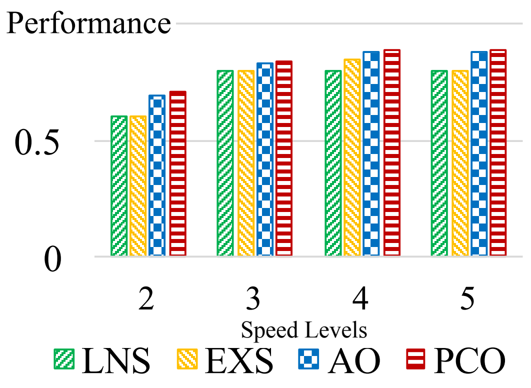
(a) 2-core



(b) 3-core

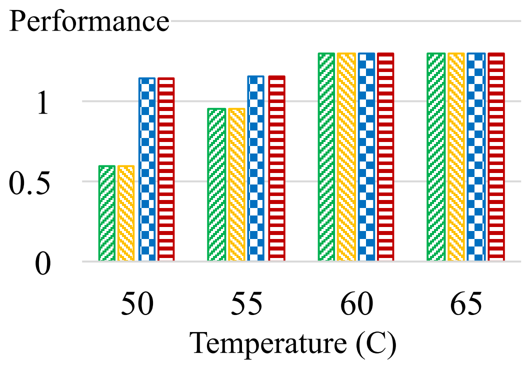


(c) 6-core

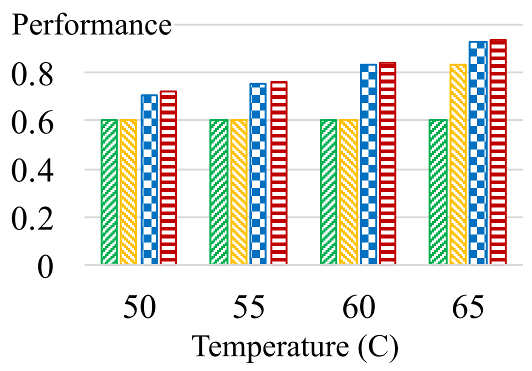


(d) 9-core

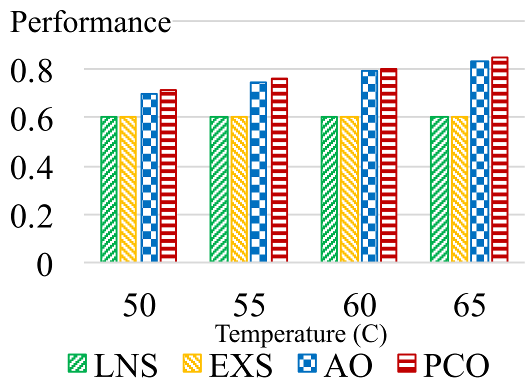
Figure 4.10: Performance comparisons with different numbers of cores and voltage levels, when $T_{max} = 55\,^{\circ}\mathrm{C}$.



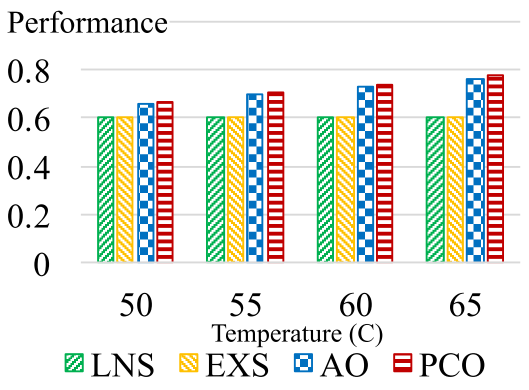
(a) 2-core



(b) 3-core



(c) 6-core



(d) 9-core

Figure 4.11: Performance comparisons with different numbers of cores and different T_{max} on 2 speed-level platforms.

Table 4.3: Different numbers of modes with different voltages.

Case	Voltage Level Selection
2 levels	$\{0.6V, 1.3V\}$
3 levels	$\{0.6V, 0.8V, 1.3V\}$
4 levels	$\{0.6V, 0.8V, 1.0V, 1.3V\}$
5 levels	$\{0.6V, 0.8V, 1.0V, 1.2V, 1.3V\}$

4.4.2 Performance Comparison of Different Approaches and Speed Levels

Next, we studied the performance of Algorithm 2. The experiment was conducted on different multi-core configurations. The RC-model was abstracted by matrix modeling method [145] from HotSpot-5.02 [131] and the transition overhead was set as $5us$. The maximum allowed temperature is $T_{max} = 55\,°\mathrm{C}$.

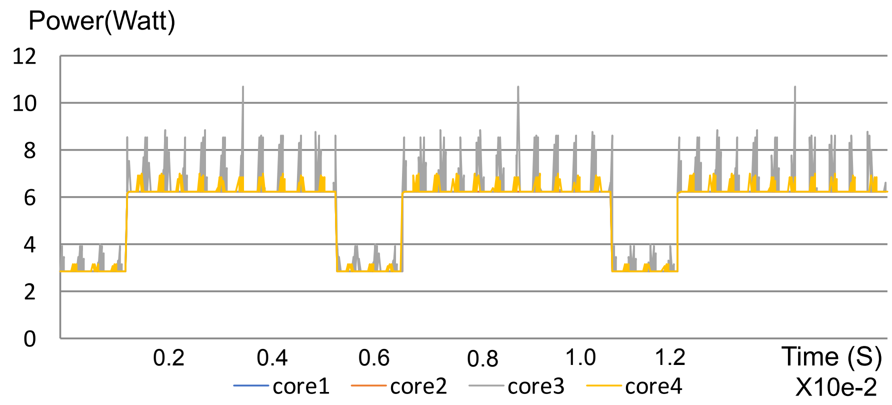
There are four approaches in this experiment: (1) *Lower neighboring speed* method (**LNS**) (i.e. choosing the lower neighboring speeds to guarantee the peak temperature constraint, when the continuous speeds are not available); (2) *Exhaustive search approach* (**EXS**) (as depicted as Algorithm 1); (3) *Aligned oscillation* (**AO**) is our proposed approach as shown in Algorithm 2; (4) The *phase-conscious oscillation* (**PCO**), with the periodic schedule obtained by shifting the initial starting time of the schedules obtained in **AO**, as introduced in Algorithm 3 in Section 4.3.

Fig. 4.10 compares the performances for different approaches on different number of cores (2, 3, 6, 9 cores) and different numbers of available speed levels (Table 4.3). The performance of **EXS** is better than **LNS**, because **EXS** checks all the possible speed combinations and has a deeper exploration of the design space than **LNS**. The proposed **AO** and **PCO** approaches always outperform **EXS** and **LNS**, especially when the number of available discrete speeds is small. The reason is that **LNS** and **EXS** are only allowed to use a single speed level for each core, which might be over pessimistic to the ideal ones. As shown in the figure, for 2 voltage levels, the average performance improvement by **AO** and **PCO** over **EXS** is 55.2%, and the improvement becomes 24.8% when the number of available voltage levels is 5.

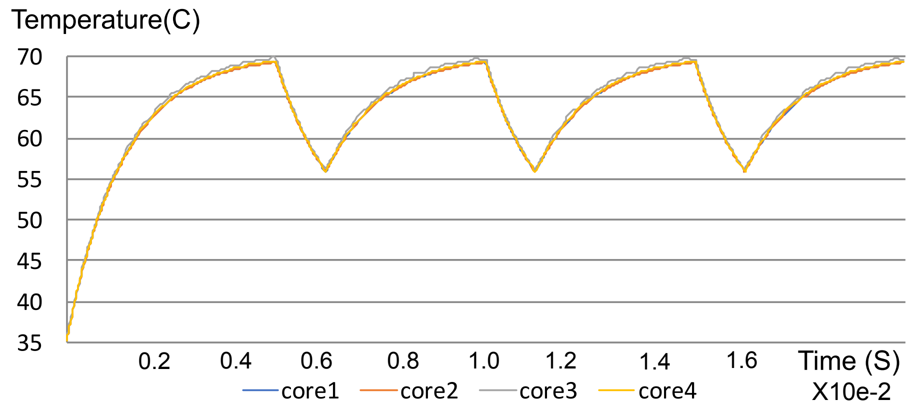
An interesting observation in this experiment is that the performances of **AO** and **PCO** are quite close. Even though our simulation study in section 3.4.1 shows large differences for schedules with different high-speed modes' starting time, this happens only when the period of the schedule is long, e.g. 6 seconds, for the results in section 3.4.1. As both **AO** and **PCO** adopt the m-Oscillating schemes, the scheduling periods are significantly reduced and, therefore, the differences become really insignificant, as shown in Fig. 4.10.

A similar conclusion can be drawn from our experimental results by changing the temperature threshold, as shown in Fig. 4.11. By varying the maximal allowed temperature T_{max} from 50 °C to 65 °C with 5 °C step size and two available speeds, we can see the throughput increases as the T_{max} increases. Note that all three approaches have the same performances on a 2-core platform when T_{max} is greater than 55 °C. The reason is that all the cores can use their highest speeds without violating the peak temperature constraint. For a 2-core platform in Fig. 4.11(a), when $T_{max} = 50\,°C$, **AO** and **PCO** have an improvement over **EXS** as high as 89.6%. For a 6-core platform in Fig. 4.11(c) when $T_{max} = 65\,°C$, **AO** and **PCO** have an improvement over **EXS** by 40.4%. For all the possible configurations with speed levels in Table 4.3, T_{max} from 50 °C to 65 °C with 5 °C step size, and on 2,3,6,9 core platforms, the average performance improvement of **AO** and **PCO** over **EXS** in is 11%.

We have also tested m-Oscillating apporach (**AO**) on a 4-core platform of a 2×2 topology, with application chosen from MiBench benchmark [50] and the



(a)



(b)

Figure 4.12: AO method on a 4-core platform with $T_{peak} = 70\,°\mathrm{C}$ (a) Power trace (b) Temperature trace .

power dynamics were abstracted from PTScalar. The temperature threshold is set to $T_{peak} = 70\,°\mathrm{C}$ with 2 speed levels. The workloads of *cjpeg*, *djpeg*, *h263* and *mpeg2* were partitioned in the order of core 1 to core 4. The high and low frequency operation were set to 3GHz and 1.8GHz, respectively. In Figure 4.12, the power and temperature have been abstracted with performance gain over **LNS** and **EXS** of 80% and 13.6%, respectively.

4.4.3 Computation Time Comparison

We then compare the computational costs of different approaches. Since **EXS** outperforms **LNS**, we select **EXS** to compare with **AO** and **PCO**. For each core configuration, we tested 2, 3, 4, 5 speed levels. For each setting, we run up to 100 cases and take the average CPU time to fill in Table 4.4.

In general, the computational cost increases as the number of cores or the available voltage level increases. When the design space is small, e.g. on 2 or 3 cores, the proposed **AO** and **PCO** take longer computation time than **EXS**. However, as the number of cores and the available speeds increases, the computational cost of **EXS** increases exponentially with the design space. For example, when searching 9 cores with 5 speed levels, **EXS** takes more than 2 hours, while **AO** only takes $1.55s$. In addition, the computational time of **PCO** is larger than **AO** because the **PCO** method needs to search the best *phase* as shown in Section 3.4.2 on different cores. Overall, the proposed **AO** method is computationally efficient in performance maximization problem under peak temperature constraints.

4.5 Conclusions

Due to the advancement of IC technology, high power density leads to high temperatures, which becomes a primary concern in design of high-performance systems. In this chapter, we developed a novel frequency oscillation-based technique to maximize the throughput performance of multi-core platforms under the maximally allowed temperature constraints. The proposed analytical approaches are built upon two concepts: *step-up schedule* and *m-Oscillating schedule*, and a number of well-formulated and proved theorems. The experimental results showed that our proposed method can effectively enhance the overall throughput by 11% on

Table 4.4: Computation time comparisons with different cores and voltage levels (Seconds).

	Scheme	**2 levels**	**3 levels**	**4 levels**	**5 levels**
2 cores	**AO**	0.13	0.08	0.08	0.16
	PCO	0.27	0.17	0.16	0.30
	EXS	0.01	0.01	0.01	0.01
	LNS	0.0008	0.0008	0.0008	0.0008
3 cores	**AO**	3.07	2.86	2.62	2.55
	PCO	9.11	25.94	40.97	19.41
	EXS	0.01	0.01	0.02	0.03
	LNS	0.001	0.0011	0.0013	0.0012
6 cores	**AO**	3.13	2.89	2.29	2.54
	PCO	27.76	31.32	31.68	43.08
	EXS	0.13	1.36	8.01	28.22
	LNS	0.0016	0.0015	0.0016	0.0016
9 cores	**AO**	2.59	1.96	1.81	1.55
	PCO	120.99	118.38	135.92	106.10
	EXS	1.53	43.36	581.14	>2-hours
	LNS	0.0025	0.0025	0.0023	0.0025

average with a reduced computational cost of orders of magnitudes compared to the traditional exhaustively search method. More important, the fundamental principles established in this chapter are also general to be applied for thermal-aware design on 2D, 3D multi-core systems.

CHAPTER 5

ENERGY REDUCTION ON MULTI-CORE PLATFORMS

In previous chapter, we present a frequency oscillating methodology to reduce the peak temperature and maximize the system throughput on multi-core platforms. In this chapter, we study the problem of how to improve the energy efficiency when scheduling a set of hard periodic real-time tasks on a multi-core platform under a given peak temperature constraint. The complexity of the multi-core energy reduction problem lies in the fact that multi-core energy consumptions relate to the task-to-core partitioning, tasks' execution speeds, the subset of active cores on multi-core platforms under the requirements of system throughput performance and temperature/power limitations.

To address the energy reduction problem, we first establish the theoretical upper bound for the energy efficiency by formulating the problem as a convex optimization problem. We then develop two heuristic approaches, i.e. *the leakage-aware load-balancing approach* and *the thermal-balancing approach*, and algorithms to bound the energy efficiency for these two approaches. Next, we transform the multi-core task partitioning problem to the bin packing problem. Specifically, we formulate our thermal-balancing approach as a variable sized bin-packing problem (VSBP) and develop a polynomial time task partitioning algorithm. We prove that our algorithm can obtain an approximation ratio of 3/2 over the optimal partitioning solution. Last, we develop an enhanced algorithm to continue improving the energy efficiency of the task partitions in our thermal-balancing approach.

The rest of this chapter is organized as follows. Section 5.1 presents the related work. Section 5.2 introduces preliminary and the problems. Section 5.3 and Section 5.4 discussed our proposed energy efficient algorithm. Experimental results are shown in Section 5.5, and Section 5.6 concludes the chapter.

5.1 Related Works

Energy minimization has long been a research problem that has been extensively studied. Due to the quadratic relationship between the dynamic power and processor speed, it has been a well-known principle to employ a constant processor speed and also lower the processor speed as much as possible to save energy consumption [148, 80]. This makes the workload balancing a good heuristic approach for energy minimization on multi-core or multi-computer platforms [69, 132]. As leakage power increases dramatically, there is a need to balance the dynamic and leakage power consumption, which leads to the approach of employing "the critical speed" [68, 70] to balance the dynamic energy reduction and leakage energy increment to minimize the overall energy consumption. There are many other works presented in the literature, but most of them, if not all, follow the same principles. However, these principles were established without considering thermal impacts.

Earlier work has been focused on dynamic energy conservation, which can be minimized by using the slowest constant executing speed either on single-core [148] or multi-core architectures [80]. As the IC industry enters the deep sub-micro domain, the leakage power becomes more and more prominent to the degree that is comparable or even surpasses the dynamic counterpart. Thus, monotonically decreasing the execution speed causes an increasing leakage energy consumption due to the extended completion time, which may increase the total energy consumption. Therefore, the idea of "the critical speed" [68, 70] has been developed to balance the dynamic power reduction and leakage power increment to minimize the overall energy consumption.

There are many other works presented in the literature [160, 139, 102, 9]. They differ by system models, design constraints, and computer architectures, but most of them, if not all, follow the same principles for energy optimization. For example, among some of these works, Chen et al. [24] proposed a greedy task mapping strategy for periodic task sets and proved a worst-case performance bound by searching the task allocation that each computing unit can be assigned more tasks with less energy consumption. Pagani et al. [103] proposed energy minimization for a set of periodic tasks assigned on different voltage islands using the lowest voltage/frequency that satisfied the timing constraints. Li et al. [80] proposed a Relaxation-Based Iterative Rounding Algorithm (RIRA) to minimize the energy consumption for non-preemptive tasks. However, the rounding-based mapping strategy may degrade the optimality in comparison with the ILP method in [24]. Lee et al. [79] explored the energy trade off when using the overabundant cores for parallel processing with a lower frequency, with the assumption that the tasks can split. Chen et al. [22] used mixed integrated linear programming (MILP) method to seek the optimal combination of DVFS and DPM for periodic-dependent tasks on multi-core platforms, but the complexity is too high. However, none of the approaches in [24, 80, 79, 22, 103] take the temperature constraint into consideration.

Nowadays, the exponentially increased transistor count in the IC chip has made the power density and heat dissipation a tremendous challenge in the design of computing systems, which exhibit even worse thermal impacts in 3D architectures. As the violation of the thermal constraint can automatically shut down the system for self-protection cooling purpose, it becomes necessary to consider the temperature constraint in design of real-time systems, e.g. [42, 2, 124, 78]. While energy consumption and temperature are closely related, as shown in [39], energy minimization and temperature reduction are not necessarily always in sync with each other. Due to the convex correlation between the running speed and the total energy consumption [54], an ideal energy-favored solution usually intends to let all the processing cores run at a uniformed speed. Since each core has a different heat removal path, they are thermally heterogeneous. Thus, taking the thermal factor into consideration along with energy awareness is crucial for the feasibility reason, especially when the system utilization is high or the temperature constraint is tight.

There are a few approaches on the temperature-constrained energy optimization problems. For example, when taking the peak temperature constraint into consideration, Saha et al. [117] proposed a genetic approach, essentially a meta-heuristic search algorithm, to minimize the energy under a pre-defined temperature threshold. Later, Hanumaiah et al. [54] formulated an integer linear programming method (ILP) for a task-to-core optimal assignment and fan speeds, to achieve the energy reduction under a given temperature constraint. They assumed that the peak temperature of a core must occur at a scheduling pint, which was not necessarily true, as indicated in the existing work (e.g. [39, 104, 124]). In addition, both [117] and [54] are computationally expensive as the design space becomes larger and/or when they are incorporated in other optimization loops. Barrefors et al. [11] formulated the task partitioning as a knapsack problem to minimize the energy under a thermal constraint. However, it ignored the heat transfer among cores and, thus, it is overly optimistic, especially for 3D-ICs. Zhou et al. [156] proposed an offline iterative approach to minimize the energy consumption when running real-time tasks on a heterogeneous multi-core platform under a temperature limit. The algorithm consists of two stages: the first stage of the algorithm intends to minimize the dynamic power consumption among the cores by allocating tasks to cores such that the overall dynamic energy consumption is minimized, and the second stage distributes the possible slack to tasks on each core in a way that the peak temperature is minimized. For homogeneous multi-core platforms, this approach is simply reduced to be the traditional *load-balancing* approach.

5.2 Preliminaries

We present the system model and formulate our research problem in this chapter. A similar multi-core model as Chapter 3 has been employed, i.e. multi-core platform $\mathfrak{N}$ has N cores and each core is DVFS independent. Each running mode is denoted by (v, f). By applying the power gating techniques, the idle cores (denoted by $\mathfrak{N}_{dark}$) without any task assignment can be shut down to avoid leakage power consumptions. Other active cores with task assignments belong to $\mathfrak{N}_{active}$, as $\mathfrak{N}_{active} = \mathfrak{N} \setminus \mathfrak{N}_{dark}$.

We assume a periodic task set with M tasks, $\mathbf{\Gamma} = \{\tau_1, \cdots, \tau_M\}$. Each task is defined by its inter-arrival time *(Period)* and the worst-case-execution-time ($WCET$) at the maximum speed, i.e. $\tau_k = \{Period_k, WCET_k\}$. Each task's deadline equals to its period. Since *earliest deadline first* (EDF) policy is optimal to schedule multiple periodic tasks on a core, in this chapter, we assume all tasks are scheduled by EDF policy.

5.2.1 Power/Thermal Model

Similar power model as Chapter 2 has been employed, the total power of the i-th core is

$$\begin{cases} P_i(t) = \alpha(v_i) + \beta \cdot T_i(t) + \gamma(v_i) \cdot v_i^3, & \textit{if } core_i \in \mathfrak{N}_{active}; \\ P_i(t) = 0, & \textit{if } core_i \in \mathfrak{N}_{dark}, \end{cases} \tag{5.1}$$

where α and γ are positive constants within the interval that $core_i$ runs at supply voltage v_i. β is a constant.

When employing the power model in Equation 5.1 to the aforementioned multi-core thermal model in Equation 3.2, the thermal dynamic is

$$\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\mathbf{T}(t) + \mathbf{C}^{-1}(\mathbf{\Psi}(\mathbf{v}) + \boldsymbol{\eta}), \tag{5.2}$$

where $\mathbf{\Psi}(\mathbf{v}) = [\alpha(v_i) + \gamma(v_i)v_i^3]_{N\times 1}$; $\boldsymbol{\eta} = [\frac{T_{amb}}{R_{ii}}]_{N\times 1}$ are constants, and R_{ii} is the thermal resistance of $core_i$ to itself. When running a multi-core processor under a constant supply voltage profile $\mathbf{v}$ long enough (i.e. $t \to \infty$), it will eventually reach a constant temperature $\mathbf{T}^{\infty} = \mathbf{T}(\infty) = -\mathbf{A}^{-1}\mathbf{C}^{-1}(\mathbf{\Psi}(\mathbf{v}) + \boldsymbol{\eta})$ as $\frac{d\mathbf{T}(\infty)}{dt} = \mathbf{0}$. ($\mathbf{A}$ is nonsingular [144] Lemma 1).

5.2.2 Energy Model

Consider a periodic schedule $\mathbb{S}(t) = \{\mathbb{I}_q : q = 1\cdots z\}$ with z *state intervals* in one hyperperiod $[t_0, t_p]$, which starts at t_0 and ends at t_p. The energy consumption vector of the q-th state-interval $\mathbb{I}_q = [t_{q-1}, t_q]$, which starts at t_{q-1} and ends at t_q, can be formulated as [39]

$$\mathbf{E}(t_{q-1}, t_q) = (\mathbf{I} - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{C}^{-1})l_q\mathbf{\Psi}_q - l_q\mathbf{\Phi}\mathbf{A}^{-1}\mathbf{C}^{-1}\boldsymbol{\eta} + \mathbf{\Phi}\mathbf{A}^{-1}[\mathbf{T}(t_q) - \mathbf{T}(t_{q-1})], \tag{5.3}$$

where $l_q = t_q - t_{q-1}$; $\mathbf{\Phi} = diag\{\beta\}_{N\times N}$; $\mathbf{\Psi}_q$ is the power-related factor of the q-th interval; $\mathbf{I}$ is an identity matrix. For interval $\mathbb{I}_q$, the total energy is $E_{total}(\mathbb{I}_q) = \sum E_i(t_{q-1}, t_q)$, in which $E_i(t_{q-1}, t_q)$ is the i-th entry of $\mathbf{E}(t_{q-1}, t_q)$.

When repeating $\mathbb{S}(t)$ long enough, the system enters its *thermal stable status*, with the starting temperature equals to the ending temperature in one period. Thus, the energy consumption in one period in the stable status is

$$\mathbf{E}_{ss}(t_0, t_p) =(\mathbf{I} - \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{C}^{-1})\sum_{q=1}^{z} l_q\mathbf{\Psi}_q - t_p\mathbf{\Phi}\mathbf{A}^{-1}\mathbf{C}^{-1}\boldsymbol{\eta}, \tag{5.4}$$

and the total energy consumption in $\mathbb{S}(t)$ is $E_{total}(t_0, t_p) = \sum_{\mathfrak{N}_{active}} \mathbf{E}_{ss,i}(t_0, t_p)$, where $\mathbf{E}_{ss,i}(t_0, t_p)$ is the i_{th} entry of $\mathbf{E}_{ss}(t_0, t_p)$

To evaluate the energy efficiency of a periodic schedule in the stable status, we adopt the concept of *"workload-per-Joule"* (WPJ) [54], which is defined as the ratio of the total completed workload and the total amount of energy consumed in one period. Specifically, for a schedule that contains z state intervals with period t_p, the energy efficiency is $WPJ = W/E_{ss}(t_0, t_p)$, where in one period, W denotes the total workload as $W = \sum_{q=1}^{z}\sum_{i=1}^{N} v_i l_q$. A larger *WPJ* value indicates that more workload can be completed with each unit of energy consumption, which means a better energy efficiency.

5.2.3 Problem Formulation

With the models introduced above, our problem can be formulated as follows.

Problem 5.2.1. *Given a periodic hard real-time task set* $\mathbf{\Gamma} = \{\tau_k | k = 1, \cdots, M\}$ *scheduled on a multi-core platform* $\mathfrak{N} = \{core_i | i = 1, \cdots, N\}$ *with maximum allowed temperature* (T_{max})*, find the task-to-core assignment matrix* ($\mathbf{\Theta}_{N\times M}$) *and the speed for each task* ($\mathbf{S}_{M\times 1}$)*, to maximize the overall energy efficiency (WPJ) in the thermal stable status.*

$$\begin{aligned} Max: \quad & WPJ\ ; \\ St: \quad & \sum_{k=1}^{M}\Theta_{i,k} = 1\ ; \\ & \mathbf{0} \le \mathbf{S}_{M\times 1} \le \mathbf{1}\ ; \\ & T_{peak} \le T_{max}\ ; \\ & \text{Utilization}_{corei} \le 1\ , \end{aligned} \tag{5.5}$$

where $\Theta_{i,k} = 1$, if task τ_k assigned to the *corei*; otherwise $\Theta_{i,k} = 0$; $\mathbf{S}_{M\times 1}$ is the speed vector for all the tasks; $Utilization_{corei}$ represent *corei*'s utilization. In this chapter, we consider the energy minimization problem for a periodic task set that runs long enough in the stable status.

5.3 Temperature-Constrained Energy Minimization on Multi-core Platforms

In this section, we first establish a theoretical lower bound for the energy consumption under a given temperature constraint on a multi-core platform. We then present two heuristics and study their energy efficiency potentials.

5.3.1 The Energy Consumption Lower Bound

The energy minimization problem can be formulated as a convex optimization problem [54], with the control variable as core-level processing speed to maximize the overall energy efficiency under the throughput requirements and the peak temperature constraint. Specifically, given a task set $\mathbf{\Gamma} = \{\tau_1, \cdots, \tau_M\}$, and an N-core platform $\mathfrak{N}$ with a temperature constraint T_{max}, the speed setting for each core that can lead to the minimum energy consumption can be found by solving the following convex optimization problem

$$Min: \quad \sum_{i=1}^{N} P_i, \qquad core_i \in \mathfrak{N}_{active} \ ; \tag{5.6a}$$

$$St: \quad \sum_{task\_k \in \mathbf{\Gamma}} \frac{ET_k}{Period_k} \leq \sum_{core\_i \in \mathfrak{N}_{active}} v_i \ ; \tag{5.6b}$$

$$T_i^{\infty} \leq T_{max} \ ; \qquad v_{min} \leq v_i \leq v_{max} \ ; \tag{5.6c}$$

While the above formulation can lead to the solution with optimal energy consumption, as the problem size (i.e. the number of cores and tasks) increases, the computational cost becomes extremely high. To this end, in what follows, we seek to reduce the computational complexity with two different heuristics, i.e. *"the leakage-aware load-balancing approach"* and *"the thermal-balancing approach"*.

5.3.2 The Leakage-Aware Load-Balancing Approach

Due to the convex correlation of the dynamic power and the processing speed, it is the most effective way to reduce the dynamic energy to balance the workload among multiple cores and use the processor speed as low as possible. Recall that the total energy consumption of an IC chip consists of both the dynamic and leakage part, and the leakage power consumption increases rapidly with the scaling of feature size to the degree that is comparable or even surpasses the dynamic power consumption [39]. While balancing workload among more cores can reduce core speeds and thus the dynamic energy consumption, the reduced dynamic energy consumption may not be able to offset the increase of the leakage power consumption for activating more cores. It is therefore a reasonable approach to make the appropriate tradeoff between the turning off of cores and reducing the core speeds when completing a given workload. In this regard, we can search for the proper subgroup of active cores and balance the workload among these cores in such a way that the temperature constraint can be satisfied and the *overall* energy consumption can be optimized. We call this approach the *Leakage-Aware Load-Balancing Approach* (LALB), as shown in Algorithm 4.

Algorithm 4 enumerates all the possible core configurations with different numbers of active cores. For each active core configuration, we can readily obtain the balanced workload for each core. Then, we can compute the corresponding WPJ index and choose the best solution that is feasible. When dealing with the discrete speed level cases, we can simply round up the speed to the upper neighboring level (after line 12 of Algorithm 4). Since there are totally N different numbers of active core scenarios, the complexity of Algorithm 4 is 2^N.

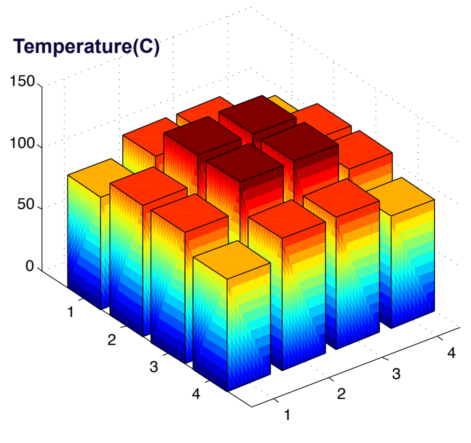
LALB explores all possible active core configurations and searches the optimal one that can balance the dynamic and leakage power consumption to achieve the

Algorithm 4 Leakage-aware load-balancing approach (**LALB**)

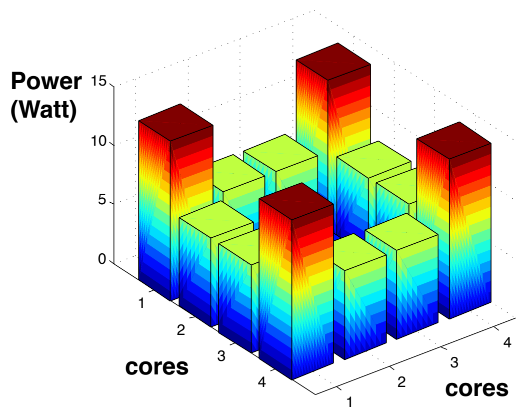
```
1: Input: Multi-core platform $\mathfrak{N} = \{core_i | i = 1 \cdots N\}$;
2:     Peak temperature constraint $T_{max}$;
3:     Total utilization $size(\mathbf{\Gamma}) = \sum_{\mathbf{\Gamma}} \frac{ET_k}{Period_k}$;
4: Output: Active core subset $\mathfrak{N}_{active}$, speeds $\mathbf{S}_{active}$;
5:     Overall energy-efficiency criteria WPJ;

6: for each possible $\mathfrak{N}_{active}$ with $1 \leq number(\mathfrak{N}_{active}) \leq N$ do
7:   $speed = size(\mathbf{\Gamma})/number(\mathfrak{N}_{active})$;
8:   if $speed < v_{min}$ then
9:     $speed = v_{min}$;
10:  else if $speed > v_{max}$ then
11:    return infeasible;
12:  end if
13:  Compute WPJ and check real-time/thermal feasibility;
14: end for
15: Output the highest WPJ solutions;
```

overall energy efficiency. It works well when temperature constraint is not a concern. However, balancing the workload among the active cores is not always a good choice to optimize the energy consumption under a given temperature constraint, especially when the temperature constraint is tight.

To better understand the limitation of the LALB approach, we first consider the thermal characteristics of multiple cores when all cores run at the same speed to complete the same workload. As shown in Figure 5.1(a), even though the workload is uniformly distributed among multiple cores, their temperatures are not uniform. By enforcing load balancing on the active cores, LALB can only choose the maximum speed such that the hottest core does not exceed its temperature threshold. This would result in activating more cores when the given temperature constraint is tight or system utilization is high and hence possibly degrade the energy efficiency. Under such scenarios, we believe that a thermal balanced approach can better utilize the temperature "head space" and achieve a better energy efficiency.



(a)



(b)

Figure 5.1: (a) Different cores exhibit different stable state temperature, when all cores are with the same amount of load. (b) Different cores have different maximal allowed power, when all the cores reach the temperature threshold contemporarily.

5.3.3 The Thermal-Balancing Approach

The LALB approach in Section 5.3.2 can be conservative by restricting all active cores to use a uniform execution speed. As explained before, this can lead to degraded energy efficiency when the temperature constraint is tight and/or the task set utilization is high, or some other factors (such as when only a few discrete speed levels are available.) Under such circumstance, we believe that a thermal balanced approach, as illustrated in Figure 5.1(b), which can adopt different processing speeds for different cores with different heat dissipation capability, can potentially better utilize the temperature head space to improve the energy efficiency. A necessary condition for thermal feasibility of executing a periodic task set on a multi-core platform has been developed in [5]; however, it did not take energy reduction into consideration. In what follows, we first formally define the concept of *"thermal-balancing state"* and show its interesting characteristics. We then introduce our proposed thermal-balancing algorithm for energy minimization on a multi-core platform under the given temperature constraint [125].

Definition 5.3.1. *Given a multi-core platform with n active cores, the multi-core platform achieves the thermal-balancing state at T_m, if all active cores maintain the same constant temperature T_m.*

When a multi-core platform achieves its thermal balance state, it can maximize the throughput under the given peak temperature constraint. This property is formulated in the following theorem.

Theorem 5.3.2. *Given a multi-core platform (n active cores) and the maximal allowed temperature of T_{max}, the overall throughput of the platform is maximized, if the multi-core platform achieves the thermal-balancing state at $T_m = T_{max}$.*

Proof. Let T_i^{∞} be the i-th element of $\mathbf{T}^{\infty}$ and $\mathscr{A}_{i,j}$ be the element of $-\mathbf{A}^{-1}$ on the position of the i-th row and j-th column. The problem depicted in Theorem 5.3.2 is

$$\begin{aligned} Max: \quad & \sum_{i=1}^{N} v_i, \qquad core_i \in \mathfrak{N}_{active}; \\ St: \quad & T_i^{\infty} = \sum_{j=1}^{N} \mathscr{A}_{i,j} C_j^{-1} (\alpha + \gamma v_j^3 + \frac{T_{amb}}{R_{jj}}); \\ & T_i^{\infty} \leq T_{max}; \\ & v_{min} \leq v_i \leq v_{max}\ ; \end{aligned} \tag{5.7}$$

Let $\xi_{1,i}$, $\xi_{2,i}$ and $\xi_{3,i}$ be the Lagrange multipliers associated with (5.7). The optimal solution to the linear problem by Karush-Kuhn-Tucker (KKT) optimality conditions [14] satisfies

$$\xi_{1,i}\big[\sum_{j=1}^{N} \mathscr{A}_{i,j} C_j^{-1} (\alpha + \gamma v_j^3 + \frac{T_{amb}}{R_{jj}}) - T_{max}\big] = 0 \tag{5.8}$$

$$\xi_{2,i}(v_i - v_{min}) = 0, \qquad \xi_{3,i}(v_{max} - v_i) = 0 \tag{5.9}$$

$$\xi_{1,i} \geq 0, \quad \xi_{2,i} \geq 0 \quad \text{and} \quad \xi_{3,i} \geq 0 \tag{5.10}$$

In addition, the Lagrangian function is

$$\mathcal{L}(v_i, \xi_{1,i}, \xi_{2,i}, \xi_{3,i}) = -\sum_{i=1}^{N} v_i + \xi_{1,i}\big[\sum_{j=1}^{N} \mathscr{A}_{i,j} C_j^{-1}(\alpha + \gamma v_j^3 + \frac{T_{amb}}{R_{jj}}) - T_{max}\big] \\ + \xi_{2,i}(v_i - v_{min}) + \xi_{3,i}(v_{max} - v_i) \tag{5.11}$$

Then, the supply voltage of the i-th core in the optimal solution should also satisfy

$$\frac{\partial \mathcal{L}}{\partial v_i} = -1 + \xi_{1,i}\mathscr{A}_{i,i}C_i^{-1}\gamma 3 v_i^2 + \xi_{2,i} - \xi_{3,i} = 0 \tag{5.12}$$

Consider in an optimal solution, the stable state temperature of the i-th core is lower than the temperature threshold, i.e. $T_i^{\infty} < T_{max}$, we can infer $\xi_{1,i} = 0$ from (5.8). So, (5.12) can be written as $-1 + \xi_{2,i} - \xi_{3,i} = 0$.

If not the case that all the active cores run at v_{max} the peak temperature still stays below T_{max}, there must be at least one core, e.g. the k-th core satisfies $T_k^{\infty} = T_{max}$, so $\xi_{1,k} \neq 0$. Further, we can infer $\xi_{2,k} = 0$, because $v_k \neq v_{min}$. Thus, the k-th core in the optimal solution should satisfy $-1 + \xi_{1,k}\mathscr{A}_{k,k}C_k^{-1}\gamma 3 v_k^2 - \xi_{3,k} = 0$ by (5.12).

Overall, to maximize the overall throughput, each active core should either run at the maximal speed or reach the temperature threshold. $\square$

As shown in Theorem 5.3.2, when a multi-core platform reaches the thermal-balancing state, its throughput is maximized for the given temperature constraint, which helps to reduce the number of active cores to minimize leakage energy consumption. Note that, under thermal-balancing state, even though all cores have the same temperature, their running speeds are different. To determine the speeds of active cores, we can use the following technique. Specifically, for each $core_i \in \mathfrak{N}_{active}$, let their stable state temperatures be uniformly defined as $T_i = T_m$. In the meantime, for $core_i \in \mathfrak{N}_{dark}$, we have $v_i = 0$. Note that, with given T_m and $\mathfrak{N}_{active}$ (and thus $\mathfrak{N}_{dark}$), the supply voltage v_i for each $core_i \in \mathfrak{N}_{active}$ and T_j for each $core_j \in \mathfrak{N}_{dark}$ are uniquely defined, which is formulated as follows.

Without losing generality, assume the first h cores are turned off and the rest n cores are activated, where $n + h = N$. We have [125] $\mathbf{T} = [\mathbf{T}_h, \mathbf{T}_n]_{N\times 1}$ and $\mathbf{\Psi} = [\mathbf{\Psi}_h, \mathbf{\Psi}_n]_{N\times 1}$, in which $\mathbf{T}_h = [T_1, \cdots, T_h]_{h\times 1}$; $\mathbf{T}_n = [T_m, \cdots, T_m]_{n\times 1}$; $\mathbf{\Psi}_h = [\alpha_0, \cdots, \alpha_0]_{h\times 1}$; $\mathbf{\Psi}_n = [\Psi_{h+1}, \cdots, \Psi_N]_{n\times 1}$. Since $\Psi_i = \alpha(v_i) + \gamma(v_i)v_i^3$, when $v_i = 0$, the power-related factor becomes a constant as $\Psi_i = \alpha(0) = \alpha_0$. Let $\mathbf{U} = -\mathbf{A}^{-1}\mathbf{C}^{-1}$ and $\mathbf{\Omega} = \mathbf{\Psi} + \boldsymbol{\eta}$. Then, according to Equation (5.2), we have

$$\begin{bmatrix} \mathbf{T}_h \\ \mathbf{T}_n \end{bmatrix} = \begin{bmatrix} \mathbf{U}_0 & \mathbf{U}_1 \\ \mathbf{U}_2 & \mathbf{U}_3 \end{bmatrix} \begin{bmatrix} \mathbf{\Omega}_h \\ \mathbf{\Omega}_n \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathbf{\Omega}_h \\ \mathbf{\Omega}_n \end{bmatrix} = \begin{bmatrix} \mathbf{\Psi}_h + \boldsymbol{\eta}_h \\ \mathbf{\Psi}_n + \boldsymbol{\eta}_n \end{bmatrix} \tag{5.13}$$

where $\boldsymbol{\eta}_h = [\eta_i]_{h\times 1}$ and $i = 1, \cdots, h$; $\boldsymbol{\eta}_n = [\eta_i]_{n\times 1}$ and $i = h+1, \cdots, N$. In Equation (5.13), the dimensions for $\mathbf{U}_0$, $\mathbf{U}_1$, $\mathbf{U}_2$ and $\mathbf{U}_3$ are $h \times h$, $h \times n$, $n \times h$ and $n \times n$, respectively. Note that, matrices/vectors $\mathbf{T}_n$, $\mathbf{U}_0$ to $\mathbf{U}_3$, $\mathbf{\Psi}_h$, $\boldsymbol{\eta}_h$ and $\boldsymbol{\eta}_n$ are determined once the power and thermal characteristics of the multi-core platform are given. Accordingly, $\mathbf{T}_h$ and $\mathbf{\Psi}_n$ can be solved as follows.

$$\begin{cases} \mathbf{T}_h = \mathbf{U}_0\mathbf{\Omega}_h + \mathbf{U}_1\mathbf{\Omega}_n \\ \mathbf{T}_n = \mathbf{U}_2\mathbf{\Omega}_h + \mathbf{U}_3\mathbf{\Omega}_n \end{cases} \Rightarrow \begin{cases} \mathbf{T}_h = \mathbf{U}_0\mathbf{\Omega}_h + \mathbf{U}_1\mathbf{\Omega}_n \\ \mathbf{U}_3\mathbf{\Omega}_n = \mathbf{T}_n - \mathbf{U}_2\mathbf{\Omega}_h \end{cases}$$
$$\Rightarrow \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_3 \end{bmatrix} \begin{bmatrix} \mathbf{T}_h \\ \mathbf{\Omega}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{U}_1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{T}_h \\ \mathbf{\Omega}_n \end{bmatrix} + \begin{bmatrix} \mathbf{U}_0\mathbf{\Omega}_h \\ \mathbf{T}_n - \mathbf{U}_2\mathbf{\Omega}_h \end{bmatrix} \tag{5.14}$$
$$\Rightarrow \begin{bmatrix} \mathbf{T}_h \\ \mathbf{\Omega}_n \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{U}_1 \\ \mathbf{0} & \mathbf{U}_3 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{U}_0\mathbf{\Omega}_h \\ \mathbf{T}_n - \mathbf{U}_2\mathbf{\Omega}_h \end{bmatrix}$$

After solving for $\mathbf{\Omega}_n$ and $\mathbf{\Psi}_n$, we can then obtain the supply voltage (v_i) for each active core, so that they can maintain their temperatures at T_m. The detailed algorithm is shown in Algorithm 5.

Algorithm 5 Thermal-balancing approach (**TB**)

```
1: **Input**: Multi-core platform $\mathfrak{N} = \{core_i | i = 1 \cdots N\}$;
2:       Peak temperature constraint $T_{max}$;
3:       Total utilization $\sum_{\mathbf{\Gamma}} \frac{ET_k}{Period_k}$, $task\ k \in \mathbf{\Gamma}$;
4: **Output**: Active core subset $\mathfrak{N}_{active}$, speeds $\mathbf{S}_{active}$;
5:       Overall energy-efficiency criteria WPJ;

6: **for** each possible $\mathfrak{N}_{active}$ with $1 \leq number(\mathfrak{N}_{active}) \leq N$ **do**
7:    Solve Eq. (5.13) and (5.14);
8:    Binary search the lowest $T_m \in [T_{amb}, T_{max}]$ that satisfy Eq. (5.6b);
9: **end for**
10: Output the highest WPJ solutions;
```

Algorithm 5 enumerates all the possible subsets of active core combinations. For each active core's topology, the lowest T_m and core speed are determined in line 7 and line 8. When there are only a limited number of discrete speeds available, we can round down the continuous speed value to the lower neighboring discrete one. Then, we can iteratively determine the core speed. The highest WPJ can thus be found by exhaustively searching different configurations (with different numbers of active cores, different active core combinations, and different thermal-balance temperatures) that satisfy the performance requirement.

On an N-core platform, for each active core subset, Algorithm 5 employs the binary search method to find the lowest thermal-balancing temperature (T_m) that can make the given task set feasible, with a complexity of $ln(T_{max} - T_{amb})$. Then, we iteratively determine the core-speed one by one from solving Eq. (5.13) and (5.14), so it needs 2^N iterations. Overall, the complexity of Algorithm 5 is $2^N \cdot N \cdot ln(T_{max} - T_{amb})$.

For both LALB in Algorithm 4 and TB in Algorithm 5, to reduce the complexity that enumerates 2^N subset of active core combinations, we can also adopt the patterning approach in [73] to balance the power density across the chip by activating different number of cores at different locations. The reduced complexity of LALB and TB are N and $N^2 \cdot ln(T_{max} - T_{amb})$, respectively.

5.4 Task Partitioning via Bin Packing Approaches

It is worthy of mentioning that both Algorithm 4 and Algorithm 5 assume that real-time tasks can be freely divided according to the processing capability of each core. Therefore, the outputs from Algorithm 4 and Algorithm 5 are in fact the upper bound for the energy efficiency. In reality, real-time tasks cannot be split arbitrarily, and mapping real-time tasks to multiple cores is itself an NP-hard problem [160].

One common heuristic approach for multi-core task partitioning is to transform it into a *bin packing* problem [30]. In the LALB approach, once the optimal choice of the group of active cores and their processing speeds are determined, the bin capacity is determined. Then, the problem is to pack objects (tasks), each of which has a different size (utilization), to bins such that the required bin is no more than the available one. Note that in LALB approach, each core is running at the same speed, which implies that all the available bins (active cores) have the same capacity. For the TB approach, however, different cores may have different speeds, we therefore need to pack tasks into a series of bins with different bin capacities.

5.4.1 Task Partitioning by Variable-Sized Bin Packing Approach

Consider the thermal constraint, the maximal allowed supply voltages across the multi-core platforms are different, which can be translated to each core having different "capacities," i.e. the maximum total task utilizations that can be accommodated in that core, as shown in Figure 5.1(b). To this end, we transform Problem 5.2.1 into the following variable-sized-bin-packing problem (VSBP) [30]. Then, the task partitioning problem can be transformed to a VSBP problem as follows:

Problem 5.4.1. *Given a set of objects* $\mathbf{\Gamma} = \{\tau_k | k = 1, \cdots, M\}$ *with each item size* $\frac{ET_k}{Period_k}$*, and a set of bins* $\mathfrak{N} = \{core_i | i = 1, \cdots, N\}$ *of capacities* $\{c_i | i = 1, \cdots, N\}$*, pack* $\mathbf{\Gamma}$ *into* $\mathfrak{N}$ *such that the total cost (proportional to the total bin size) is minimized.*

The optimization goal of Problem 5.4.1 is to minimize the total cost, actually the total active cores' capacities, used for packing the task set to the given platform, assuming the larger space used for packing the task set, the more energy the task set consumes.

Different from typical VSBP packing problems, such as the one in Section 4 of [43], assuming there are an unlimited number of bins for each bin type, we only have a limited number of bins for each type. Therefore, in our approach, we develop an iterative algorithm built upon the general principle of Algorithm A1 in [150], which has a proven approximation ratio of 3/2. Specifically, it clusters items into four size ranges as $(0, \frac{1}{3})$, $(\frac{1}{3}, \frac{1.5}{3})$, $(\frac{1.5}{3}, \frac{2}{3})$, $(\frac{2}{3}, 1)$; then, by matching different objects from different clusters, Algorithm A1 in [150] ensures that each allocated bin is at least 2/3 full or there must be a matching bin in the optimal solution using less capacity. Additional details for this approach can be found in [150]. Algorithm 6 depicts the details of our solution to Problem 5.4.1.

The rationale behind Algorithm 6 is assuming that the energy consumption is proportional to the total core-capacity (bin sizes), so we are seeking the lowest total core capacity that can hold the given task set. The feasible total core capacity must be larger than the total utilization of the task set but no greater than the maximal allowed throughput of the given platform. Since we adopt the Algorithm A1 [150], which can ensure that each bin can be filled at least 2/3 of its bin-capacity except

Algorithm 6 Variable-Sized-Bin-Packing **(VSBP)** method

```
1: Input: Multi-core platform $\mathfrak{N} = \{core_i | i = 1 \cdots N\}$;
2:       Peak temperature constraint $T_{max}$;
3:       Task set $\mathbf{\Gamma} = \{\tau_k | k = 1 \cdots M\}$, $\tau_k = \{\text{Period}_k, \text{ET}_k\}$;
4: Output: Task allocation matrix $\mathbf{\Theta}$;
5:       Task speeds vector $\mathbf{S}$;
6:       Overall energy-efficiency WPJ;

7: Solve Eq. (5.7) for the max throughput $THR_{max}$ under $T_{max}$;
8: $Ub = min\{THR_{max}, 3/2 * size(\mathbf{\Gamma})\}$;
9: $Lb = size(\mathbf{\Gamma})$;
10: while (1) do
11:    Determine active core topology/capacity by Algorithm 5 based on throughput requirement THR=(Ub+Lb)/2;
12:    for each core type in $\mathfrak{N}_{active}$ do
13:       Cluster and order core types decreasingly by capacities ;
14:       $N_\mu$ = The number of available cores in the $\mu$-th type;
15:       Packing tasks according to Algorithm A1 [150] assuming there is unlimited number of cores in this type;
16:       if succeed then
17:          Save workload assignment of first $N_\mu$ cores to $\mathbf{\Theta}$;
18:       else
19:          Return (Task set is not schedulable!);
20:       end if
21:       $\mathbf{\Gamma}' =$ workload in first $N_\mu$ cores of current type;
22:       $\mathbf{\Gamma} = \mathbf{\Gamma} - \mathbf{\Gamma}'$;
23:    end for
24:    Binary search the lowest $THR \geq size(\mathbf{\Gamma})$, break if $Ub - Lb \leq \epsilon$;
25: end while
26: Return the best WPJ solution;
```

for the last one, the upper bound of the searching range should be the larger value between maximal allowed throughput and the 3/2 times total core capacity. Specifically, in Algorithm 6, we first determine the system capacity that can maximize the feasibility and compute core capacities (line 11). The cores are then categorized to different types based on their capacities and sorted in a decreasing order (line 13). Then, we pack tasks to each core type using Algorithm A1 [150] by assuming an unlimited number of cores available in this type. This ensures that, except for the last core, each core is filled at least 2/3 full of its capacity, except the last one core of the last core type. Then, we save the task assignment for the first N_μ cores (line 17), due to the limitation of available cores of that type. Algorithm 6 iteratively improves the task partitioning results. At the end of each iteration, the lowest system throughput performance that can ensure the timing constraints for the tasks allocated to that core and the peak temperature based on the task partitioning results are searched.

For bin-packing approaches, the *absolute approximation ratio*, which is defined as the ratio of the number of bins produced by a heuristic over the minimum number of bins required to pack all the items, indicates the performance of a packing heuristic. It is not difficult to prove that Algorithm 6 has the following property.

Theorem 5.4.2. *Assuming there exists a feasible solution in Algorithm 6, the absolute approximation ratio of Algorithm 6 is 3/2, and the bound is tight.*

Proof. Consider there are total Ω different core types, the μ-th core type is represented by $\mathfrak{B}_\mu$ and $size(\mathfrak{B}_\mu)$ denotes the capacity. Let $cont(\mathfrak{B}_\mu)$ denote the contents that filled in one core of $\mathfrak{B}_\mu$ core type. Let $\mathcal{OPT}$ be the optimal packing fashion, i.e. all the used cores have been fully filled. We define $\mathcal{H}(\Gamma)$ as the total space used in Algorithm 6.

Assume Algorithm 6 successfully packs task set Γ in the first w core types ($w \leq \Omega$). The last core type uses ϖ cores ($\varpi < N_\mu$). For each consecutive core type, except for the last core of the last core type $\mathfrak{B}_w$, we have

$$\begin{aligned}
&cont.(\mathfrak{B}_1)N_1 \geq \frac{2}{3}size(\mathfrak{B}_1)N_1 \\
&\dots \quad \dots \quad \dots \quad \dots \\
&cont.(\mathfrak{B}_{w-1})N_{w-1} \geq \frac{2}{3}size(\mathfrak{B}_{w-1})N_{w-1} \\
&cont.(\mathfrak{B}_w)(\varpi - 1) \geq \frac{2}{3}size(\mathfrak{B}_w)(\varpi - 1)
\end{aligned} \tag{5.15}$$

Note that, besides the bins listed in (5.15), the contents in the last bin of the last bin type may not be larger than 2/3 of the bin size, i.e. there exists at most one bin that $cont.(\mathfrak{B}_w)$ may be smaller than $\frac{2}{3}size(\mathfrak{B}_w)$. Then, according to the definition, we have

$$\begin{aligned}
\mathcal{OPT}(\Gamma) &= \sum_{\mu=1}^{w-1} cont.(\mathfrak{B}_\mu)N_\mu + cont.(\mathfrak{B}_w)(\varpi - 1) + cont.(\mathfrak{B}_w) \\
\mathcal{H}(\Gamma) &= \sum_{\mu=1}^{w-1} size(\mathfrak{B}_\mu)N_\mu + size(\mathfrak{B}_w)(\varpi - 1) + size(\mathfrak{B}_w)
\end{aligned} \tag{5.16}$$

Then, we have

$$\begin{aligned}
\frac{\mathcal{OPT}(\Gamma)}{\mathcal{H}(\Gamma)} &= \frac{\sum_{\mu=1}^{w-1} cont.(\mathfrak{B}_\mu)N_\mu + cont.(\mathfrak{B}_w)(\varpi - 1) + cont.(\mathfrak{B}_w)}{\sum_{\mu=1}^{w-1} size(\mathfrak{B}_\mu)N_\mu + size(\mathfrak{B}_w)(\varpi - 1) + size(\mathfrak{B}_w)} \\
&\geq \frac{\sum_{\mu=1}^{w-1} 2/3size(\mathfrak{B}_\mu)N_\mu + 2/3size(\mathfrak{B}_w)(\varpi - 1) + 2/3size(\mathfrak{B}_w) + \Delta}{\sum_{\mu=1}^{w-1} size(\mathfrak{B}_\mu)N_\mu + size(\mathfrak{B}_w)(\varpi - 1) + size(\mathfrak{B}_w)} \\
&= 2/3 + \frac{\Delta}{\sum_{\mu=1}^{w-1} size(\mathfrak{B}_\mu)N_\mu + size(\mathfrak{B}_w)(\varpi - 1) + size(\mathfrak{B}_w)}
\end{aligned} \tag{5.17}$$

in which $\Delta = cont.(\mathfrak{B}_w) - 2/3 * size(\mathfrak{B}_w)$. Since for the last core, we have $cont.(\mathfrak{B}_w) \leq size(\mathfrak{B}_w) \leq min(\mathfrak{B}_1 \cdots \mathfrak{B}_w) \ll \sum_{\mu=1}^{w-1} size(\mathfrak{B}_\mu)N_\mu$, we can infer $\Delta/(\sum_{\mu=1}^{w-1} size(\mathfrak{B}_\mu)N_\mu + size(\mathfrak{B}_w)(\varpi - 1) + size(\mathfrak{B}_w)) \approx 0$. Thus, we have $\frac{\mathcal{OPT}(\Gamma)}{\mathcal{H}(\Gamma)} \geq 2/3$. □

5.4.2 The Enhanced Bin-Packing Method

A major drawback of Algorithm 6 is that it does not consider the task's characteristics when it determines the capacity (running speed) of a bin. For example, on a 3-core platform with $T_{max} = 50°\text{C}$, the maximum supply voltage for each core is determined as $[0.64, 0.51, 0.64]V$. Assume there is a task set including a task $\tau = \{Period = 10ms, ET = 7.5ms\}$ with utilization of 0.75. Item size 0.75 is larger than all the bin sizes, so Algorithm 6 cannot partition the task set no matter how small the total utilization of this task set can be. On the other hand, if we turn off both core 2 and core 3, the maximum voltage of core 1 becomes 0.81, and thus this task can be feasibly scheduled without violating the given thermal constraint. Therefore, judiciously choosing the active core sets based on task's utilization characteristics may help to improve the feasibility and energy efficiency performance when partitioning tasks.

Algorithm 7 Enhanced Varialbe-sized-bin-packing **(En-VSBP)** method

```
1: Input: $\mathfrak{N}$, $\mathbf{\Gamma}$, $T_{max}$;
2: Output: $\mathbf{\Theta}$, $\mathbf{S}$, WPJ;

3: $\mathbf{\Theta} = \emptyset$;
4: while $\mathbf{\Gamma} \neq \emptyset$ do
5:    Pack tasks by Algorithm 6;
6:    if fail then
7:       Move $\mathbf{\Gamma}'$ back to $\mathbf{\Gamma}$ in Algorithm 6 line 22;
8:       for each unpackable $\tau_k$ do
9:          Pack heavy task $\tau_k$;
10:         if failed then
11:            Return current best solution;
12:         else
13:            $\mathbf{\Gamma} = \mathbf{\Gamma} - \tau_k$;
14:         end if
15:      end for
16:   end if
17: end while
18: Return $\mathbf{\Theta}$, $\mathbf{S}$, WPJ;
```

Considering the limitation caused by "heavy tasks," we develop an Enhanced VSBP (En-VSBP) heuristic for the TB approach in Algorithm 7. One major difference between En-VSBP (Algorithm 7) and VSBP (Algorithm 6) is how to schedule *heavy task*, i.e. the task with utilization higher than any available utilization in any type. For VSBP, if a heavy task cannot fit in any type, it simply claims failure for the task partitioning. For En-VSBP, if a task cannot fit in any available type of core, we check if there is any idle core (with no task assignment) in the available active core set. If such a core does exist, we can turn other cores off (to be an inactive core), which potentially leads to a higher capacity for this core to accommodate the *heavy task*. Otherwise, there is no way we can assign the task without violating the peak temperature constraint (line 9).

The computational complexity of Algorithm 7 depends on how many iterations the algorithm needs to go through, which can be controlled using a threshold to limit the difference of peak temperatures for two consecutive iterations. Within each iteration, the complexity to compute core capacity is $O(N^2 \cdot ln(T_{max} - T_{amb}))$, the bin packing (line 8-15) has a complexity of $O(MN)$, and temperature calculation has a complexity of $O(N^3 * M)$. Therefore, the overall computational complexity is $O(N^3 * M)$.

5.5 Experimental Results

In this section, we first compare the energy efficiency, feasibility ratio and computational cost for the ideal cases, assuming all tasks can be arbitrarily split, in Section 5.5.1, Section 5.5.2 and Section 5.5.3, respectively. Then, for partitioning real-time tasks that are not arbitrarily divisible, we compare the energy efficiency and feasibility in Section 5.5.4.

The thermal and power parameters are abstracted from HotSpot 5.02 [66] and the McPAT simulator [82]. The ambient temperature is $T_{amb} = 35°C$, unless otherwise specified. There are four multi-core configurations: 2×3, 3×3, 3×4 and 4×4 corresponds to 6, 9, 12,16 cores, respectively. Each core size is $4 \times 4mm^2$ and

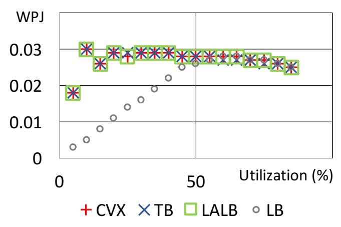
DVFS independent. In our experiments, we assumed the processing cores with either continuous variable speed between 0.6V to 1.3V or discrete speed levels, e.g. 3 levels as $\{0.6V, 0.95V, 1.3V\}$ or 5 levels as $\{0.6V, 0.775V, 0.95V, 1.125V, 1.3V\}$.

We first compare the energy consumption lower bound of four different herustics: (1) Convex solver-based approach (**CVX**) (see Section 5.3.1). (2) Leakage-aware load-balancing approach (**LALB**) (see Section 5.3.2). (3) Thermal-balancing approach (**TB**) (see Section 5.3.3). (4) Traditional load-balancing approach (**LB**), in which all the cores are turned on and running at a uniform speed. Specifically, assuming the speeds for processing cores are continuously variable, the formulation of **CVX** in Equation (5.6) is a disciplined convex program (DCP), which can be solved by the convex solver (CVX) [14]. When only a limited number of discrete supply voltages/speeds are available, this problem can be solved by mixed integer disciplined convex programs (MIDCPs) with MOSEK or Gurobi [47] package in a combination with the convex solver.

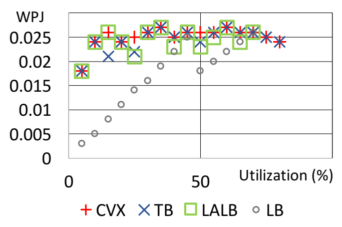
5.5.1 Lower Bound of Energy Efficiency (WPJ) Comparison

To compare the lower bound of energy consumption in Section **??**, we select different core configurations with different numbers of cores and different numbers of available discrete speed levels. For each method, the continuous speed mode bounds the WPJ value of discrete speed cases. Figure 5.2 shows the energy-efficiency comparison on 6, 9, 12 and 16-core, with continuous variable speeds, 3-speed-level and 5-speed-level scenarios.

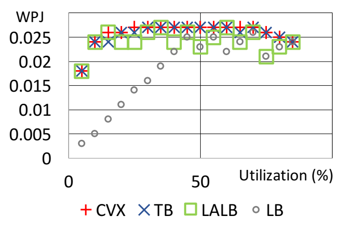
The lower bounds of the energy efficiencies are similar for different approaches, when the continuous speeds are available. For example, in the continuous variable speeds scenario in Figure 5.2(a), 5.2(d) 5.2(g) and 5.2(j), the energy efficiency



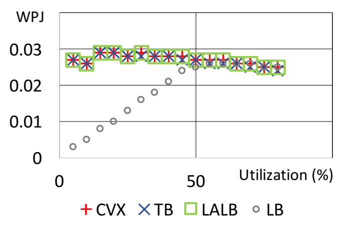
(a) 6-core, T_{max} = 65°C, continuous speed



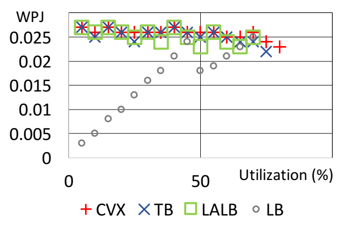
(b) 6-core, T_{max} = 65°C, 3 speeds



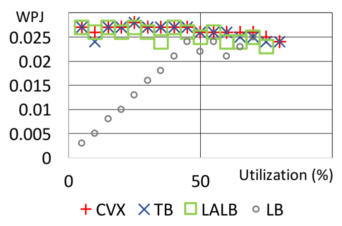
(c) 6-core, T_{max} = 65°C, 5 speeds



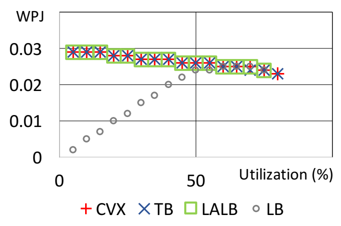
(d) 9-core, T_{max} = 75°C, continuous speed



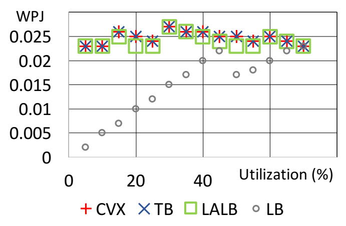
(e) 9-core, T_{max} = 75°C, 3 speeds



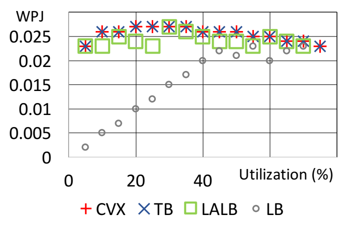
(f) 9-core, T_{max} = 75°C, 5 speeds



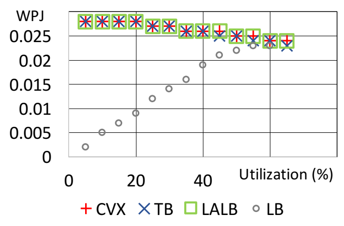
(g) 12-core, T_{max} = 85°C, continuous speed



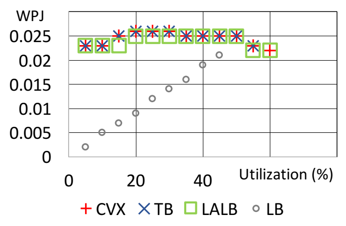
(h) 12-core, T_{max} = 85°C, 3 speeds



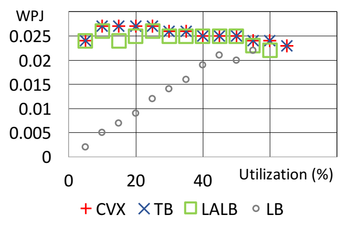
(i) 12-core, T_{max} = 85°C, 5 speeds



(j) 16-core, T_{max} = 85°C, continuous speed



(k) 16-core, T_{max} = 85°C, 3 speeds



(l) 16-core, T_{max} = 85°C, 5 speeds

Figure 5.2: WPJ comparison for different core configurations and different number of available speed levels

of **CVX**, **TB** and **LALB** are very close, because each method is very flexible to choose its own energy-favored voltages/speeds by selecting any continuous value in the valid range with respect to different algorithms.

However, when only a limited number of voltage levels/speeds are available, the energy efficiencies for different approaches become more obvious. Specifically, when the number of available voltages/speeds is small, **LALB** can be slightly better than **TB**. When the number of available voltage/speeds becomes larger, the energy efficiency of each approach grows quickly, and the **TB** approach benefits more from the increases of available voltage levels. For example, on a 3 discrete speed-level platform, as shown in Figure 5.2(b), 5.2(e) and 5.2(h), the average WPJ of **LALB** exceeds **TB** by 1.8%, 1.5% and 2.7%, respectively. When it increases to 16 cores with 3 discrete speed levels, the average WPJ of **TB** exceeds **LALB** by 2.2%, as shown in Figure 5.2(k). As more discrete speeds become available, e.g. 5 speed levels in Figure 5.2(c), 5.2(f) 5.2(i) and 5.2(l), the average WPJ of **TB** exceeds **LALB** by 4.6%, 1.4%, 5.8% and 4.1%, respectively. It is worth noting that the more discrete speeds are available, the higher the WPJ index will be for each method. More discrete speeds favor the **TB** method even more, because **TB** is more likely to achieve the thermal-balancing status. When less speed levels are available, **LALB** is better than the **TB** method, because **LALB** enumerates all the possible active core topologies to maximize the searching space. In addition, all the scenarios have shown that the **LB** method results in the lowest WPJ, especially when the system utilization is low. The reason is that the **LB** method requires all the cores be activated and at least running at their lowest speed, even though the workload is light.

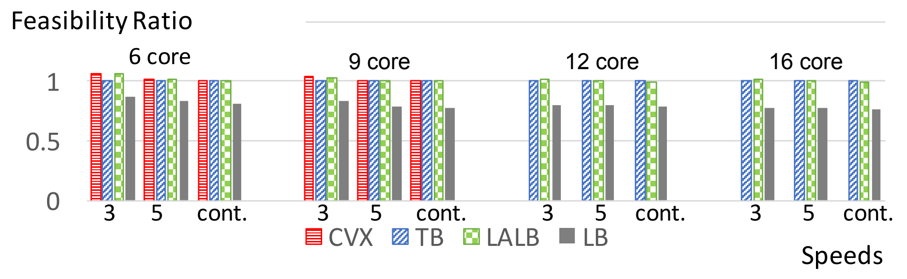
Overall, for each configuration, the proposed **CVX** method results in the optimal energy efficiency (WPJ). Our proposed **TB** slightly degrades from **CVX** results by

1.2%, and it is better than **LALB** by 1.8% on average based on a large number of random tests. In the meantime, there are significant differences in terms of system feasibility by different approaches, especially when the system utilization is high, as shown below.

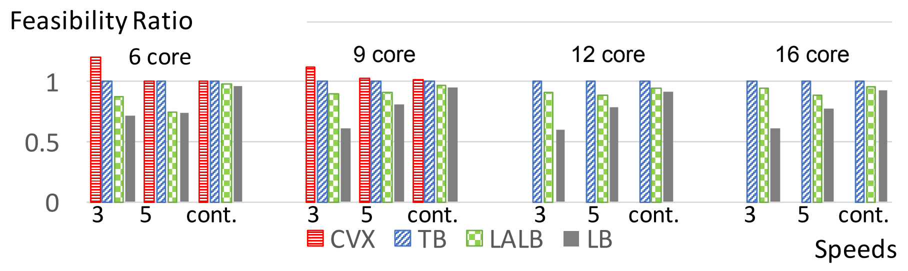
5.5.2 The Feasibility Comparison for Different Heuristics

In this section, we compare the feasibilities of different heuristics. For each method, we randomly generate up to 100 random cases and count the number of feasible cases. Then, we normalize the results to the **TB** result, as shown in Figure 5.3. To capture the feasibility characteristics by different workload requirements, we define the system utilization as the required throughput divided by the highest achievable throughput with all the processing cores run at their full speeds. Then, we conduct the experiments based on both low utilization and high utilization, which is defined as $0\% - 50\%$ and $50\% - 100\%$, respectively. We did not profile the feasibility ratio for 12 and 16-core cases in Figure 5.3, since their computation time is too long, as shown in Table 5.1 of Section 5.5.3.

When the system utilization is low, our proposed method **TB** shows a similar or slight degradation, when compared with the **CVX** and the **LALB** method. For example, in Figure 5.3(a) with system utilizations fall between 0 and 50%, the average feasibility ratio of **CVX**, **TB** and **LALB** are quite similar (102.1%, 100%, and 101.0%, respectively), which all outperform the **LB** method (80.0%) significantly. The reason is that **LB** always requires all the cores be activated and wastes a big portion of energy to execute at the minimum active speed, even though some redundant throughputs might be delivered. In some cases, e.g. 6-core and 9-core with 3 speeds, the **TB** method shows slight degradation from the **LALB**



(a)



(b)

Figure 5.3: Feasibility comparison when system utilization lies (a) between 0% and 50%; (b) between 50% and 100%

approach. The reason is that in Algorithm 5, we use a patterning approach to determine the active cores and iteratively determine the running speed for each core in line 7 to save the computational cost, which degrades the result's quality of **TB**.

When the system utilization is high, the **TB** approach exhibits a higher feasibility ratio than **LALB** and **LB**. For example, when system utilizations are between 50% and 100%, Figure 5.3(b) shows the feasibility ratio ranked as **CVX**> **TB**> **LALB**> **LB** (e.g. 105.9%, 100%, 90.9% and 78.7%, respectively). It is not difficult to understand that **CVX** has the highest feasibility because the convex solver provides the optimal solution within the validation range. The feasibility of **TB** exceeds **LALB** when the system utilization is high, because the "thermal-balancing" heuristic intends to maximize the system throughput under a given temperature constraint, as shown in Theorem 5.3.2. Therefore, when the peak temperature constraint is tight

Table 5.1: Computation time comparison (Seconds)

	6 core			**9 core**			**12 core**			**16 core**		
Speeds	**3**	**5**	**cont.**	**3**	**5**	**cont.**	**3**	**5**	**cont.**	**3**	**5**	**cont.**
CVX	28.03	27.87	27.77	220.14	219.85	216.84	$\approx 30min$	$\approx 30min$	$\approx 30min$	>5hours	>5hours	>5hours
TB	0.72	0.72	0.60	1.82	1.81	1.61	5.13	3.83	3.99	7.08	7.26	6.29
LALB	0.03	0.04	0.03	0.09	0.08	0.09	0.06	0.07	0.07	0.13	0.12	0.14
LB	0.003	0.001	0.001	0.002	0.002	0.002	0.05	0.03	0.01	0.13	0.13	0.13

or the system utilization is high, the **TB** method is still able to seek a valid solution. Overall, the feasibility of **TB** exceeds **LALB** by 9.14% and exceeds **LB** by 21.29%.

5.5.3 The Computational Time Comparison

We also compare the computational efficiency of different heuristics in Section **??**. Specifically, for different core configurations and numbers of available speeds, we randomly generate up to 100 cases under each configuration. From Table 5.1, we can see the the computational cost varies significantly with the number of cores, but does not change much with different number of speeds. For different approaches, the **CVX** method always needs the longest computation time and consumes approximately 30 minutes for 12-core platforms and more than 5 hours for 16-cores platforms. The **TB** method uses a polynomial computational time, as the design space becomes larger with the number of cores. number of tasks, etc. Although **LALB** and **LB** methods use a shorter time than **TB**, their average energy efficiencies and feasibilities are very poor.

5.5.4 Energy Efficiency (WPJ) and Feasibility When Packing Tasks

In this section, we compare the energy efficiency (WPJ) and feasibility when packing the actual tasks by thermal-balancing (**TB**), leakage-aware load-balancing (**LALB**) and traditional load-balancing (**LB**) approaches. Under each heuristic, two different packing methods are applied and compared in Algorithm 6 line 15: (1) First-Fit Decreasing (**FFD**) orderes tasks by their utilization before performing the first-fit packing; (2) **2/3-VSBP** represents Algorithm 6 line 15 with the 2/3 approximation heuristic. Thus, there are six combined approaches, including **LALB+FFD**, **LALB+2/3**, **TB+FFD**, **TB+2/3**, **LB+FFD**, **LB+2/3**. The last one **En-VSBP** is the Enhanced VSBP algorithm, as illustrated in Algorithm 7, which is built upon the thermal-balancing (**TB**) heuristic and the **2/3-VSBP** bin packing approach. The experiment runs on a randomly selected number of cores, number of tasks, number of speed levels, system utilizations and peak temperature constraints for 100 times.

First, we evaluate the energy efficiency (WPJ) by profiling the results that all the methods are feasible, as shown in Figure 5.4. It is worth noting that different packing heuristics do not influence on the energy efficiency much, and thus, **FFD** and **2/3-VSBP**-based packing methods have similar energy efficiency results. However, the heuristics for the bin-size determination plays an important role in energy-saving purposes. For example, the experimental results show that **LALB** and **TB**-based methods have similar energy efficiency (WPJ), which is higher than the **LB**-based methods by 8.5% and 9.4%, respectively. The reason is that **LB**-based methods fail to consider the energy savings from turning off redundant cores, so **LB**-based methods' energy efficiency is extremely low, which conforms to the results in Section 5.5.1. The energy efficiency of **En-VSBP** is the same as the **TB**-based approach, because **En-VSBP** determines the bin capacity based on the **TB** heuristic. In the meantime, different bin-size determination methods exhibit very different feasibility ratios as shown below.

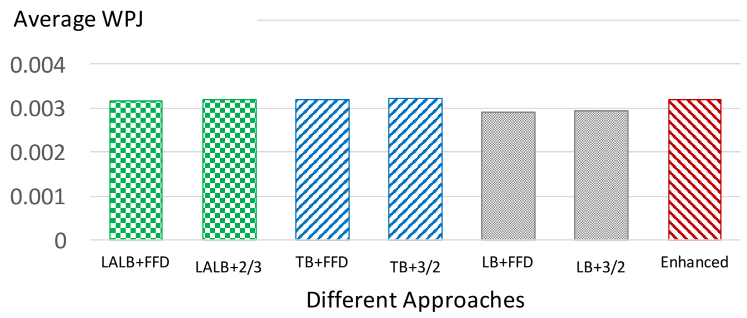
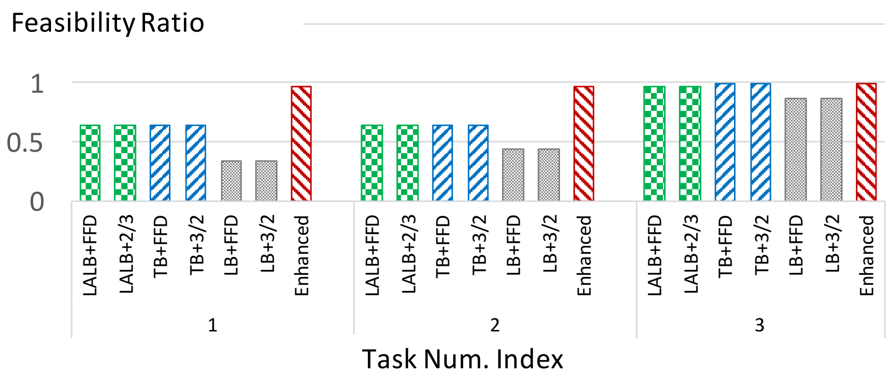
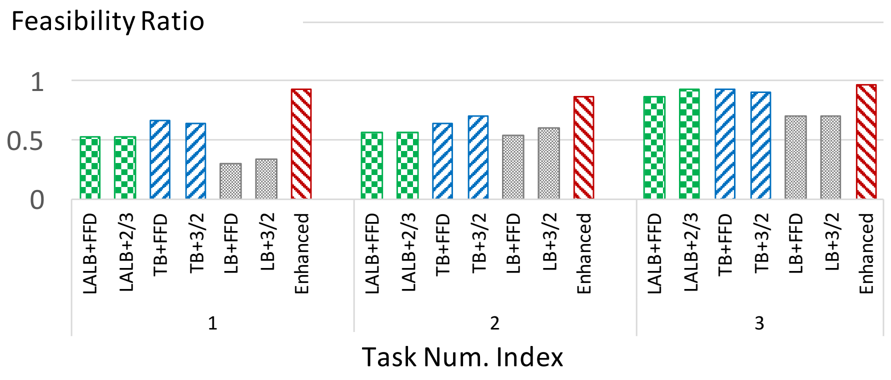


Figure 5.4: Average Energy Efficiency (WPJ) Comparison on large volumn of random cases

Next, we study the "heavy task" impacts on the feasibility. To this end, we varied the number of tasks in a given task set with a predefined system utilization. The smaller the task number is, the more likely "heavy tasks" will be generated. From Figure 5.5(a) and 5.5(b), we can see that the **En-VSBP** method always has the highest feasibility ratio in different configurations. The reason is that (1) The **En-VSBP** method considers the influence of "heavy tasks," which other approaches cannot pack successfully. (2) **En-VSBP** adopts the **TB** heuristic to determine the bin-sizes, which has a better average feasibility ratio than **LALB**, **TB**, as shown in Section 5.5.2. For example, the feasibility of **En-VSBP** exceeds the **LALB**, **TB** and **LB**-based approaches by 39.22%, 31.58% and 64.92%, respectively, when system utilization is between 50% and 100%. We also find that different bin-packing heuristics, e.g. **FFD** and **2/3-VSBP**, have similar feasibility ratios, which means that the feasibility of **2/3-VSBP** is not inferior to the **FFD** heuristic. Overall, the bin-size determination heuristic **TB** is better than **LALB** and **LB**, and the **En-**



(a)



(b)

Figure 5.5: Feasibility comparison when packing actual tasks for system utilization lies (a) between 0% and 50%; (b) between 50% and 100%

VSBP approache has the highest feasibility and energy efficiency for a large number of random cases.

5.6 Conclusion

As the IC industry enters a multi-core and many-core era, the energy efficiency becomes a more prominent criterion in the design of real-time schedules. In this chapter, we present a novel technique to schedule a real-time task set with maximized energy efficiency under a given peak temperature constraint. Our techniques are built upon the *thermal-balancing* heuristic and use the *variable-sized-bin-packing* method to maximally utilize system resources under a peak temperature constraint for energy minimization purposes. The validation results show that the *thermal-balancing* approach leads to significant improvement on energy efficiency and task partitioning feasibility, especially when the given temperature constraint is tight or the system utilization is high.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this chapter, we first summarize our contributions presented in this dissertation. We then discuss the possible directions for our future research work.

6.1 Summary

The advancement of IC technology enables confining more transistors within a single chip, as predicted by "Moore's Law". However, the consequent soaring power density and heat dissipation are two major obstacles in technology scaling. While multi-core architectures help to lower the power/thermal barrier for single core architectures, power/thermal issues are still the primary limiting factors to improve the system throughput. High temperature can negatively impact the system performance, degrade the system reliability and even permanently damage the chip. In addition, the heat flux and local hotspot on multi-core platforms worsen the thermal environment and make thermal management more complicated. The thermal/power-aware computing system design is urgently demanded in modern IC industry.

In this dissertation, my research deploys system-level real-time control technics on multi-core platforms to realize different design optimization goals (e.g. peak temperature reduction, throughput maximization and energy reduction, etc.) under a variety of system constraints (e.g. temperature threshold, power cap, etc.).

First, we introduce a set of provable fundamentals and principles for thermal-aware design based on the well-known multi-core RC-thermal model. Then, we develop an effective approach to identify and safely bound the peak temperature on multi-core platform, so-called "step-up schedule". We show that the traditional WCET-based peak temperature prediction is inaccurate. Instead, we formally prove that translating the given schedule to a *step-up schedule* can effectively bound the peak temperature, especially when the actual execution time varies from WCET. These principles are general enough to be applied on 2D and 3D multi-core platforms, and form the theoretical basis for a more rigorous analytical study of multi-core thermal problems.

We next solve the throughput maximization problem based on the step-up schedule and a frequency oscillating method. We found that oscillating on one/part of the multi-core platform cannot always reduce the peak temperature. Instead, synchronously oscillating all the cores monotonically reduces the peak temperature, when not considering the transition overhead. Further, we proposed design-time frequency/voltage oscillating approach on multi-core platforms incorporating the switching overhead.

Finally, we study the problem on how to reduce the energy consumption for a periodic real-time system under a given peak temperature constraint. We observed that evenly distributes the workload on all the processing cores no longer maximize the energy efficiency when the leakage power becomes significant. To this end, we propose a thermal-balancing approach to improve the overall system energy efficiency, especially when the temperature constraints are tight. We first identify the lower bound for energy consumption by this approach, and then transform the task partitioning problem to a variable sized bin packing problem. We further propose an enhanced algorithm to optimize the task partitioning results.

In sum, as technology scaling is becoming prohibitively expensive, seeking novel computing system design methodologies with real-time reconfigurability is an effective way to achieve different optimization goals. The undergoing research intends to understand the fundamentals through rigorous analytical formal methods, with an emphasis on the guaranteed performance and thermal constraints in the design of next generation of computing systems. More important, our system-level approaches can strictly guarantee the peak temperature constraint on multi-core platforms and they are general enough to be applied on other 2D and 3D multi-core thermal-aware design.

6.2 Future work

In the long term, the radical changes that involve completely different ways to compute will certainly happen, e.g. quantum, neuromorphic or mobile computing, etc. What is more, the way people use computing devices, e.g. smartphone or smart drive, and the new type of workload features, e.g. in visual processing, big data, AR/VR or cryptography technologies, challenge the semiconductor industry in many fronts. For example, the system scope, from single device to the system-of-systems, serves as a catalyst to accelerate the system innovation, both from architecture and system design perspectives. In addition, the real-time analysis and prediction becomes more complicated in consideration of human interference and environmental dynamics [98, 99, 83, 41, 60, 49, 61, 62]. My research aims to design adequate methodologies to predict/optimize the system behavior from the resource management standpoint to cope with the full complexity of future computing systems [146, 121, 94, 90, 143, 137, 155, 97, 152]. In particular, my future research aims to (1) design/optimize the computing systems that can better utilize system resources for performance improvement, (2) enhance the system power/thermal predicability with dynamic environment, (3) develop more aggressive and smart heat removal packages and methodologies.

3D IC Design From the hardware design perspective, 3D IC, integrating transistors vertically in three-dimension is a promising solution to achieve higher computing performance for future generations of IC chips. However, it becomes insufficient to use the traditional cooling techniques, such as cooling fan and heat sink, to remove the tremendous heat in a high power density and longer heat removal path. The thermal problem has become the bottleneck in the design of future generations of high-performance computing systems [116, 112, 33, 32, 116, 149, 23, 123, 29, 107, 128, 135].

The new liquid-based cooling method attracts researchers' and industrials' attention; however, the different thermal characteristics and controllability of liquid coolant raise new challenge in 3D processor design, e.g. the coolant temperature/heat removal capacity is quite different near the microchannel inlet and outlet, which exaggerates the thermal/performance imbalance across the chip [58, 75, 147, 129, 8, 108, 105, 159, 77, 120, 142, 3, 154, 67, 110, 34, 1]

One of my research interests is to build a more aggressive and finer granularity cooling infrastructure, that can be incorporated into the task allocation strategy, such that the cooling itself is a dynamic and a smart self-adjustable mechanism. The research can be conducted from two directions: (1) hardware innovation: designing a non-uniformed microchannel with different pipe widths/densities/topologies to mitigate the 3D thermal gradient in nature; (2) hardware and software co-schedule: developing coolant speed control schedules that match the task-assignment and execution speed control strategies [93, 88, 26, 27, 81]. The design outcomes are expected to enhance the existing 3D temperature prediction accuracy and response time, and deliver a higher system performance/reliability, etc.

Cyber-Physical Systems (CPS) Cyber-Physical Systems (CPS) links physical and computational counterparts to realize a smarter and seamless integration of computing, communication and control systems, and it drives innovative view of human and societal activities, including intelligent traffic monitoring, healthcare and agriculture, etc. However, how to enhance the system utilization to improve the CPS real-time responsibility and controllability is a major concern. The challenge lies in the fact that: (1) A large scale of different devices and systems are connected in a complex network. (2) The distributed configurations may rapidly change, which, in turn, challenge the real-time feasibility and controllability.

It is worth to study the spatial, temporal and hierarchical distribution characteristics in CPS system by capturing the coupled correlations on the system level to re-evaluate the system performance, reliability and power/energy from a statistical view. For example, many of the existing performance and power prediction tools are built upon the worst-case execution time, which is over pessimistic in a large dynamic distributed environment. To capture the reality of the system behavior, the statistical Quality-of-Service (QoS) on CPS need to be improved of its real-time schedulability. The future CPS system is also expected to deliver a higher service capacity to cope with "big data" and "Internet-of-Thing" for real-time control and adaptation. Since many applications exhibit large data volumes, the response time or energy in data storage, movement and processing dominates the system performance [136, 133, 45, 127, 25, 97, 118, 51]. I would like to conduct research on the memory-centric design, e.g. 3D memory stacking, processing in memory (PIM) [6, 36, 28, 158], etc, to enable future data processing beyond the state-of-the-art.

BIBLIOGRAPHY

[1] July 1975.

[2] M. Ahmed, N. Fisher, S. Wang, and P. Hettiarachchi. Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources. *Sustainable Computing: Informatics and Systems*, 1(3):226 – 240, 2011. Theoretical aspects of Sustainable Computing.

[3] R. Ahmed, P. Huang, M. Millen, and L. Thiele. On the design and application of thermal isolation servers. *ACM Trans. Embed. Comput. Syst.*, 16(5s):165:1–165:19, Sept. 2017.

[4] R. Ahmed, P. Ramanathan, and K. Saluja. Necessary and sufficient conditions for thermal schedulability of periodic real-time tasks. In *ECRTS*, pages 243–252, 2014.

[5] R. Ahmed, P. Ramanathan, and K. K. Saluja. Necessary and sufficient conditions for thermal schedulability of periodic real-time tasks under fluid scheduling model. *ACM Trans. Embed. Comput. Syst.*, 15(3):49:1–49:26, May 2016.

[6] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi. A scalable processing-in-memory accelerator for parallel graph processing. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pages 105–117, June 2015.

[7] H. Anton. *Elementary Linear Algebra, Applications Version 11E with Wiley-Plus Card.* John Wiley & Sons, Incorporated, 2014.

[8] K. Baati and M. Auguin. Temperature-aware dvfs-dpm for real-time applications under variable ambient temperature. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 13–20, June 2013.

[9] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo. Energy-aware scheduling for real-time systems: A survey. *ACM Trans. Embed. Comput. Syst.*, 15(1):7:1–7:34, Jan. 2016.

[10] M. Bao, A. Andrei, P. Eles, and Z. Peng. On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration. In *2009 46th ACM/IEEE Design Automation Conference*, pages 490–495, July 2009.

[11] B. Barrefors, Y. Lu, S. Saha, and J. S. Deogun. A novel thermal-constrained energy-aware partitioning algorithm for heterogeneous multiprocessor real-time systems. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, Dec 2014.

[12] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini. Cooling-aware node-level task allocation for next-generation green hpc systems. In *2016 International Conference on High Performance Computing Simulation (HPCS)*, pages 690–696, July 2016.

[13] F. Beneventi, A. Bartolini, A. Tilli, and L. Benini. An effective gray-box identification procedure for multicore thermal modeling. *IEEE Transactions on Computers*, 63(5):1097–1110, May 2014.

[14] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[15] R. W. Brodersen. Minimizing power consumption in digital cmos circuits. 2004.

[16] W. L. Brogan. *Modern Control Theory*. Ergodebooks, Richmond, TX, second edition, 1985.

[17] J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah. A categorization of real-time multiprocessor scheduling problems and algorithms. In *HANDBOOK ON SCHEDULING ALGORITHMS, METHODS, AND MODELS*. Chapman Hall/CRC, Boca, 2004.

[18] T. Chantem, X. S. Hu, and R. P. Dick. Online work maximization under a peak temperature constraint. In *ISLPED*, pages 105–110, 2009.

[19] T. Chantem, X. S. Hu, and R. P. Dick. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(10):1884–1897, Oct 2011.

[20] V. Chaturvedi, H. Huang, S. Ren, and G. Quan. On the fundamentals of leakage aware real-time dvs scheduling for peak temperature minimization. *Journal of Systems Architecture*, 58(10):387–397, 2012.

[21] A. Chen. International technology roadmap for semiconductors, emerging research devices workshop. 2015.

[22] G. Chen, K. Huang, and A. Knoll. Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination. *ACM Trans. Embed. Comput. Syst.*, 13(3s):111:1–111:21, Mar. 2014.

[23] G. Chen, J. Kuang, Z. Zeng, H. Zhang, E. F. Y. Young, and B. Yu. Minimizing thermal gradient and pumping power in 3d ic liquid cooling network design. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2017.

[24] J. J. Chen and L. Thiele. Task partitioning and platform synthesis for energy efficiency. In *RTCSA*, pages 393–402, Aug 2009.

[25] M. Chen, X. Zhang, H. Gu, t. wei, and Q. Zhu. Sustainability-oriented evaluation and optimization for mpsoc task allocation and scheduling under thermal and energy variations. *IEEE Transactions on Sustainable Computing*, PP(99):1–1, 2017.

[26] Y.-J. Chen, C.-L. Yang, P.-S. Lin, and Y.-C. Lu. Opportunities of synergistically adjusting voltage-frequency levels of cores and drams in cmps with 3d-stacked drams for efficient thermal control. *SIGAPP Appl. Comput. Rev.*, 16(1):26–35, Apr. 2016.

[27] W. K. Cheng, R. Y. Wang, and X. L. Li. 3d architecture exploration on thermal effect of dram refresh. In *2016 11th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT)*, pages 285–288, Oct 2016.

[28] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, pages 27–39, Piscataway, NJ, USA, 2016. IEEE Press.

[29] H.-W. Chiou and Y.-M. Lee. Thermal simulation for two-phase liquid cooling 3d-ics. *Journal of Computer and Communications*, (4):33–45, 2016.

[30] E. G. Coffman Jr., J. Csirik, G. Galambos, S. Martello, and D. Vigo. *Bin Packing Approximation Algorithms: Survey and Classification*, pages 455–531. Springer New York, New York, NY, 2013.

[31] S. R. Corporation. International technology roadmap for semiconductors. 2015.

[32] A. K. Coskun, D. Atienza, T. S. Rosing, T. Brunschwiler, and B. Michel. Energy-efficient variable-flow liquid cooling in 3d stacked architectures. In *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pages 111–116, March 2010.

[33] A. K. Coşkun, J. L. Ayala, D. Atienza, and T. S. Rosing. *Thermal Modeling and Management of Liquid-Cooled 3D Stacked Architectures*, pages 34–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[34] Y. Cui, W. Zhang, V. Chaturvedi, and B. He. Decentralized thermal-aware task scheduling for large-scale many-core systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(6):2075–2088, June 2016.

[35] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35:1–35:44, Oct. 2011.

[36] P. Emma, A. Buyuktosunoglu, M. Healy, K. Kailas, V. Puente, R. Yu, A. Hartstein, P. Bose, and J. Moreno. 3d stacking of high-performance processors. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 500–511, Feb 2014.

[37] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Power limitations and dark silicon challenge the future of multicore. *ACM Trans. Comput. Syst.*, 30(3):11:1–11:27, Aug. 2012.

[38] M. Fan, R. Rong, S. Liu, and G. Quan. Energy calculation for periodic multi-core scheduling in system thermal steady state with consideration of leakage and temperature dependency. *The Journal of Supercomputing*, 71(7):2565–2584, 2015.

[39] M. Fan, R. Rong, S. Liu, and G. Quan. Energy calculation for periodic multi-core scheduling in system thermal steady state with consideration of leakage and temperature dependency. *J. Supercomput.*, 71(7):2565–2584, July 2015.

[40] S. Fan, S. M. Zahedi, and B. C. Lee. The computational sprinting game. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, pages 561–575, New York, NY, USA, 2016. ACM.

[41] X. Feng. *Design of Real-time Virtual Resource Architecture for Large-scale Embedded Systems.* PhD thesis, 2004. AAI3127085.

[42] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. Thermal-aware global real-time scheduling and analysis on multicore systems. *J. Syst. Archit.*, 57(5):547–560, May 2011.

[43] D. K. Friesen and M. A. Langston. Variable sized bin packing. *SIAM J. Comput.*, 15(1):222–230, Feb. 1986.

[44] Y. Fu, N. Kottenstette, C. Lu, and X. D. Koutsoukos. Feedback thermal control of real-time systems on multicore processors. In *Proceedings of the Tenth ACM International Conference on Embedded Software*, EMSOFT '12, pages 113–122, New York, NY, USA, 2012. ACM.

[45] B. Gaudette, C. J. Wu, and S. Vrudhula. Improving smartphone user experience by balancing performance and energy with probabilistic qos guarantee. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 52–63, March 2016.

[46] Y. Ge and Q. Qiu. Dynamic thermal management for multimedia applications using machine learning. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 95–100, June 2011.

[47] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, Mar. 2014.

[48] N. Guan, M. Stigge, W. Yi, and G. Yu. Fixed-priority multiprocessor scheduling with liu and layland's utilization bound. In *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 165–174, April 2010.

[49] C. Guo, X. Hua, H. Wu, D. Lautner, and S. Ren. Best-harmonically-fit periodic task assignment algorithm on multiple periodic resources. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1303–1315, May 2016.

[50] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization. WWC-4 (Cat. No.01EX538)*, pages 3–14, Dec 2001.

[51] M. Halpern, Y. Zhu, and V. J. Reddi. Mobile cpu's rise to power: Quantifying the impact of generational mobile cpu design trends on performance,

energy, and user satisfaction. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 64–76, March 2016.

[52] Q. Han, M. Fan, O. Bai, S. Ren, and G. Quan. Temperature-constrained feasibility analysis for multi-core scheduling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, PP(99):1–1, 2016.

[53] V. Hanumaiah, D. Desai, B. Gaudette, C.-J. Wu, and S. Vrudhula. Steam: A smart temperature and energy aware multicore controller. *ACM Trans. Embed. Comput. Syst.*, 13(5s):151:1–151:25, Oct. 2014.

[54] V. Hanumaiah and S. Vrudhula. Energy-efficient operation of multicore processors by dvfs, task migration, and active cooling. *IEEE Transactions on Computers*, 63(2):349–360, 2014.

[55] V. Hanumaiah, S. Vrudhula, and K. Chatha. Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(11):1677–1690, Nov 2011.

[56] V. Hanumaiah, S. Vrudhula, and K. S. Chatha. Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(11):1677–1690, 2011.

[57] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, 2011.

[58] S. M. Hassan and S. Yalamanchili. Understanding the impact of air and microfluidics cooling on performance of 3d stacked memory systems. In *Proceedings of the Second International Symposium on Memory Systems*, MEMSYS '16, pages 387–394, New York, NY, USA, 2016. ACM.

[59] S. Hong, T. Chantem, and X. S. Hu. Local-deadline assignment for distributed real-time systems. *IEEE Transactions on Computers*, 64(7):1983–1997, July 2015.

[60] X. Hua, C. Guo, H. Wu, D. Lautner, and S. Ren. Schedulability analysis for real-time task set on resource with performance degradation and dual-level periodic rejuvenations. *IEEE Transactions on Computers*, PP(99):1–1, 2016.

[61] X. Hua, Z. Li, H. Wu, C. Guo, and S. Ren. Periodic resource integration. *Journal of Systems and Software*, 110:193 – 204, 2015.

[62] X. Hua, Z. Li, H. Wu, and S. Ren. Scheduling periodic tasks on multiple periodic resources. *2014 The Fourth International Conference on Advanced Communications and Computation (INFOCOMP)*, PP:35–40, 2014.

[63] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu. Throughput maximization for periodic real-time systems under the maximal temperature constraint. *ACM Trans. Embed. Comput. Syst.*, 13(2s):70:1–70:22, Jan. 2014.

[64] H. Huang, G. Quan, and J. Fan. Leakage temperature dependency modeling in system level analysis. In *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, pages 447–452, March.

[65] K. Huang, L. Santinelli, J. J. Chen, L. Thiele, and G. C. Buttazzo. Periodic power management schemes for real-time event streams. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 6224–6231, Dec 2009.

[66] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. 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, May 2006.

[67] A. Iranfar, M. Kamal, A. Afzali-Kusha, M. Pedram, and D. Atienza. Thespot: Thermal stress-aware power and temperature management for multiprocessor systems-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, PP(99):1–1, 2017.

[68] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4), Nov. 2007.

[69] H. Izakian, A. Abraham, and V. Snasel. Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. In *CSO*, volume 1, pages 8–12, April 2009.

[70] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *DAC*, pages 275–280, 2004.

[71] M. Kadin and S. Reda. Frequency planning for multi-core processors under thermal constraints. In *ISLPED*, pages 213–216, 2008.

[72] A. B. Kahng. The itrs design technology and system drivers roadmap: Process and status. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 34:1–34:6, New York, NY, USA, 2013. ACM.

[73] A. Kanduri, M. H. Haghbayan, A. M. Rahmani, P. Liljeberg, A. Jantsch, and H. Tenhunen. Dark silicon aware runtime mapping for many-core systems: A patterning approach. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 573–580, Oct 2015.

[74] H. Khdr, S. Pagani, . Sousa, V. Lari, A. Pathania, F. Hannig, M. Shafique, J. Teich, and J. Henkel. Power density-aware resource management for heterogeneous tiled multicores. *IEEE Transactions on Computers*, 66(3):488–501, March 2017.

[75] E. Kim, K. G. Shin, and J. Lee. Real-time battery thermal management for electric vehicles. In *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*, pages 72–83, April 2014.

[76] Z. Kishka, M. Abul-Ez, M. Saleem, and H. Abd-Elmageed. Lhospital rule for matrix functions. *Journal of the Egyptian Mathematical Society*, 21(2):115 – 118, 2013.

[77] C. Krishna and I. Koren. Thermal-aware management techniques for cyber-physical systems. *Sustainable Computing: Informatics and Systems*, 15(Supplement C):39 – 51, 2017.

[78] K. Lampka, B. Forsberg, and V. Spiliopoulos. Keep it cool and in time: With runtime monitoring to thermal-aware execution speeds for deadline constrained systems. *Journal of Parallel and Distributed Computing*, 95:79 – 91, 2016. Special Issue on Energy Efficient Multi-Core and Many-Core Systems, Part I.

[79] W. Y. Lee. Energy-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):530–537, March 2012.

[80] D. Li and J. Wu. Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms. *IEEE Transactions on Parallel and Distributed Systems*, 26(3):810–823, March 2015.

[81] D. Li, K. Zhang, A. Guliani, and S. Ogrenci-Memik. Adaptive thermal management for 3d ics with stacked dram caches. In *Proceedings of the 54th Annual*

Design Automation Conference 2017, DAC '17, pages 3:1–3:6, New York, NY, USA, 2017. ACM.

[82] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO*, pages 469–480, 2009.

[83] Y. Li, A. M. K. Cheng, and A. K. Mok. Regularity-based partitioning of uniform resources in real-time systems. In *Proceedings of the 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, RTCSA '12, pages 368–377, Washington, DC, USA, 2012. IEEE Computer Society.

[84] C. H. Liao and C. H. P. Wen. Thermal-constrained task scheduling on 3-d multicore processors for throughput-and-energy optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(11):2719–2723, Nov 2015.

[85] C. H. Liao, C. H. P. Wen, and K. Chakrabarty. An online thermal-constrained task scheduler for 3d multi-core processors. In *DATE*, pages 351–356, 2015.

[86] W. Liao, L. He, and K. M. Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(7):1042–1053, July 2005.

[87] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20:46–61, January 1973.

[88] W. H. Lo, K. z. Liang, and T. Hwang. Thermal-aware dynamic page allocation policy by future access patterns for hybrid memory cube (hmc). In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1084–1089, March 2016.

[89] G. H. Loh and Y. Xie. 3d stacked microprocessor: Are we there yet? *IEEE Micro*, 30(3):60–64, 2010.

[90] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean. A statistical response-time analysis of real-time embedded systems. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, pages 351–362, Dec 2012.

[91] Y. Ma, T. Chantem, R. P. Dick, and X. S. Hu. Improving system-level lifetime reliability of multicore soft real-time systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, PP(99):1–11, 2017.

[92] M. Marcus and H. Minc, editors. *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Boston, MA, USA, 1964.

[93] M. D. Marino. Abat-fs: Towards adjustable bandwidth and temperature via frequency scaling in scalable memory systems. *Microprocessors and Microsystems*, 45(Part B):339 – 354, 2016.

[94] D. Maxim, O. Buffet, L. Santinelli, L. Cucu-Grosjean, and R. I. Davis. Optimal priority assignment algorithms for probabilistic real-time systems. In *19th International Conference on Real-Time and Network Systems, RTNS'11, Nantes, France, September 29-30, 2011. Proceedings*, pages 129–138, 2011.

[95] C. D. Meyer, editor. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[96] M. Mitchell Waldrop. The chips are down for moore's law. *Nature News*, 530:144, 02 2016.

[97] M. Mohaqeqi, M. Kargahi, and K. Fouladi. Stochastic thermal control of a multicore real-time system. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 208–215, Feb 2016.

[98] A. K. Mok and X. Alex. Towards compositionality in real-time resource partitioning based on regularity bounds. In *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, pages 129–138, Dec 2001.

[99] A. K. Mok, X. Feng, and D. Chen. Resource partition for real-time systems. In *Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE*, pages 75–84, 2001.

[100] F. Mulas, D. Atienza, A. Acquaviva, S. Carta, L. Benini, and G. D. Micheli. Thermal balancing policy for multiprocessor stream computing platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(12):1870–1882, Dec 2009.

[101] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. D. Micheli, and R. Gupta. Processor speed control with thermal constraints. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(9):1994–2008, Sept 2009.

[102] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, 46(4):47:1–47:31, Mar. 2014.

[103] S. Pagani and J. J. Chen. Energy efficiency analysis for the single frequency approximation (sfa) scheme. In *RTCSA*, pages 82–91, Aug 2013.

[104] S. Pagani, J. J. Chen, M. Shafique, and J. Henkel. Matex: Efficient transient and peak temperature computation for compact thermal models. In *DATE*, pages 1515–1520, 2015.

[105] S. Pagani, H. Khdr, J. J. Chen, M. Shafique, M. Li, and J. Henkel. Thermal safe power (tsp): Efficient power budgeting for heterogeneous manycore systems in dark silicon. *IEEE Transactions on Computers*, PP(99):1–1, 2016.

[106] S. Pagani, H. Khdr, J. J. Chen, M. Shafique, M. Li, and J. Henkel. Thermal safe power (tsp): Efficient power budgeting for heterogeneous manycore systems in dark silicon. *IEEE Transactions on Computers*, 66(1):147–162, Jan 2017.

[107] P. R. Parida, A. Sridhar, A. Vega, M. D. Schultz, M. Gaynes, O. Ozsun, G. McVicker, T. Brunschwiler, A. Buyuktosunoglu, and T. Chainer. Thermal model for embedded two-phase liquid cooled microprocessor. In *2017 16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 441–449, May 2017.

[108] L. Piga, I. Paul, and W. Huang. Performance boosting opportunities under communication imbalance in power-constrained hpc clusters. In *2016 45th International Conference on Parallel Processing (ICPP)*, pages 31–40, Aug 2016.

[109] G. D. Poole. Generalized m-matrices and applications. *Mathematics of Computation*, 29(131):903–910, 1975.

[110] A. Prakash, H. Amrouch, M. Shafique, T. Mitra, and J. Henkel. Improving mobile gaming performance through cooperative cpu-gpu thermal management. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2016.

[111] M. Pricopi and T. Mitra. Task scheduling on adaptive multi-core. *IEEE Transactions on Computers*, 63(10):2590–2603, Oct 2014.

[112] D. Qiu, L. Cao, Q. Wang, F. Hou, and X. Wang. Experimental and numerical study of 3d stacked dies under forced air cooling and water immersion cooling. *Microelectronics Reliability*, 74(Supplement C):34 – 43, 2017.

[113] R. Rao and S. Vrudhula. Fast and accurate prediction of the steady-state throughput of multicore processors under thermal constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1559–1572, Oct 2009.

[114] A. Rezaei, D. Zhao, M. Daneshtalab, and H. Wu. Shift sprinting: Fine-grained temperature-aware noc-based mcsoc architecture in dark silicon age. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 155:1–155:6, New York, NY, USA, 2016. ACM.

[115] M. M. Sabry, A. K. Coskun, D. Atienza, T. . Rosing, and T. Brunschwiler. Energy-efficient multiobjective thermal control for liquid-cooled 3-d stacked architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(12):1883–1896, Dec 2011.

[116] M. M. Sabry, A. Sridhar, J. Meng, A. K. Coskun, and D. Atienza. Greencool: An energy-efficient liquid cooling design technique for 3-d mpsocs via channel width modulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):524–537, April 2013.

[117] S. Saha, Y. Lu, and J. S. Deogun. Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems. In *RTCSA*, pages 41–50, 2012.

[118] O. Sahin and A. K. Coskun. Qscale: Thermally-efficient qos management on heterogeneous mobile platforms. In *Proceedings of the 35th International Conference on Computer-Aided Design*, ICCAD '16, pages 125:1–125:8, New York, NY, USA, 2016. ACM.

[119] O. Sahin, P. T. Varghese, and A. K. Coskun. Just enough is more: Achieving sustainable performance in mobile devices under thermal limitations. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 839–846, Nov 2015.

[120] B. Salami, H. Noori, F. Mehdipour, and M. Baharani. Physical-aware predictive dynamic thermal management of multi-core processors. *Journal of Parallel and Distributed Computing*, 95(Supplement C):42 – 56, 2016. Special Issue on Energy Efficient Multi-Core and Many-Core Systems, Part I.

[121] S. Schliecker, M. Negrean, and R. Ernst. Bounding the shared resource load for the performance analysis of multiprocessor systems. In *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, pages 759–764, March 2010.

[122] L. Schor, H. Yang, I. Bacivarov, and L. Thiele. Worst-case temperature analysis for different resource models. *IET Circuits, Devices Systems*, 6(5):297–307, Sept 2012.

[123] C. Serafy, A. Bar-Cohen, A. Srivastava, and D. Yeung. Unlocking the true potential of 3-d cpus with microfluidic cooling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4):1515–1523, April 2016.

[124] S. Sha, W. Wen, M. Fan, S. Ren, and G. Quan. Performance maximization via frequency oscillation on temperature constrained multi-core processors. In *2016 45th International Conference on Parallel Processing (ICPP)*, pages 526–535, Aug 2016.

[125] S. Sha, W. Wen, S. Ren, and G. Quan. A thermal-balanced variable-sized-bin-packing approach for energy efficient multi-core real-time scheduling. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, GLSVLSI '17, pages 257–262, New York, NY, USA, 2017. ACM.

[126] S. Sharifi, D. Krishnaswamy, and T. . Rosing. Prometheus: A proactive method for thermal management of heterogeneous mpsocs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(7):1110–1123, July 2013.

[127] H. F. Sheikh, I. Ahmad, and D. Fan. An evolutionary technique for performance-energy-temperature optimized scheduling of parallel tasks on multi-core processors. *IEEE Transactions on Parallel and Distributed Systems*, 27(3):668–681, March 2016.

[128] L. Siddhu and P. R. Panda. Thermal aware runtime management of 3d memory architecture. *CSI Transactions on ICT*, 5(2):129–134, Jun 2017.

[129] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel. Analysis and mapping for thermal and energy efficiency of 3-d video processing on 3-d multicore processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(8):2745–2758, Aug 2016.

[130] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 960–965, March 2015.

[131] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1):94–125, 2004.

[132] W. Sun and T. Sugawara. Heuristics and evaluations of energy-aware task mapping on heterogeneous multiprocessors. In *IPDPSW*, pages 599–607, May 2011.

[133] T.-C. Tang and Y.-S. Chen. Thermal-aware mapreduce real-time scheduling in heterogeneous server systems. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '16, pages 207–212, New York, NY, USA, 2016. ACM.

[134] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 4, pages 101–104 vol.4, 2000.

[135] T.-H. Tsai and Y.-S. Chen. Thermal-throttling server: A thermal-aware real-time task scheduling framework for three-dimensional multicore chips. *Journal of Systems and Software*, 112(Supplement C):11 – 25, 2016.

[136] S. K. S. Tyagi, D. K. Jain, S. L. Fernandes, and P. K. Muhuri. Thermal-aware power-efficient deadline based task allocation in multi-core processor. *Journal of Computational Science*, 19(Supplement C):112 – 120, 2017.

[137] I. Ukhov, P. Eles, and Z. Peng. Probabilistic analysis of power and temperature under process variation for electronic system design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(6):931–944, June 2014.

[138] I. Ukhov, P. Eles, and Z. Peng. Temperature-centric reliability analysis and optimization of electronic systems under process variation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(11):2417–2430, Nov 2015.

[139] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry. An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*, 16(1):3–15, Mar. 2013.

[140] R. Viswanath, W. Vijay, A. Watwe, and V. Lebonheur. Thermal performance challenges from silicon to systems. In *Intel Technology Journal*, Q3, 2000.

[141] W. Wahby, L. Zheng, Y. Zhang, and M. S. Bakir. A simulation tool for rapid investigation of trends in 3-dic performance and power consumption. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 6(2):192–199, Feb 2016.

[142] J. Wang, Z. Lu, Y. Li, Y. Fu, and J. Guo. A high-level thermal model-based task mapping for cmps in dark-silicon era. *IEEE Transactions on Electron Devices*, 63(9):3406–3412, Sept 2016.

[143] T. Wang, L. Niu, S. Ren, and G. Quan. Multi-core fixed-priority scheduling of real-time tasks with statistical deadline guarantee. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 1335–1340, San Jose, CA, USA, 2015. EDA Consortium.

[144] Z. Wang and S. Ranka. A simple thermal model for multi-core processors and its application to slack allocation. In *IPDPS*, pages 1–11, 2010.

[145] Z. Wang and S. Ranka. Thermal constrained workload distribution for maximizing throughput on multi-core processors. In *IGCC*, pages 291–298, 2010.

[146] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström. The worst-case execution-time problem&mdash;overview of methods and survey of tools. *ACM Trans. Embed. Comput. Syst.*, 7(3):36:1–36:53, May 2008.

[147] Q. Xie, J. Kim, Y. Wang, D. Shin, N. Chang, and M. Pedram. Dynamic thermal management in mobile devices considering the thermal coupling be-

tween battery and application processor. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 242–247, Nov 2013.

[148] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 374–382, Oct 1995.

[149] P. Zajc, C. Maj, and A. Napieralski. Peak temperature reduction by optimizing power density distribution in 3d ics with microchannel cooling. *Microelectronics Reliability*, 2017.

[150] A. N. Zehmakan. Bin packing problem: Two approximation algorithms. 2015.

[151] H. Zhang and H. Hoffmann. Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, pages 545–559, New York, NY, USA, 2016. ACM.

[152] K. Zhang, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman. Minimizing thermal variation across system components. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 1139–1148, May 2015.

[153] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. Technical report, 2003.

[154] J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan, and Y. Ma. Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms. *Journal of Systems and Software*, 133(Supplement C):1 – 16, 2017.

[155] J. Zhou and T. Wei. Stochastic thermal-aware real-time task scheduling with considerations of soft errors. *Journal of Systems and Software*, 102:123 – 133, 2015.

[156] J. Zhou, T. Wei, M. Chen, J. Yan, S. Hu, and Y. Ma. Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, PP(99):1–1, 2015.

[157] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma. Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(8):1269–1282, Aug 2016.

[158] Y. Zhu, B. Wang, D. Li, and J. Zhao. Integrated thermal analysis for processing in die-stacking memory. In *Proceedings of the Second International Symposium on Memory Systems*, MEMSYS '16, pages 402–414, New York, NY, USA, 2016. ACM.

[159] Z. Zhu, V. Chaturvedi, A. K. Singh, W. Zhang, and Y. Cui. Two-stage thermal-aware scheduling of task graphs on 3d multi-cores exploiting application and architecture characteristics. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 324–329, Jan 2017.

[160] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto. Survey of energy-cognizant scheduling techniques. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1447–1464, July 2013.

[161] C. Zweben. Ultrahigh-thermal-conductivity packaging materials. In *STHERM*, pages 168–174, 2005.

VITA

SHI SHA

2007	B.S., Electrical Engineering Beihang University Beijing, China
2010	M.S., Telecommunications Systems Management Murray State University Kentucky, USA
2018	Ph.D. candidate, Electrical and Computer Engineering Florida International University Florida, USA

PUBLICATIONS

Shi Sha, W. Wen, S. Ren and Gang Quan, "M-Oscillating: Performance Maximization on Temperature-Constrained Multi-Core Processors", *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2018 (conditionally accepted).

Shi Sha, G. A. Chaparro-Baquero, W. Wen and Gang Quan,"On the Fundamentals of thermal analysis on Multi-core Platforms", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2018 (under review).

Shi Sha, W. Wen, S. Ren and Gang Quan,"Thermal-Constrained Energy Efficient Real-Time Scheduling on Multi-Core Platforms", *Journal of Parallel Computing (PARCO)*, 2018 (under review).

G. A. Chaparro-Baquero, Shi Sha, S. Homsi, W. Wen, Gang Quan, "Thermal-Aware Joint CPU and Memory Scheduling for Hard Real-Time Tasks on Multi-Core 3D platforms", *8th International Green and Sustainable Computing Conference (IGSC)*, Orlando, FL 2017.

Shi Sha, W. Wen, S. Ren and Gang Quan, "A Thermal-Balanced Variable-Sized-Bin-Packing Approach for Energy Efficient Multi-Core Real-Time Scheduling", *2017 International Great Lakes Symposium on VLSI (GLSVLSI)*, Banff, Alberta, Canada, May 10-12, 2017.

G. A. Chaparro-Baquero, Shi Sha, S. Homsi, Gang Quan, "Process/Memory Co-scheduling Using Periodic Resource Server for Real-Time Systems Under Peak Temperature Constraints", *18th International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA 2017.

Shi Sha, W. Wen, M. Fan, S. Ren and Gang Quan, "Performance Maximization via Frequency Oscillation on Temperature Constrained Multicore Processors", *International Conference on Parallel Processing (ICPP) 2016*, Philadelphia, PA, August 16-19, 2016.

T. Wang, Q. Han, Shi Sha, W. Wen, Gang Quan and M. Qiu, "Fixed-Priority Periodic Real-Time Tasks with Explicit Deadlines", *IEEE/ACM Design Automation Conference (DAC) 2016*, Austin, TX, June 5-9, 2016.

M. Fan, V. Chaturvedi, Shi Sha and Gang Quan, "An analytical solution for multi-core energy calculation with consideration of leakage and temperature dependency," *International Symposium on Low Power Electronics and Design (ISLPED)*, Beijing, 2013, pp. 353-358.

Shi Sha; J. Zhou; C. Liu; Gang Quan, "Power and energy analysis on intel Single-Chip Cloud Computer system," *IEEE SoutheastCon 2012*, Orlando, FL, March 15-18, 2012.