On Fundamental Principles for Thermal-Aware Design on Periodic Real-Time Multi-Core Systems

SHI SHA, Wilkes University, USA
AJINKYA S. BANKAR, Florida International University, USA
XIAOKUN YANG, University of Houston Clear Lake, USA
WUJIE WEN, Lehigh University, USA
GANG QUAN, Florida International University, USA

With the exponential rise of the transistor count in one chip, the thermal problem has become a pressing issue in computing system design. While there have been extensive methods and techniques published for design optimization with thermal awareness, there is a need for more rigorous and formal thermal analysis in designing real-time systems and applications that demand a strong exception guarantee. In this paper, we analytically prove a series of fundamental properties and principles concerning RC thermal model, peak temperature identification and peak temperature reduction for periodic real-time systems, which are general enough to be applied on 2D and 3D multi-core platforms. These findings enhance the worst-case temperature predictability in runtime scenarios, help to develop more effective thermal management policy, which is key to thermal-constrained periodic real-time system design.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Multicore architectures*; *Real-time operating systems*; *Real-time system architecture*; • **Hardware → Temperature simulation and estimation**; **Temperature control**; **Temperature optimization**.

Additional Key Words and Phrases: multi-core architecture; thermal-aware design; temperature bound; peak temperature minimization; dynamic thermal management (DTM); dynamic voltage frequency scaling (DVFS); worst- case execution time (WCET).

ACM Reference Format:
Shi Sha, Ajinkya S. Bankar, Xiaokun Yang, Wujie Wen, and Gang Quan. 2018. On Fundamental Principles for Thermal-Aware Design on Periodic Real-Time Multi-Core Systems. *ACM Trans. Des. Autom. Electron. Syst.* 37, 4, Article 111 (August 2018), 31 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The advancement of IC technology has changed human life and society profoundly, largely due to the tremendous progress of computing systems ranging from large-scale data centers to daily used mobile devices. However, the semiconductor industry development is now reaching a saturation point of Moore's Law due to high power consumption and heat dissipation, among other factors.

Authors' addresses: Shi Sha, shi.sha@wilkes.edu, Wilkes University, 84 W. South Street, Wilkes-Barre, Pennsylvania, 18766, USA; Ajinkya S. Bankar, abank013@fiu.edu, Florida International University, 10555 West Flagler Street, EC3900, Miami, Florida, 33174, USA; Xiaokun Yang, yangxia@uhcl.edu, University of Houston Clear Lake, 2700 Bay Area Blvd, Houston, Texas, 77058, USA; Wujie Wen, wuw219@lehigh.edu, Lehigh University, 19 Memorial Drive W., Bethlehem, Pennsylvania, 18015, USA; Gang Quan, gaquan@fiu.edu, Florida International University, 10555 West Flagler Street, EC3900, Miami, Florida, 33174, USA.

1084-4309/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

The soaring power and accompanied thermal crisis negatively impacts the system computational performance, reliability, shortens devices lifespan or even damages the processor permanently. Temperature has become one of the primary concerns in modern microprocessor design and the thermal environment is becoming even worse in many-core and 3D architectures.

While it is a common practice to limit the power consumption – using so-called *thermal design power* (TDP) – to ensure that thermal issues are under control, the challenges of temperature management on 2D or 3D multi-core platform come from not only the high power dissipation, but also the uneven heat dissipation temporally and spatially. For example, on an Intel Xeon E5-2699 v3 CPU [5], the intra-die temperature difference can be up to 10°C and 24°C under balanced and unbalanced workload scenarios, respectively. In addition, thermal nodes with longer heat removal paths in 3D architecture exacerbate the thermal gradient. Therefore, conventional power budgeting on multi-core platforms become insufficient in thermal guarantee [39].

To reduce the chip temperature, mechanical cooling solutions, e.g. thermal-aware floor-planning, heat removal fan and micro-channel liquid cooling, fall short to many computing systems, especially the mobile ones. Alternatively, *dynamic thermal management* (DTM) is widely accepted as an effective solution both in design and runtime to smooth the thermal gradient by *dynamic voltage frequency scaling* (DVFS), or to shut down the unused cores by *dynamic power management* (DPM) [48].

Many DTM strategies are proposed, such as thermal-balancing [34], "hot-and-cold" job swapping [41], allocating hot tasks to cores closer to the heat sink [29], etc. These heuristic approaches developed cannot guarantee the temperature constraint in computing system design. There are also other approaches that resort to traditional control and optimization methods, such as feedback control [4, 19], machine learning [12, 15], mathematical programming [23, 54]. While some of these approaches, e.g. the runtime learning approach in [15], can guarantee the temperature constraints, it could be extremely challenging when employing the above approach to ensure both the timing and peak temperature constraints for real-time systems with complicated scheduling policies.

To facilitate more rigorous analytical thermal analyses, which we believe is indispensable for system level thermal-aware real-time system design, we intend to develop general and provable principles and fundamentals on characteristics of heat dissipation for ease of formal verification and analysis. The contribution of this paper includes:

- First, we introduce a series of provable lemmas and theorems, which unveil some interesting characteristics of the complex RC-thermal model and facilitate more formal analytical thermal-aware analyses and design on multi-core platforms;
- Second, peak temperature plays a critical role in the thermal-aware periodic real-time system design, but identifying the peak temperature on multi-core platforms is non-trivial. In this paper, we propose to employ the so-called "step-up execution trace" to quickly and effectively bound the peak temperature for periodic real-time systems, i.e. the commonly used real-time system model, and formally prove its validity. Our proposed method can be broadly adopted in real-time systems design when developing both online and offline thermal management policies;
- Third, it is known that on a single-core platform, splitting the tasks with different power/thermal characteristics into multiple sections and executing them interchangeably, namely *frequency oscillating*, can reduce the peak temperature [25]. However, we find that applying such a mechanism on one core or on a part of the chip cannot always reduce the peak temperature on multi-core platforms. Instead, we develop the multi-core *"m-Oscillating"* method and formally prove that it can reduce the overall peak temperature. Moreover, we also prove

that the *"m-Oscillating"* can enhance the system's real-time service throughput under peak temperature constraints in the real-time domain.

These conclusions can be applied to both online and offline design stages and address the urgent demands of runtime performance/thermal guarantees in the real-time system design. They are also 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.

2 RELATED WORKS

There have been extensive research efforts for thermal related optimizations on multi-core platforms, including throughput maximization (e.g. [13, 35, 45, 54]), power/energy reduction (e.g. [37, 42, 56]), peak temperature reduction (e.g. [8, 14, 57]) and reliability enhancement (e.g. [52]), 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 different methodologies, the existing work can be largely classified into the following three categories.

First, many heuristic methods rely on extensive testing and hand-tuning. For example, several solutions have been proposed for the peak temperature minimization problem, including interleaving the hot/cool tasks in 3D platforms temporally and spatially [29], assigning slacks to split hot tasks [57] and associating active cooling with task assignment [5]. However, in these approaches, to determine the accurate and strongly justifiable metrics to classify hot/cool tasks/cores can be difficult. In addition, without capturing solid analytical correlations, it can be challenging to make other design tradeoffs in the meantime, such as task migration overhead *vs.* scheduling interval length. 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 [4, 19] or optimization methods, such as machine learning [15], mathematical programming or meta-heuristic searching methods [55], to deal with thermal issues. For example, using feedback control techniques on multi-core platforms, Bartolini et al. [4] proposed a feedback control framework to enforce thermal control, energy minimization and thermal model self-calibration. Hanumaiah et al. [19] 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. [55] developed a look-up table based DTM method on a thermal coupled processor/battery model to achieve the maximal throughput. These approaches help to uncover the rationales in temperature management. However, it is still difficult to strongly guarantee the temperature constraints due to the inaccurate sensor-based temperature reading and control.

The mathematical programming methods are also employed to optimize resource allocation under temperature and other design constraints. For example, the throughput maximization problem on a temperature-constrained multi-core platform has been studied by the linear programming (LP) approach in [54], and by a convex optimization method in [35]. However, [35] ignores the lateral heat transfer among different cores. Other approaches, such as mixed-integer linear programming (MILP), have been used to minimize the peak temperature in [8], or reduce the peak temperature and energy contemporarily for video streaming in [49]. A genetic programming approach has been proposed to minimize the energy consumption for periodic tasks under a peak temperature constraint in [42]. A meta-heuristic approach has been proposed to boost system performance in a small interval by supplying additional power to the system without exceeding the temperature and power supply limit in [13]. These mathematical programming-based approaches can provide

a solution, but it cannot qualitatively express the impacts between different variables and the optimization goals. In addition, the computational costs of many mathematical programming-based solutions, e.g. [8, 49, 54], increase too fast and can be prohibitive as the system scale becomes larger in future many-core processors.

The third type of approaches (e.g. [2, 14, 39, 45, 55]) intend to ensure a 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 the 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, a thermal-aware global scheduling algorithm for sporadic task sets has been proposed in [14]; the energy optimization under a given temperature constraint has been solved by a convex method in [22] and by a variable-sized bin packing approach in [44]; the throughput maximization problem has been studied in [21, 23]; the reliability has been enhanced in [20]. However, since [14, 20–23] incorrectly assume the peak temperature only occurred at a scheduling point and [20, 21, 23] ignore the lateral heat transfer among different cores, the inaccurate peak temperature prediction may trigger the DTM or even shut down the processors, which may lead to a real-time violation. To achieve higher computational capacity, the thermal safe power (TSP) has been proposed in [39], which achieves a higher power budget than the traditional thermal design power (TDP). Sha et al. [45] present a frequency oscillating method to maximize the throughput with a guaranteed peak temperature on a multi-core platform. Assisted with rigorous mathematical analysis, these approaches help to uncover fundamental principles for more efficient and effective thermal-aware design, which would be otherwise unavailable. Moreover, the formal analytical analyses do not suffer from prohibitive computational cost in many mathematical programming approaches.

In this paper, based on the traditional RC-thermal model for multi-core platforms, we present a series of analyses on thermal models, peak temperature identification and reduction, which we believe can greatly enhance the formal analytical research on thermal-constrained design problem on multi-core platforms. The rest of this paper is organized as follows. Section III introduces the multi-core system and thermal models. Section IV studies the general principle of the RC-thermal model. Peak temperature bound and minimization is discussed in Section V and Section VI, respectively. Section VII shows the experiment results and is followed by a conclusion in Section VIII.

3 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 1 are used in the paper.

3.1 System Model

We consider a multi-core platform. Each processing core is DVFS-independent. 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 idle core, we assume $v = f = 0$ and for an inactive node (such as heat sink/spreader), we assume there is no power consumption. 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**.

Table 1. Summary of Notations

Symbol	Meaning
N	Total number of thermal nodes;
$\mathbb{S}(t)$	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. $l_q = t_q - t_{q-1}$;
$\mathbf{v}_q$	The supply voltage of the q_{th} state interval;
$\mathbf{T_0}$	The starting temperatures;
$\mathbf{T}_{ss}(t)$	The stable status temperatures at time t;
$\mathbf{T}_{peak}(\mathbb{S}(t))$	The peak temperatures of running $\mathbb{S}(t)$;
$\mathbf{T}_q^{\infty}$	The constant temperature when running processor using supply voltage $\mathbf{v}_q$ long enough to enter the stable state;
$\mathbf{1}_{N\times 1}$	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 $\forall j \in [1, N_2]$.	

3.2 Thermal Model

The thermal model, similar to that in [18, 44, 45, 53], is built upon the duality between heat transfer and electrical phenomena as an RC-lumped circuit, which is general enough to be used to model any multi-core platforms with multiple processing units of different characteristics of power consumption and heat dissipation. Specifically, the RC-model consists of a network of N active and inactive thermal nodes, with active nodes consuming power and generating heat (e.g. execution cores, memories, etc.) and inactive nodes having no power/heat dissipation (e.g. heat sink, etc.). Please note that active nodes in the RC model are not limited to CPUs alone. Cache units, TSVs and other components in a 2D or 3D architecture can be well modeled as active nodes in the RC-model when they consume a significant amount of power and generate a noticeable amount of heat. The thermal behavior of a multi-core platform within a state interval can be formulated as

$$\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\mathbf{T}(t) + \mathbf{B}(\mathbf{v}), \tag{1}$$

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

$$G_{i,j} = \begin{cases} \sum_{\theta \neq i} \frac{1}{R_{i,\theta}}, & \text{if } i = j, \\ -\frac{1}{R_{i,j}}, & \text{otherwise,} \end{cases} \tag{2}$$

in which $R_{i,i}$ (or $R_{i,j}$) denotes the thermal resistance of the i-th thermal node to itself (or the j-th thermal node). More details on the thermal model can be found in [18].

Existing studies show that matrix $\mathbf{G}$ has the following properties:

Property 1. *Matrix* $\mathbf{G}$ *has the 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 [18];*
(2) $\mathbf{G}$ *is strictly diagonally dominant, real symmetric and nonsingular (Lemma 1 in [53]);*

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 active thermal node. It is reasonable to assume that $\forall \mathbf{v}_1 \geq \mathbf{v}_2$ leads to $\mathbf{B}(\mathbf{v}_1) \geq \mathbf{B}(\mathbf{v}_2)$. The power-related vector $\mathbf{B}$ does not make any assumption on power parameters, so it is general enough to adopt different power models. In this paper, we take leakage-temperature dependency [18, 44] on the active processing units into accounts in Section 7.

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 [18], 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

$$\mathbf{T}(t) = e^{\mathbf{A}(t-t_{q-1})}\mathbf{T}(t_{q-1}) + (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})\mathbf{T}_q^{\infty}, \tag{3}$$

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 to enter the stable state and $\mathbf{I}$ is an identity matrix.

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 [18]

$$\mathbf{T}_{ss}(t_q) = \mathbf{T}(t_q) + \mathbf{K}_q(\mathbf{I} - \mathbf{K})^{-1}(\mathbf{T}(t_p) - \mathbf{T}(0)), \tag{4}$$

in which $\mathbf{T}(t_q)$ and $\mathbf{T}_{ss}(t_q)$ are the temperatures at time t_q in the first period and that 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 $\mathbf{K} = e^{\mathbf{A}\sum_{\theta=1}^{z} l_\theta} = e^{\mathbf{A}t_p}$. A more detailed explanation of the notations and equations can be found in [18].

4 THE PROPERTIES OF THE THERMAL MODEL

In this section, we focus on some inherent properties related to the multi-core RC thermal model itself. We believe that a better understanding of the thermal models helps to develop more effective thermal management policies in computing systems design.

The thermal model in (1) 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 a vital role in determining temperature dynamics according to the transient power dissipation in runtime. Note that $\mathbf{A}$ affects how current temperature influences the future temperature change in $d\mathbf{T}(t)/dt$ [18] and $\mathbf{A}$ also affects the stable state temperature in $\mathbf{T}^{\infty}$. Moreover, the properties of $\mathbf{A}$ determine the system stability [6], 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. To better understand the properties related

to matrix $\mathbf{A}$, we present several important lemmas and theorems as follows. In this paper, all the detailed proofs can be found in the Appendix, if otherwise specified.

Lemma 4.1. *Matrix* $\mathbf{A}$ *has all negative real eigenvalues.*

Lemma 4.2. *Matrix* $\mathbf{A}$ *is diagonalizable.*

Lemma 4.3. *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* $\mathscr{A}_{i,j} > 0$.

In control theory, a system is *asymptotically stable* [7], if all the eigenvalues of the system matrix are strictly negative real values. An *asymptotically stable* system is bounded-input, bounded-output (BIBO) stable, which means that the output will be bounded for every input to the system that is bounded. Therefore, from Lemma 4.1, there always exists a peak temperature for any schedule executed on a given platform, with its power supply stay below the maximal threshold.

Since $\mathbf{A}$ is diagonalizable (Lemma 4.2) and all of its eigenvalues are negative (Lemma 4.1), we can easily calculate its eigenvalues. Let $-\lambda_i$ be the *i-th* eigenvalue of $\mathbf{A}$ and $\lambda_i > 0$, we have $\mathbf{A} = \mathbf{W}\mathbf{D}\mathbf{W}^{-1}$, 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

$$e^{\mathbf{A}l} = \sum_{h=0}^{\infty} \frac{l^h(\mathbf{W}\mathbf{D}\mathbf{W}^{-1})^h}{h!} = \mathbf{W}\Big(\sum_{h=0}^{\infty} \frac{l^h\mathbf{D}^h}{h!}\Big)\mathbf{W}^{-1} = \mathbf{W}e^{\mathbf{D}l}\mathbf{W}^{-1}, \tag{5}$$

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 $e^{\mathbf{A}l}$.

In addition, since none of the eigenvalue of $\mathbf{A}$ equal to zero (Lemma 4.1), matrix $\mathbf{A}$ is invertible. The negative inverted matrix $-\mathbf{A}^{-1}$ plays an important role in determining the stable state temperature $\mathbf{T}^{\infty}(\mathbf{v})$, which, in turn, impacts the thermal dynamics in (3). This attribute of $-\mathbf{A}^{-1}$ in Lemma 4.3 also enables the comparison of the stable state temperatures of different constant running modes. In particular, since power consumption is proportional to the supply voltages of different running modes, i.e. $\forall \mathbf{v_1} \geq \mathbf{v_2}$ leads to $\mathbf{B}(\mathbf{v_1}) \geq \mathbf{B}(\mathbf{v_2})$, we can readily conclude that $\mathbf{T}^{\infty}(\mathbf{v_1}) \geq \mathbf{T}^{\infty}(\mathbf{v_2})$ as formulated in the following lemma.

Lemma 4.4. *Given a multi-core platform running two constant modes long enough to enter the stable state, if the supply voltages of the two running modes satisfy* $\mathbf{v}_1 \geq \mathbf{v}_2$, *we have* $\mathbf{T}^{\infty}(\mathbf{v}_1) \geq \mathbf{T}^{\infty}(\mathbf{v}_2)$.

Lemma 4.4 implies that when a multi-core platform executes multiple state intervals from the same initial temperature, a higher speed results in higher transient temperature. We can also infer that when a multi-core platform executes two (series of) state intervals of the same (series of) modes, the one starting with higher initial temperature results in higher transient temperature. While these observations are intuitive and straightforward, they serve as the foundation for comparing the temperature in more complex schedules.

In thermal analyses, capturing the thermal dynamics is key to guarantee the maximal temperature constraint. To this end, matrix $\mathbf{K}$ in (4) is used to determine the temperature dynamics in the first period and the stable status as shown in [18]. Specifically, for matrix $\mathbf{K}$, we have the following lemma and theorem.

Lemma 4.5. *All the elements in the matrix* $(\mathbf{I} - \mathbf{K})^{-1}$ *are positive and each entry monotonically decreases with* l, *where* $\mathbf{K} = e^{\mathbf{A}l}$, $l > 0$.

Theorem 4.6. *Let* $l > 0$ *and* $\mathbf{0} \leq \mathbf{T} \leq (\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v_{min}}))$, *then* $(\mathbf{I} - \mathbf{K})\mathbf{T} \geq \mathbf{0}$, *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.*

According to Theorem 4.6, 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 $\mathbf{T}$.

For a multi-core system initially starting from the ambient temperature, its peak temperature occurs when the temperature reaches a stable status [18]. However, with the help of Lemma 4.5 and Theorem 4.6, we can compare the peak temperatures of periodic schedules based on the temperatures in their first periods rather than their temperatures in the stable status. To put our discussions into perspective, consider two periodic schedules $\mathbb{S}(t)$ and $\mathbb{S}'(t)$ with the same intervals (totally z intervals) and period lengths t_p but different speeds, and let $\mathbf{T}(t_h)$ (or $\mathbf{T}_{ss}(t_h)$) and $\mathbf{T}'(t_h)$ (or $\mathbf{T}'_{ss}(t_h)$) represent the temperature at t_h in the first period (or in the stable status) of $\mathbb{S}(t)$ and $\mathbb{S}'(t)$, respectively. We can compare their stable state temperature at t_h by (4) as

$$\mathbf{T}_{ss}(t_h) - \mathbf{T}'_{ss}(t_h) = \mathbf{T}(t_h) - \mathbf{T}'(t_h) + \mathbf{K}_h(\mathbf{I}-\mathbf{K})^{-1}(\mathbf{T}(t_p) - \mathbf{T}'(t_p)). \tag{6}$$

Since $(\mathbf{I}-\mathbf{K})^{-1} > \mathbf{0}$ from Lemma 4.5 and $\mathbf{K}_h = e^{\mathbf{A}\sum_{\theta=1}^{h} l_\theta} > \mathbf{0}$, we have $\mathbf{T}_{ss}(t_h) \geq \mathbf{T}'_{ss}(t_h)$, if $\mathbf{T}(t_h) \geq \mathbf{T}'(t_h)$ and $\mathbf{T}(t_p) \geq \mathbf{T}'(t_p)$. With the knowledge of these properties, we are ready to introduce the peak temperature identification and bounding method.

Our thermal model is general enough to be widely adopted in solving a variety of 2D and 3D optimization problems. For example, under peak temperature constraints, 3D thermal-aware task allocation strategies are proposed in [11] and 3D thermal-aware energy optimization with memory-awarenesses are proposed in [31]. In these works, the thermal impact of the through-silicon-vias (TSV) is treated as a homogeneous via distribution on the die, whose thermal resistivity depends on TSV density [58]. Our thermal model has also been widely adopted in other thermal-aware optimization problems including 3D floor-planning [24], TSV placement [1, 10], reliability optimization [33], etc.. Note that the thermal model in this paper is valid for solving heat conduction in solid materials, which cannot be applied on the liquid-cooling architectures [51], or phase-change materials [16].

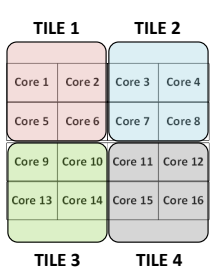
5 PEAK TEMPERATURE IDENTIFICATION AND BOUNDING

Peak temperature is one of the primary parameters for on-board thermal monitoring technologies. For example, Intel Xeon Processor E5 v3 family, ranging from 4 to18-core used for embedded and server domain, utilizes on-die Digital Thermal Sensors (DTS) to monitor the core's temperature and automatically trigger the DTM or even shut down the processors, when the runtime temperature approaches the threshold. In the absence of an effective runtime peak temperature prediction and bounding methodology, such a self-triggered thermal protection scheme may cause an unplanned performance degradation at the risk of deadline violation for real-time tasks. The peak temperature of a system is also closely related to other critical design metrics such as reliability [20].

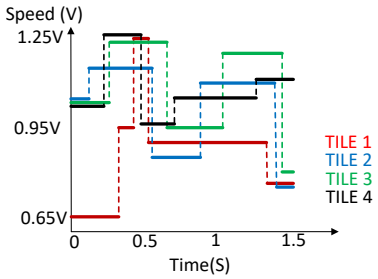
On single-core platforms, it is easy to capture the peak temperature since it always occurs at a scheduling point [9]. However, on multi-core platforms, identifying and bounding the peak temperature becomes more complicated, because different components may follow different execution schedules and the power densities vary significantly in one chip, which may shift the peak temperature away from a scheduling point.

There are a few approaches proposed to identify the peak temperature for a given schedule on multi-core platforms. 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. [40, 47, 50]). This approach is computationally expensive and its accuracy heavily depends on the checking granularity.

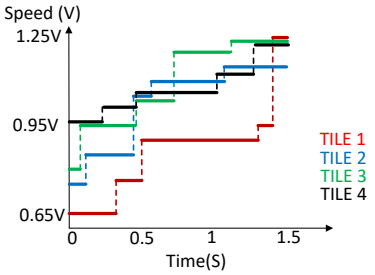
There are also some other approaches that intend to find the solution analytically. For example, Pagani et al. [38] developed an analytical method to identify the transient peak temperature in a



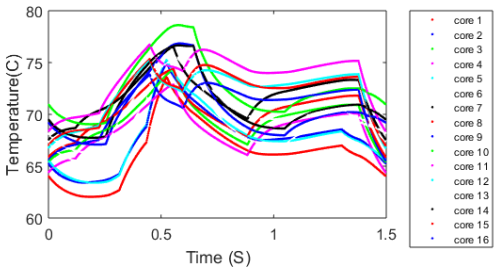
(a) A 16-core platform



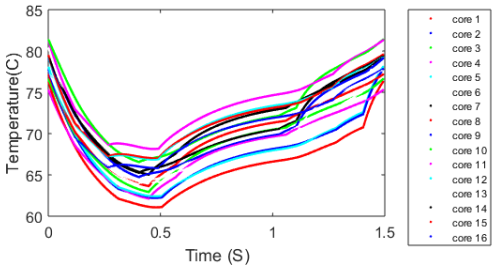
(b) A predefined periodic WCET-schedule.



(c) A hypothetical "step-up execution trace" for bounding the runtime peak temperature.



(d) WCET-schedule temperature trace and $T_{peak} = 78.61°C$



(e) Step-up schedule temperature trace and $T_{peak} = 81.42°C$

Fig. 1. A motivation example on a 16-core platform.

single short time by solving the differential equation array directly. This approach works for small systems and schedules, e.g. with only a few thermal nodes in the system and a small number of state intervals in the schedule. However, when increasing the system and schedule's complexity, the computational cost of [38] grows quickly. As shown in the experimental results in Section 7.3, the computational cost of this approach (i.e. *MatEx*) increases rapidly with system complexity and can quickly exceed the numerical methods. To safely bound the peak temperature, [43] proposes to check all possible scenarios of task arrivals for the critical set of cumulative workload trace, but its computational cost can be prohibitively high as evidenced in our experimental results for the "*WorstTpeak*" method shown in Section 7.3.

To reduce the computational cost, some approximations are applied, e.g. without considering the lateral heat transfer [35] or simply assuming thermal nodes can immediately reach the stable state temperature [36, 57]. However, these approximations can lead to large error margins, which either causes a thermal violation or wastes precious thermal resources.

For the rest of this section, we first use an example to motivate our approach of employing the hypothetical "step-up execution trace" to bound the peak temperature. Then, we present several lemmas and theorems to validate this approach.

5.1 The Motivation Example

In this section, we utilize a motivation example to show that a hypothetical *step-up execution trace* can tightly bound the runtime temperature for any feasible execution trace based on a given periodic schedule.

Table 2. Motivation Example Speed Schedule

	Consecutive execution modes and lengths on each tile				
Tile 1	$(0.65V, 0.315s)$	$(0.95V, 0.12s)$	$(1.25V, 0.09s)$	$(0.9V, 0.78s)$	$(0.75V, 0.195s)$
Tile 2	$(1.05V, 0.12s)$	$(1.15V, 0.435s)$	$(0.85V, 0.33s)$	$(1.1V, 0.495s)$	$(0.75V, 0.12s)$
Tile 3	$(1.1V, 0.255s)$	$(1.25V, 0.39s)$	$(0.95V, 0.405s)$	$(1.2V, 0.375s)$	$(0.8V, 0.075s)$
Tile 4	$(1.05V, 0.225s)$	$(1.25V, 0.225s)$	$(0.95V, 0.15s)$	$(1.05V, 0.78s)$	$(1.1V, 0.12s)$

Note: Each interval is represented by a (supply voltage in V, execution length in sec) pair.

Consider a 4×4 multi-core platform with 4 tiles as shown in Figure 1(a), with all cores on one tile sharing a common voltage supply and following a common periodic schedule. The hyper-period of the schedule is $1.5s$ and there are 5 different running modes on each tile, which leads to 17 state intervals globally in one period as shown in Figure 1(b). The detailed state interval lengths and the execution mode for each state-interval in Figure 1(b) can be found in Table 2. We start the simulation from the ambient temperature 35°C, and run long enough to enter their thermal stable status. More detailed runtime power and thermal parameters can be seen in Section 7.

To identify the peak temperature, a common way is to build a temperature trace numerically for the WCET-schedule by checking the temperature of each small interval. The problem is that its computational time scales linearly with the period length and numerical checking granularity.

Figure 1(d) shows the temperature trace on a 16-core platform with a period of $1.5s$ and a $1ms$ checking granularity with a computational cost of $0.42s$. Instead, we can adopt the approach in [38] to identify the peak temperature by solving the first-order differential equation via $10K$ Newton-Raphson method iterations, and this method takes $52.70s$ for this test case. Moreover, the worst-case temperature guarantee method in [43] needs to greedily search all the possibilities in terms of task arrival, etc. The computational time is longer than 1 hour for this test case.

For a quick response time, some works (e.g. [36, 57]) use the highest stable-state temperature in each state interval to approximate the interval-wise peak temperature, and the overall peak temperature of running a periodic schedule is selected from the highest interval-wise stable-state temperatures, as $T_{peak}(\mathbb{S}(t)) = max(\mathbf{T}_q^{\infty})$, where $q = 1, \cdots, z$. This method simply assumes thermal nodes can immediately reach the stable state temperature. However, this method either cannot sufficiently bound the runtime peak temperature [38] or too pessimistic for a periodic schedule. For example, for the schedule shown in Figure 1(b), the overall highest stable-state temperature is 88.82°C, which is 10.21°C higher than the peak point of the numerical temperature trance in Figure 1(d). The reason is that in practical scenarios the temperature dynamic for in each state-interval of any schedule could not reach its state-state temperature immediately due to thermal capacitance. So, the estimation using the stable-state temperature can be very pessimistic. On the other hand, such simplification may not be able to bound the peak temperature on a multi-core platform [38]. The reason could be due to the multi-core heat transfer and the thermal delay effect, so the highest temperature may not always in sync with the highest power dissipation.

To find a quick and effective temperature bound, we reorder the WCET intervals in Figure 1(b) following a non-decreasing order (step-up) of its voltage levels to build a hypothetical execution trace in Figure 1(c). Its peak temperature is 81.42°C in Figure 1(e). This approach can safely bound the highest temperature of Figure 1(d) by a 2.81°C error margin with a computational cost of no more than $30ms$. Note that due to the real-time constraint, the task execution still follows the schedule in Figure 1(b). The hypothetical "step-up execution trace" in Figure 1(c) is for bounding the runtime peak temperature.

This motivation example shows that a hypothetical "step-up execution trace" can quickly and effectively bound the highest runtime temperature for a given periodic schedule. 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).

5.2 Bounding the Peak Temperature

Identifying and bounding a multi-core peak temperature is challenging in nature because different cores usually execute different speed schedules/power densities at different time instants. With the advancement of real-time scheduling techniques and the system scaling, the intricate power density on multi-core platforms usually shifts the peak temperature away from a scheduling point [18, 38, 47]. So it is difficult to predict where and when could the peak temperature occur. To this end, in this section, we propose and formally prove that a hypothetical "step-up execution trace" can effectively bound the highest runtime temperature for a given periodic schedule.

First, to bound the peak temperature for running a periodic schedule, we prove that the starting temperature of a multi-core schedule does not influence its stable status temperature.

Theorem 5.1. *Let $\mathbb{S}(t) = \{\mathbb{I}_q : q = 1 \cdots z\}$ be a periodic multi-core schedule. Then, the stable-state temperature $\mathbf{T}_{ss}(t)$ achieved by running $\mathbb{S}(t)$ is independent of the initial temperature $\mathbf{T_0}$.*

Theorem 5.1 shows that when identifying the peak temperature in the stable status, we can assume $\mathbf{T_0} = \mathbf{0}$ to ease the presentation in this paper, if otherwise specified. In addition, since the peak temperature occurs in thermal steady status, when all the cores start from the ambient temperature [18], we can infer that the peak temperature of a periodic schedule must occur within one period in the thermal stable status, if the peak temperature does not occur at the starting time as $max(\mathbf{T_0})$. Then, we formally define the "step-up execution trace" for peak temperature bounding purposes as follows.

Definition 5.2. Let $\mathbb{S}_u(t)$ be a periodic schedule and contain z state intervals in a period, with $\mathbf{v}_q$ being the voltage vector for the q_{th} state interval. Then, $\mathbb{S}_u(t)$ is called a ***step-up execution trace*** if $\mathbf{v}_q \leq \mathbf{v}_{q+1}, \forall q \in \{1, \cdots, z-1\}$.

According to Definition 5.2, the voltage for each core is monotonically non-decreasing from the first to the last state interval. The rationale of the "step-up execution trace" is that it aggregates a higher power dissipation toward the end of the period to achieve the highest instantaneous heat dissipation, which can simplify the peak temperature identification as follows.

Theorem 5.3. *The peak temperature when repeating a step-up execution trace $\mathbb{S}_u(t)$ periodically from the ambient temperature occurs at the end of the period in the stable status.*

Theorem 5.3 greatly simplifies the peak temperature prediction comparing to existing methods (e.g. [18, 38, 43]). Based on (3) and (4), we can quickly identify the peak temperature of $\mathbb{S}_u(t)$ with linear complexity.

Theorem 5.4. *Given a periodic schedule $\mathbb{S}(t)$, the corresponding step-up execution trace $\mathbb{S}_u(t)$ bounds the peak temperature of $\mathbb{S}(t)$.*

This theorem can be interpreted based on the facts that the multi-core thermal model presented in (1) is a linear time-invariant system [2, 43], which follows the superposition principle: (i) The thermal impact at one time instant is the sum of the thermal impact by each core; (ii) The thermal impact of each core is the sum of the impact by each state interval in the schedule. So, without changing other factors, as a high-speed interval moves toward the end of a periodic schedule, the temperature at the end of the schedule tends to increase.

Based on Theorem 5.3 and 5.4, given a periodic schedule $\mathbb{S}(t)$ and its corresponding step-up execution trace $\mathbb{S}_u(t)$, as long as all tasks' deadlines can be guaranteed by $\mathbb{S}(t)$ and the peak temperature constraints can be guaranteed by $\mathbb{S}_u(t)$, then both the timing and peak temperature constraints are guaranteed if $\mathbb{S}(t)$ is executed periodically.

6 PEAK TEMPERATURE MINIMIZATION

Peak temperature minimization is one of the primary design objectives in the IC industry. Nowadays, the escalating software complexity can easily make heat generation exceed the capability of thermal dissipation systems, which are often designed with narrow or even negative margins for cost reason. In section 5, it has been proved that the runtime peak temperature can be effectively bounded by using a hypothetical *step-up execution trace*. In this section, we discuss how to employ the *m-Oscillating* methods [45] to reduce the peak temperature bound and improve the real-time service output on a multi-core platform.

The m-Oscillating scheduling method, as introduced in [25] for a single-processor platform, 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 the m-Oscillating scheduling method for an individual core or part of cores on a multicore platform cannot always reduce the peak temperature.

We set up a 3-core platform with similar settings as shown in Section 7 and capture the thermal dynamics in the stable status. In Fig. 2(a), the period is $2.4s$. Each core runs at equal times in two processing modes, with high-voltage $v_H = 1.3V$ and low-voltage $v_L = 0.6V$. Fig. 2(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. 2(b). In the stable status, as shown in Fig. 2(d), the peak temperature becomes 79.86°C, which is higher than in Fig. 2(c). This example clearly shows that the frequency oscillation scheme performed only on one core does not necessarily reduce the peak temperature in a multi-core platform. In regard to this, we formally define the *m-Oscillating schedule for a multi-core platform* as follows.

Definition 6.1. Let $\mathbb{S}(t)$ be a periodic schedule on a multi-core processor. The corresponding ***m-oscillating schedule***, denoted as $\mathbb{S}(m, t)$, is the one that scales down the length of each state interval in $\mathbb{S}(t)$ by m times without changing its voltage levels.

Figure 3 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. Essentially, m-Oscillating is a reverse utilization of Theorem 5.3 and 5.4 by temporally even out the workload for mitigating the power/heat aggregation and minimizing the peak temperature.

We conduct an empirical study of the m-Oscillating approach on a 3-core platform of a 3×1 topology, with applications chosen from MiBench benchmark [17]. The power and thermal dynamics are abstracted from Mcpat [28] and HotSpot simulator [50], respectively. The workloads of *cjpeg*, *djpeg* and *h263* are partitioned in the order of core 1 to core 3. Let the processor run a two-speed step-up execution trace with period $t_p = 0.6s$, starting from ambient temperature $T_{amb} = 35\,^{\circ}\text{C}$. On each core, the high and low-frequency modes are set to 3 and 1.8 GHz and execute $0.3s$ for each mode within one period, respectively. In Fig. 4, by applying m-Oscillating approach, the peak temperature reduces from 72.73 °C when $m = 1$ to 63.74 °C when $m = 3$, when completing the same amount of workload. We formally formulate the conclusion in the following theorem.

Theorem 6.2. *Let $\mathbb{S}_u(t) = \{\mathbb{I}_q : q = 1 \cdots z\}$ be a periodic step-up execution trace of any periodic schedule on a multi-core processor. Then, $T_{peak}(\mathbb{S}_u(m_1, t)) \geq T_{peak}(\mathbb{S}_u(m_2, t))$, if $1 \leq m_1 \leq m_2$.*

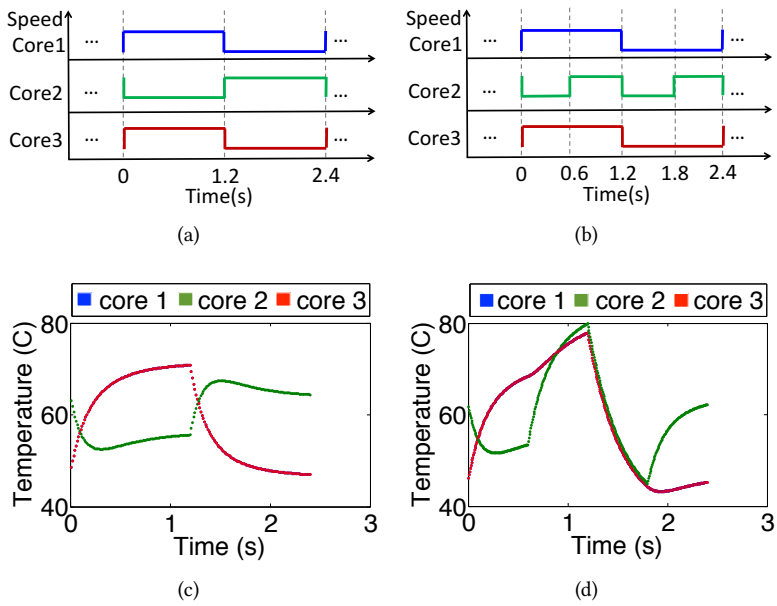


Fig. 2. (a) Core 1 and core 3 run at $1.3V$ within $[0, 1.2]s$ and $0.6V$ within $[1.2, 2.4]s$; Core 2 runs at $0.6V$ within $[0, 1.2]s$ and $1.3V$ within $[1.2, 2.4]s$. (b) Core 2 doubles its oscillating frequency from the schedule in Fig.2(a). (c) The stable status temperature trace for the schedule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

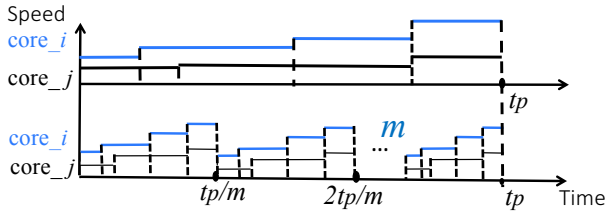


Fig. 3. Illustration of m-Oscillating schedule [46].

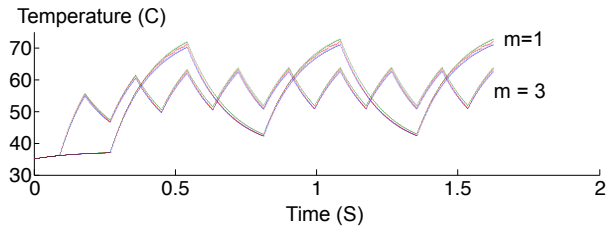


Fig. 4. Peak temperature reduction by m-Oscillating approach.

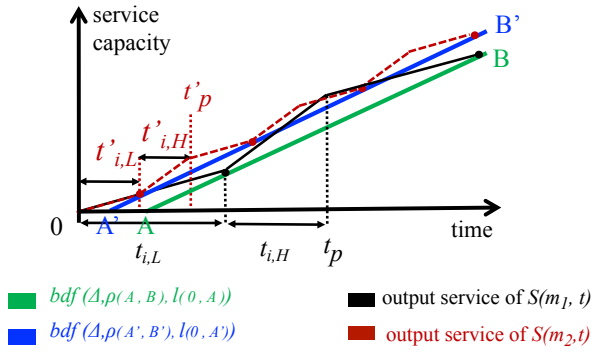


Fig. 5. Real-time calculus illustration of Theorem 6.3: the bounded-delay approximation shows a higher service bound of $\mathbb{S}(m_2, t)$ than $\mathbb{S}(m_1, t)$, if $m_2 \geq m_1 \geq 1$.

Theorem 6.2 indicates that when DVFS transition overhead is small enough to be negligible, the peak temperature monotonically decreases with the increase of m. In practical scenarios, when considering the DVFS overhead, an appropriate value of m needs to be judiciously selected as a function of the transition overhead as Algorithm 2 in [45]. Moreover, *m-Oscillating* also helps to improve the system's real-time service throughput under a peak temperature constraint, i.e. requests that can be completed without violating their timing and thermal constraints.

Consider a temperature-constrained real-time system with irregularly arrived events, to enhance the guaranteed real-time service and feasibility, we apply the *m-Oscillating* method to build a periodic server, which can fully utilize the thermal resources. Meanwhile, its low-bound of the system service throughput can be guaranteed by the bounded-delay function [26]. The rationale is that for completing the same amount of workload, *m-Oscillating* method results in a peak temperature no higher than the original schedule. Meanwhile, the bounded-delay [26] becomes smaller when increasing the oscillating frequency as formulated in the following theorem.

Theorem 6.3. *Let $\mathbb{S}_u(t)$ be a periodic step-up execution trace on a multi-core platform under a peak temperature constraint T_{max}. Using the m-Oscillating approach, $\mathbb{S}_u(m_2, t)$ is able to output a service bound that is no lower than $\mathbb{S}_u(m_1, t)$, if $m_2 \geq m_1 \geq 1$.*

As shown in Figure 5, without losing generality, $\mathbb{S}_u(t)$ is assumed to contain two speeds. Since $m_2 \geq m_1 \geq 1$, we have $T_{peak}(\mathbb{S}_u(m_2, t)) \leq T_{peak}(\mathbb{S}_u(m_1, t)) \leq T_{peak}(\mathbb{S}_u(t)) \leq T_{max}$ (Theorem 6.2).

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 $\mathbb{S}_u(t)$, respectively. Then, the bounded-delay function of the i-th node of $\mathbb{S}_u(m_1, t)$ is $bdf(\Delta, \rho(A, B), l(0, A))$ [26], 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 $\mathbb{S}_u(m_2, t)$. When transition overhead is negligible, the bounded-delay function is featured by

$$\rho(A, B) = \frac{\rho_L t_{i,L}/m + \rho_H t_{i,H}/m}{(t_{i,L} + t_{i,H})/m} \tag{7}$$

$$l(0, A) = \frac{(\rho_H - \rho_L) \cdot t_{i,H}/m \cdot t_{i,L}/m}{(\rho_H t_{i,H} + \rho_L t_{i,L})/m} \tag{8}$$

Since $l(0, A') \leq l(0, A)$ and $\rho(A', B') \not< \rho(A, B)$, the service bound of $\mathbb{S}_u(m_2, t)$ is no lower than $\mathbb{S}_u(m_1, t)$.

Theorem 6.3 implies that, when using the *m-Oscillating* approach, the larger the value of m is, the higher the server capacity becomes. However, this conclusion is built upon the assumption that the execution mode transition overhead is small and can be ignored.

When transition overhead is significant, service rates for $S_u(m_1, t)$ and $S_u(m_2, t)$ are different when they adopt the same running mode. Therefore, increasing m does not necessarily always lead to improved real-time service bound.

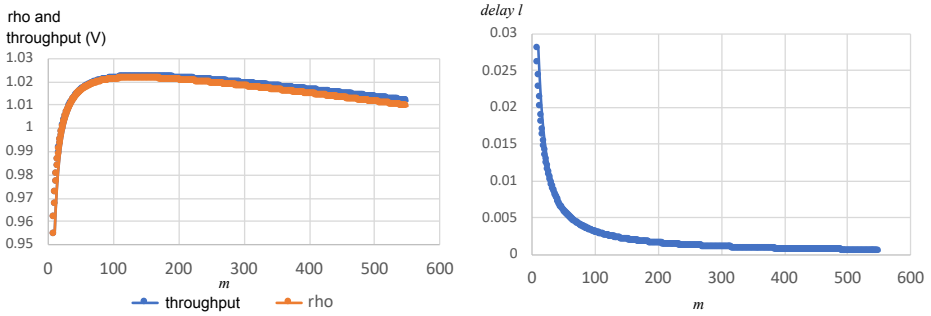
Let τ be the transition overhead on the given platform and let $bdf(\Delta, \rho(\tilde{A}, \tilde{B}, \tau), l(0, \tilde{A}, \tau))$ be the bounded-delay function of the i-th node of $\mathbb{S}_u(m, t)$ when considering the transition overhead τ. To construct the m-Oscillating schedule, we need to compensate for the performance loss caused by each DVFS transition stalls, i.e. $(\rho_H + \rho_L)\tau$. So, a small interval of $\delta = \frac{(\rho_H+\rho_L)\tau}{\rho_H-\rho_L}$ needs to be shifted from low-speed to high-speed (as shown in [46] Figure 6a). To maximize the system performances, we first find the "optimal m", which leads to the lowest peak temperature for a given schedule. Then, we judiciously increase the high-speed interval lengths until approaching the maximally allowed temperature [46]. When considering transition overhead, the bounded-delay function is featured by

$$\rho(\tilde{A}, \tilde{B}, \tau) = \frac{\rho_L(t_{i,L}/m - \delta) + \rho_H(t_{i,H}/m + \delta)}{(t_{i,L} + t_{i,H})/m + 2\tau}, \tag{9}$$

and

$$l(0, \tilde{A}, \tau) = \frac{\rho_H(t_{i,H}/m + \delta)(t_{i,L}/m + \tau - \delta) - \rho_L(t_{i,L}/m - \delta)(t_{i,H}/m + \tau + \delta)}{\rho_L(t_{i,L}/m - \delta) + \rho_H(t_{i,H}/m + \delta)}. \tag{10}$$

In what follows, we conduct an empirical study under the same power and thermal configurations as shown in Section 7. To validate Theorem 6.3, we set a two-speed schedule whose low-/high-speeds are supported by $0.6V$ and $1.3V$, respectively. The maximally allowed temperature is set to 70°C. The period is $1.5sec$ and high-/low- speed takes 80% and 20% of the period length without considering overhead, respectively. Then, we incorporate DVFS transition overhead and conduct m-Oscillating approach (Algorithm 2 in [45]). The system throughput is defined as the average speed within one period, which is defined in Equation 5 of [45].



(a) $\rho(\tilde{(A)}, \tilde{B}, \tau)$ and throughput (V)

(b) $l(0, \tilde{A}, \tau)$

Fig. 6. Results for bounded-delay approximation when considering DVFS overhead

From Figure 6(a), we can see that the throughput performance is monotonically increasing with m first and the maximal throughput is achieved when $m = 161$. Meanwhile, factor $\rho(\tilde{A}, \tilde{B}, \tau)$ in (9) reaches its maximum value. Then, both throughput and factor $\rho(\tilde{A}, \tilde{B}, \tau)$ decrease with m.

From Figure 6(b), we can see that $l(0, \tilde{A}, \tau)$ in (10) monotonically decreases with m. Overall, when incorporating transition overhead, the m-Oscillating approach is able to output the maximum service bound.

7 EXPERIMENTAL RESULTS

In this section, we focus on simulating a series of peak temperature bounding methodologies on 4 multi-core configurations, i.e. 2×3, 3×3, 3×4 and 4×4 corresponding to 6, 9, 12, 16-core, respectively. Each core size is $4 \times 4mm^2$ and follows the Alpha 21264 floorplan and there are fifteen available speed levels supported by discrete supply voltages from $0.6V$ to $1.3V$ with a $0.05V$ step-length.

The thermal and power parameters are abstracted from HotSpot 5.02 [27] and the McPAT simulator [28]. The total power consumption (P) is composed of dynamic power and leakage power. 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. In particular, we adopt the curve-fitting constants [18] in the power model as: 1) $\alpha = 0.84$; 2) $\beta = 0.0163$; and 3) $\gamma = 7.2564$. The ambient temperature was set to be $T_{amb} = 35\,^\circ$C, unless otherwise specified. In this paper, we assume the WCET-schedules are given, so there should be free of any deadline miss case.

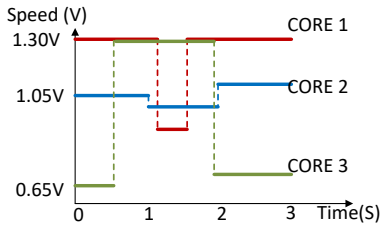
There are five approaches in this experiment: (1) Numerical method (**NM**) splits each execution interval into small pieces and finds the peak temperature by checking each small step length (e.g. [40, 47, 50]). (2) Stable State Temperature Check method (**TssCheck**) checks the peak temperature interval-by-interval using a combination of the analytical and numerical approaches [18]. (3) Maximum Transient Temperature approach (**MatEx**) uses Newton-Raphson method to solve the first-order derivative on each core in (1) as shown in [38]. (4) Worst-Case Temperature Guarantees method (**WorstTpeak**) uses an analytical method to calculate the upper bound via building the thermal critical instance under all possible scenarios of task executions in [43]. (5) The last one is our proposed method, i.e. constructing a step-up execution trace (**StepUp**) to bound the peak temperature of different runtime execution traces, as defined in Definition 5.2.

In the following experiments, the checking step length used in **NM**, **TssCheck** and **WorstTpeak** are set as 1 ms, and the number of iterations used when applying the Newton-Raphson method by **MatEx** is set as $10K$, unless otherwise specified.

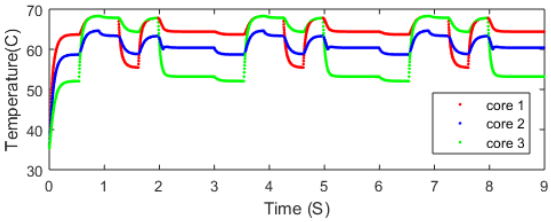
7.1 The step-up execution trace can bound the peak temperature

First, we utilize random schedules to validate that a hypothetical *step-up execution trace* can effectively bound predefined WCET-schedule. To put the runtime thermal dynamics into perspective, we select a 3-core worst-case execution time schedule with a period of 3 seconds given in Figure 7(a). Core 1 runs at the mode with the supply voltage of $1.5V$ for $1.26s$, then turns to $0.9V$ for $0.36s$ and executes at $1.5V$ again for $1.38s$. Core 2 runs at $1.05V$ for $0.9s$, then turns to $1.0V$ for $1.17s$ and executes at $1.1V$ for $0.93s$. Core 3 runs at $0.65V$ for $0.54s$, then turns to $1.3V$ for $1.44s$ and executes at $0.7V$ for $1.02s$.

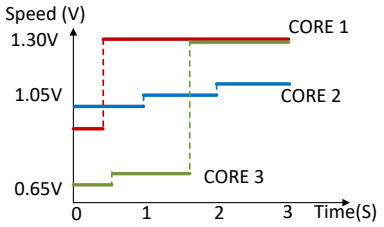
Figure 7(b) shows that the peak temperature for executing the schedule in Figure 7(a) is 68.2907°C. If we reorder the WCET intervals in Figure 7(a) following the *step-up execution trace* (in Definition 5.2) as shown in Figure 7(c), the peak temperature is calculated as 68.8824°C in Figure 7(d), which is higher than the peak temperature in Figure 7(b) (conform to Theorem 5.4). In addition, the peak temperature of a step-up execution trace occurs at the end of the period in Figure 7(d) (conform to Theorem 5.3). We also profiled the temperature trace in the thermal stable status on 6, 9 and 16-core architectures as shown in Figure 8. We can observe that all the peak temperatures in the step-up execution trace must occur at the end of the period, which conforms to Theorem 5.3.



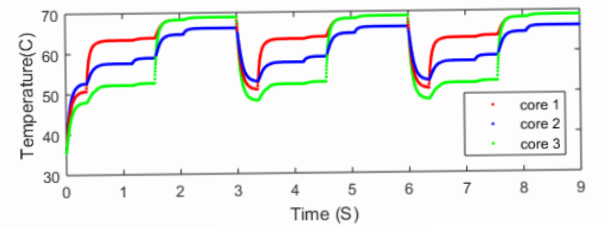
(a) A predefined periodic WCET-schedule.



(b) $T_{peak} = 68.2907°C$



(c) A hypothetical "step-up execution trace" for bounding the runtime peak temperature.



(d) $T_{peak} = 68.8824°C$

Fig. 7. An illustration of Theorem 5.3 on a 3-core platform.

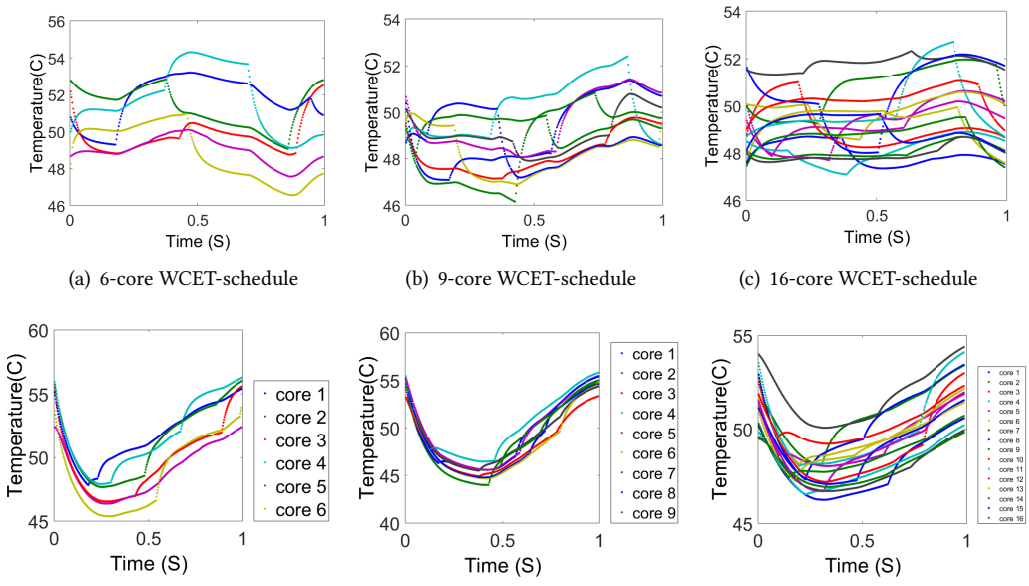
Meanwhile, we can also observe that the peak temperature of the *step-up execution trace* must be higher than WCET-schedule, which conforms to Theorem 5.4. More intensive experiments are shown in what follows regarding how effectively can a *step-up execution trace* bound the peak temperature.

7.2 The tightness of the temperature bound by StepUp method

In this section, we arbitrarily generate random WCET-schedules for each configuration by a different number of cores and different period lengths. Each schedule may contain up to 20 state intervals. Then, we use the peak temperature differences between the step-up execution traces and WCET-schedules to represent the bounding tightness of the **StepUp** methodology. In Figure 9, each bar represents the average peak temperature overestimation for $2K$ random WCET-schedules under a given configuration.

In general, the peak temperature overestimation is proportional to the period length. For example in Figure 9, on a 6-core platform, the overestimation is 0.17°C, 0.60°C and 0.85°C for the periods equal to $10ms$, $50ms$ and $100ms$, respectively. As the period becomes $0.5s$, $1s$ and $5s$, the overestimation shows 3.26°C, 5.22°C and 5.82°C, respectively, which has significantly larger margins than those cases with smaller period lengths. The reason is that the larger the period, the more likely those high-speed intervals may stack longer to the end of the period, and, thus, causes a higher peak temperature in the hypothetical step-up schedule.

However, for 3-core, 9-core and 16-core cases, the overestimation may not strictly increase with the period length. For example, on the 16-core platform, the overestimation is 1.73°C when the period equals $5s$, which is smaller than 3.12°C when the period equals $1s$. The reason can be that although the randomly generated WCET-schedules exhibit intricate power dissipation and heat transfer, its corresponding hypothetical step-up schedule may not always significantly worsen the power aggregation as periods become larger.



(a) 6-core WCET-schedule (b) 9-core WCET-schedule (c) 16-core WCET-schedule

(d) 6-core step-up execution trace (e) 9-core step-up execution trace (f) 16-core step-up execution trace

Fig. 8. The peak temperature of a step-up execution trace always occur at the end of its period (Theorem 5.3).

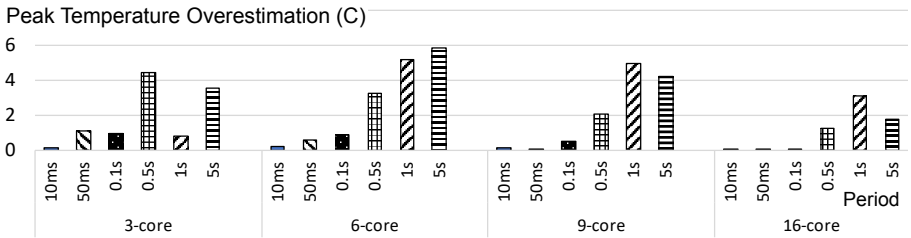


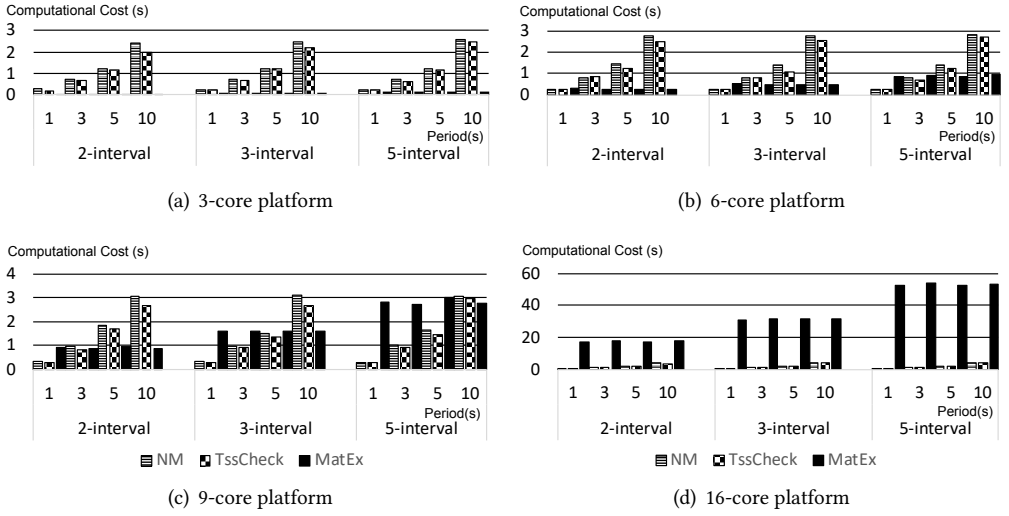
Fig. 9. The overestimation of the temperature bound by **StepUp** method.

It is also interesting to see that for any fixed period length, the average overestimation does not increase with the number of cores. For example, when the period is fixed at 0.5s, the error margins are 2.04°C and 1.26°C on 9-core and 16-core platforms, respectively, which are smaller than the average 3.26°C on a 6-core platform.

Overall, the average peak temperature overestimation by the 48K cases shown in Figure 9 is as tight as 1.88°C. The largest average overestimation 5.82°C occurs on a 6-core platform, when period equals 5s. The smallest average overestimation 0.017°C occurs on a 16-core platform, when period equals 100ms.

7.3 Computational cost comparison of different approaches

In this section, we first compare the computational cost of **NM**, **TssCheck** and **MatEx** approaches in Fig. 10. The computational time of the **NM** approach is proportional to the period length and inverse proportional to the numerical checking step length. For example, in Fig. 10(b), on a 6-core



(a) 3-core platform

(b) 6-core platform

(c) 9-core platform

(d) 16-core platform

Fig. 10. Computational cost comparison of the **NM**, **TssCheck** and **MatEx** approaches

platform with 5 tasks on each core, the **NM** computational cost increases from 0.272 to 2.848 seconds, when t_p increases from 1 to 10 seconds. The complexity of the **NM** approach is $O(\frac{t_p}{\text{step length}})$.

The second approach **TssCheck** can be seen as a refined **NM** approach by eliminating the numerical checking workloads on those intervals whose peak temperature must occur at the scheduling points (Line 3 - 8 of Algorithm 1 in [18]). For example, in Fig. 10(a), on a 3-core platform with $t_p = 10$ and 5 tasks on each core, the **TssCheck** reduces computational cost from 2.604 in the **NM** approach to 2.46 seconds. The **TssCheck** averagely saves the computational cost of the **NM** approach by 9.55, 9.24, 8.92 and 6.47 percent on 3, 6, 9 and 16-core platforms, respectively. The **TssCheck**'s complexity lower bound and upper bound are $O(\text{number of intervals})$ and $O(\frac{t_p}{\text{step length}})$, respectively.

The third approach **MatEx** is different from **NM** and **TssCheck**, because its computational cost is not related to the period length or the checking step-length. Instead, the computational overhead of **MatEx** closely relates to the number of intervals on each core, the number of cores, and the number of iterations used when applying the Newton-Raphson method [38]. For example, on 3 and 6-core platforms, **MatEx** is significantly computational efficient comparing to **NM** and **TssCheck** methods as shown in Fig. 10(a) and Fig. 10(b). In Fig. 10(b), on a 6-core platform with $t_p = 10$ and 3 different modes on each core, the **MatEx** method only uses 0.464 seconds in comparison with 2.772 and 2.558 seconds of **NM** and **TssCheck** methods, respectively. However, in Fig. 10(d), on a 16-core platform with $t_p = 10$ and 3 different modes on each core, the **MatEx** method spends as much as 31.866 seconds, while **NM** and **TssCheck** methods take 4.270 and 4.056 seconds, respectively. The complexity of **MatEx** is $O(N^3) + \sum_{i=1}^{N}[O(N^2) \times \text{Newton-Raphson iterations} \times \text{number of intervals on core i}]$.

Next, we compare the computational costs of two peak temperature bounding methodologies, i.e., **WorstTpeak** and our proposed **StepUp** approaches. In this experiment, we let each core has a fixed number of 3 different modes. The computational cost is shown in Table 3. We can see that **StepUp** saves orders of magnitudes computational costs than **WorstTpeak**. The complexity of **WorstTpeak** and our proposed **StepUp** are $O(\prod_{i=1}^{N}[\text{factorial of the number of intervals on core-i}] \times \frac{t_p}{\text{step length}})$ and $O(1)$, respectively.

Table 3. Computational Cost of the **WorstTpeak** and **StepUp** Approach

	3-Core				6-Core			
$t_p(s)$	**1**	**3**	**5**	**10**	**1**	**3**	**5**	**10**
StepUp[1]	6	4	2	8	1	1	8	1
WorstTpeak[2]	0.27	0.82	1.36	2.87	1.21	3.27	5.39	10.9
	9-Core				**16-Core**			
$t_p(s)$	**1**	**3**	**5**	**10**	**1**	**3**	**5**	**10**
StepUp[1]	16	14	16	16	30	30	36	38
WorstTpeak[2]	2.4	7.37	11.6	23.4	7.29	22.2	36.6	73.5

[1]. The unit is 10^{-3} second.
[2]. The unit is 10^5 second. Due to the long computational time, the results of **WorstTpeak** are mostly projected upon the computational complexity except for the underlined values.

Overall, we are able to conclude that our proposed **StepUp** approach not only can bound the peak temperature for any given WCET-schedule, but also it is computationally efficient, which can be easily applied to both proactive and reactive temperature-aware design.

8 CONCLUSIONS

As IC technology continues to progress, power density keeps increasing and results in soaring chip temperatures. While there have been extensive thermal-aware research approaches developed, we believe that the great challenges of thermal problems and Quality of Service guarantee requirement in design of real-time systems demand a more rigorous analytical study of the thermal problem. In this paper, we present a series of fundamental and provable principles related to thermal models, peak temperature bounding and peak temperature reduction on multi-core platforms. These principles are general enough to benefit future thermal-aware analyses on 2D and 3D multi-core platforms.

REFERENCES

[1] A. Agrawal, J. Torrellas, and S. Idgunji. 2017. Xylem: Enhancing Vertical Thermal Conduction in 3D Processor-Memory Stacks. In *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 546–559.
[2] R. Ahmed, P. Ramanathan, and K.K. Saluja. 2014. Necessary and Sufficient Conditions for Thermal Schedulability of Periodic Real-Time Tasks. In *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*. 243–252. DOI: http://dx.doi.org/10.1109/ECRTS.2014.15
[3] H. Anton. 2014. *Elementary Linear Algebra, Applications Version 11E with WileyPlus Card.* John Wiley & Sons, Incorporated. https://books.google.com/books?id=WEuloAEACAAJ
[4] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. 2013. Thermal and Energy Management of High-Performance Multicores: Distributed and Self-Calibrating Model-Predictive Controller. *IEEE Transactions on Parallel and Distributed Systems* 24, 1 (Jan 2013), 170–183. DOI: http://dx.doi.org/10.1109/TPDS.2012.117
[5] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini. 2016. Cooling-aware node-level task allocation for next-generation green HPC systems. In *2016 International Conference on High Performance Computing Simulation (HPCS)*. 690–696. DOI: http://dx.doi.org/10.1109/HPCSim.2016.7568402
[6] F. Beneventi, A. Bartolini, A. Tilli, and L. Benini. 2014. An Effective Gray-Box Identification Procedure for Multicore Thermal Modeling. *IEEE Trans. Comput.* 63, 5 (May 2014), 1097–1110. DOI: http://dx.doi.org/10.1109/TC.2012.293
[7] William L. Brogan. 1985. *Modern Control Theory* (second ed.). Ergodebooks, Richmond, TX.
[8] T. Chantem, X. S. Hu, and R. P. Dick. 2011. Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19, 10 (Oct 2011), 1884–1897.

DOI: http://dx.doi.org/10.1109/TVLSI.2010.2058873

[9] Vivek Chaturvedi, Huang Huang, Shangping Ren, and Gang Quan. 2012. On the Fundamentals of Leakage Aware Real-time DVS Scheduling for Peak Temperature Minimization. *J. Syst. Archit.* 58, 10 (Nov. 2012), 387–397. DOI: http://dx.doi.org/10.1016/j.sysarc.2012.08.002

[10] Y. Chen, E. Kursun, D. Motschman, C. Johnson, and Y. Xie. 2011. Analysis and mitigation of lateral thermal blockage effect of through-silicon-via in 3D IC designs. In *IEEE/ACM International Symposium on Low Power Electronics and Design*. 397–402. DOI: http://dx.doi.org/10.1109/ISLPED.2011.5993673

[11] A. K. Coskun, J. L. Ayala, D. Atienza, T. S. Rosing, and Y. Leblebici. 2009. Dynamic thermal management in 3D multicore architectures. In *2009 Design, Automation Test in Europe Conference Exhibition*. 1410–1415. DOI: http://dx.doi.org/10.1109/DATE.2009.5090885

[12] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli. 2014. Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. DOI: http://dx.doi.org/10.1145/2593069.2593199

[13] Songchun Fan, Seyed Majid Zahedi, and Benjamin C. Lee. 2016. The Computational Sprinting Game. *SIGOPS Oper. Syst. Rev.* 50, 2 (March 2016), 561–575. DOI: http://dx.doi.org/10.1145/2954680.2872383

[14] Nathan Fisher, Jian-Jia Chen, Shengquan Wang, and Lothar Thiele. 2011. Thermal-aware Global Real-time Scheduling and Analysis on Multicore Systems. *J. Syst. Archit.* 57, 5 (May 2011), 547–560. DOI: http://dx.doi.org/10.1016/j.sysarc.2010.09.010

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

[16] C. E. Green, A. G. Fedorov, and Y. K. Joshi. 2012. Dynamic thermal management of high heat flux devices using embedded solid-liquid phase change materials and solid state coolers. In *13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*. 853–862.

[17] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. 2001. 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)*. 3–14. DOI: http://dx.doi.org/10.1109/WWC.2001.990739

[18] Q. Han, M. Fan, O. Bai, S. Ren, and G. Quan. 2016. Temperature-Constrained Feasibility Analysis for Multi-core Scheduling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* PP, 99 (2016), 1–1. DOI: http://dx.doi.org/10.1109/TCAD.2016.2543020

[19] Vinay Hanumaiah, Digant Desai, Benjamin Gaudette, Carole-Jean Wu, and Sarma Vrudhula. 2014. STEAM: A Smart Temperature and Energy Aware Multicore Controller. *ACM Trans. Embed. Comput. Syst.* 13, 5s, Article 151 (Oct. 2014), 25 pages. DOI: http://dx.doi.org/10.1145/2661430

[20] V. Hanumaiah and S. Vrudhula. 2011. Reliability-aware thermal management for hard real-time applications on multi-core processors. In *2011 Design, Automation Test in Europe*. 1–6. DOI: http://dx.doi.org/10.1109/DATE.2011.5763032

[21] V. Hanumaiah and S. Vrudhula. 2012. Temperature-Aware DVFS for Hard Real-Time Applications on Multicore Processors. *IEEE Trans. Comput.* 61, 10 (Oct 2012), 1484–1494. DOI: http://dx.doi.org/10.1109/TC.2011.156

[22] V. Hanumaiah and S. Vrudhula. 2014. Energy-Efficient Operation of Multicore Processors by DVFS, Task Migration, and Active Cooling. *IEEE Trans. Comput.* 63, 2 (Feb 2014), 349–360. DOI: http://dx.doi.org/10.1109/TC.2012.213

[23] V. Hanumaiah, S. Vrudhula, and K.S. Chatha. 2011. 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 (Nov 2011), 1677–1690. DOI: http://dx.doi.org/10.1109/TCAD.2011.2161308

[24] M. Healy, M. Vittes, M. Ekpanyapong, C. S. Ballapuram, S. K. Lim, H. S. Lee, and G. H. Loh. 2007. Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26, 1 (Jan 2007), 38–52. DOI: http://dx.doi.org/10.1109/TCAD.2006.883925

[25] Huang Huang, Vivek Chaturvedi, Gang Quan, Jeffrey Fan, and Meikang Qiu. 2014. Throughput Maximization for Periodic Real-time Systems Under the Maximal Temperature Constraint. *ACM Trans. Embed. Comput. Syst.* 13, 2s, Article 70 (Jan. 2014), 22 pages. DOI: http://dx.doi.org/10.1145/2544375.2544390

[26] K. Huang, L. Santinelli, J. J. Chen, L. Thiele, and G. C. Buttazzo. 2009. 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*. 6224–6231. DOI: http://dx.doi.org/10.1109/CDC.2009.5400034

[27] Wei Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan. 2006. HotSpot: a compact thermal modeling methodology for early-stage VLSI design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 14, 5 (May 2006), 501–513. DOI: http://dx.doi.org/10.1109/TVLSI.2006.876103

[28] Sheng Li, Jung Ho Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, and N.P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*. 469–480.

[29] C. H. Liao and C. H. P. Wen. 2015. 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 (Nov 2015), 2719–2723. DOI: http://dx.doi.org/10.1109/TVLSI.2014.2360802

[30] M. Marcus and H. Minc (Eds.). 1964. *A Survey of Matrix Theory and Matrix Inequalities.* Allyn and Bacon, Boston, MA, USA.

[31] J. Meng, K. Kawakami, and A. K. Coskun. 2012. Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. In *DAC Design Automation Conference 2012*. 648–655.

[32] Carl D. Meyer (Ed.). 2000. *Matrix Analysis and Applied Linear Algebra.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[33] J. Minz, E. Wong, and Sung Kyu Lim. 2005. Reliability-aware floorplanning for 3D circuits. In *Proceedings 2005 IEEE International SOC Conference.* 81–82. DOI: http://dx.doi.org/10.1109/SOCC.2005.1554461

[34] F. Mulas, D. Atienza, A. Acquaviva, S. Carta, L. Benini, and G. De Micheli. 2009. Thermal Balancing Policy for Multiprocessor Stream Computing Platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 12 (Dec 2009), 1870–1882. DOI: http://dx.doi.org/10.1109/TCAD.2009.2032372

[35] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. De Micheli. 2007. Temperature-aware processor frequency assignment for MPSoCs using convex optimization. In *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2007 5th IEEE/ACM/IFIP International Conference on.* 111 –116.

[36] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. De Micheli, and R. Gupta. 2009. Processor Speed Control With Thermal Constraints. *IEEE Transactions on Circuits and Systems I: Regular Papers* 56, 9 (Sept 2009), 1994–2008. DOI: http://dx.doi.org/10.1109/TCSI.2008.2011589

[37] S. Pagani, J. Chen, and J. Henkel. 2015a. Energy and Peak Power Efficiency Analysis for the Single Voltage Approximation (SVA) Scheme. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 9 (Sept 2015), 1415–1428. DOI: http://dx.doi.org/10.1109/TCAD.2015.2406862

[38] Santiago Pagani, Jian-Jia Chen, Muhammad Shafique, and Jörg Henkel. 2015b. MatEx: Efficient Transient and Peak Temperature Computation for Compact Thermal Models. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE '15)*. EDA Consortium, San Jose, CA, USA, 1515–1520. http://dl.acm.org/citation.cfm?id=2757012.2757162

[39] S. Pagani, H. Khdr, J. J. Chen, M. Shafique, M. Li, and J. Henkel. 2017. Thermal Safe Power (TSP): Efficient Power Budgeting for Heterogeneous Manycore Systems in Dark Silicon. *IEEE Trans. Comput.* 66, 1 (Jan 2017), 147–162. DOI: http://dx.doi.org/10.1109/TC.2016.2564969

[40] R. Rao and S. Vrudhula. 2009. 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 (Oct 2009), 1559–1572. DOI: http://dx.doi.org/10.1109/TCAD.2009.2026361

[41] M. M. Sabry, A. K. Coskun, D. Atienza, T. Rosing, and T. Brunschwiler. 2011. 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 (Dec 2011), 1883–1896. DOI: http://dx.doi.org/10.1109/TCAD.2011.2164540

[42] S. Saha, Y. Lu, and J. S. Deogun. 2012. Thermal-Constrained Energy-Aware Partitioning for Heterogeneous Multi-core Multiprocessor Real-Time Systems. In *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.* 41–50. DOI: http://dx.doi.org/10.1109/RTCSA.2012.15

[43] Lars Schor, I. Bacivarov, Hoeseok Yang, and L. Thiele. 2012. Worst-Case Temperature Guarantees for Real-Time Applications on Multi-core Systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th.* 87–96. DOI: http://dx.doi.org/10.1109/RTAS.2012.14

[44] Shi Sha, Wujie Wen, Gustavo A. Chaparro-Baquero, and Gang Quan. 2019. Thermal-constrained energy efficient real-time scheduling on multi-core platforms. *Parallel Comput.* 85 (2019), 231 – 242. DOI: http://dx.doi.org/https://doi.org/10.1016/j.parco.2019.01.003

[45] S. Sha, W. Wen, M. Fan, S. Ren, and G. Quan. 2016. Performance Maximization via Frequency Oscillation on Temperature Constrained Multi-core Processors. In *2016 45th International Conference on Parallel Processing (ICPP).* 526–535. DOI: http://dx.doi.org/10.1109/ICPP.2016.67

[46] S. Sha, W. Wen, S. Ren, and G. Quan. 2018. M-Oscillating: Performance Maximization on Temperature-Constrained Multi-Core Processors. *IEEE Transactions on Parallel and Distributed Systems* 29, 11 (Nov 2018), 2528–2539. DOI: http://dx.doi.org/10.1109/TPDS.2018.2835474

[47] S. Sharifi, D. Krishnaswamy, and T. Rosing. 2013. PROMETHEUS: A Proactive Method for Thermal Management of Heterogeneous MPSoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 7 (July 2013), 1110–1123. DOI: http://dx.doi.org/10.1109/TCAD.2013.2247656

[48] Hafiz Fahad Sheikh, Ishfaq Ahmad, Zhe Wang, and Sanjay Ranka. 2012. An overview and classification of thermal-aware scheduling techniques for multi-core processing systems. *Sustainable Computing: Informatics and Systems* 2, 3 (2012), 151 – 169. DOI: http://dx.doi.org/10.1016/j.suscom.2011.06.005

[49] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel. 2016. 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 (Aug 2016), 2745–2758.

[50] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, Wei Huang, Sivakumar Velusamy, and David Tarjan. 2004. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim.* 1, 1 (March 2004), 94–125. DOI: http://dx.doi.org/10.1145/980152.980157

[51] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschwiler. 2014. 3D-ICE: A Compact Thermal Model for Early-Stage Design of Liquid-Cooled ICs. *IEEE Trans. Comput.* 63, 10 (Oct 2014), 2576–2589. DOI: http://dx.doi.org/10.1109/TC.2013.127

[52] I. Ukhov, P. Eles, and Z. Peng. 2015. Temperature-Centric Reliability Analysis and Optimization of Electronic Systems Under Process Variation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 11 (Nov 2015), 2417–2430. DOI: http://dx.doi.org/10.1109/TVLSI.2014.2371249

[53] Zhe Wang and S. Ranka. 2010. A simple thermal model for multi-core processors and its application to slack allocation. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. 1–11. DOI: http://dx.doi.org/10.1109/IPDPS.2010.5470426

[54] Zhe Wang and S. Ranka. Aug. Thermal constrained workload distribution for maximizing throughput on multi-core processors. In *Green Computing Conference, 2010 International*. 291–298. DOI: http://dx.doi.org/10.1109/GREENCOMP.2010.5598302

[55] Q. Xie, J. Kim, Y. Wang, D. Shin, N. Chang, and M. Pedram. 2013. Dynamic thermal management in mobile devices considering the thermal coupling between battery and application processor. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 242–247. DOI: http://dx.doi.org/10.1109/ICCAD.2013.6691125

[56] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma. 2016. 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 (Aug 2016), 1269–1282. DOI: http://dx.doi.org/10.1109/TCAD.2015.2501286

[57] Junlong Zhou, Jianming Yan, Jing Chen, and Tongquan Wei. 2015. Peak Temperature Minimization via Task Allocation and Splitting for Heterogeneous MPSoC Real-Time Systems. *Journal of Signal Processing Systems* (2015), 1–11. DOI: http://dx.doi.org/10.1007/s11265-015-0994-4

[58] C. Zhu, Z. Gu, L. Shang, R. P. Dick, and R. Joseph. 2008. Three-Dimensional Chip-Multiprocessor Run-Time Thermal Management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 8 (Aug 2008), 1479–1492. DOI: http://dx.doi.org/10.1109/TCAD.2008.925793

APPENDIX: On Fundamental Principles for Thermal-Aware Design on Periodic Real-Time Multi-Core Systems

Lemma 4.1 *Matrix* $\mathbf{A}$ *has all negative real eigenvalues.*

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.

$\mathbf{G}$ is positive definite, because a symmetric diagonally dominant matrix with real non-negative diagonal entries is positive definite [30]. 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 [3]).

Since there exists a nonsingular matrix $\mathbf{C}^{1/2}$ such that the *similarity transformation* (page 506 in [32]) 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 [32]). Thus, all the eigenvalues of $\mathbf{A}$ are negative real numbers. □

Lemma 4.2 *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 [3]). 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 $\mathbf{Q}\mathbf{C}^{1/2}$ such that $(\mathbf{Q}\mathbf{C}^{1/2})\mathbf{A}(\mathbf{Q}\mathbf{C}^{1/2})^{-1} = \mathbf{\Gamma}$ is a diagonal matrix, so $\mathbf{A}$ is diagonalizable (Page 303 Definition 2 in [3]). □

Lemma 4.3 *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

$$\widetilde{T}_i^\infty - T_i^\infty = \sum_{j=1}^N \mathscr{A}_{i,j}(\widetilde{B}_j - B_j) = \sum_{\substack{j=1 \\ j\neq\mu}}^N \mathscr{A}_{i,j}(\widetilde{B}_j - B_j) + \mathscr{A}_{i,\mu}(\widetilde{B}_\mu - B_\mu). \tag{11}$$

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 $\mathscr{A}_{i,j} > 0$. □

Lemma 4.4 *Given a multi-core platform running two constant modes long enough to enter the stable state, if the supply voltages of the two running modes satisfy* $\mathbf{v}_1 \geq \mathbf{v}_2$, *we have* $\mathbf{T}^\infty(\mathbf{v}_1) \geq \mathbf{T}^\infty(\mathbf{v}_2)$.

Proof. From (1), 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 4.3), we have $\mathbf{T}^{\infty}(\mathbf{v}_1) \geq \mathbf{T}^{\infty}(\mathbf{v}_2)$. □

Lemma 4.5 *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 [32]). 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 [32]) 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 [18]), 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 (5), we have

$$\begin{aligned}
&(\mathbf{I} - e^{\mathbf{A}l})^{-1} = (\mathbf{I} - \mathbf{W} \cdot e^{\mathbf{D}l} \cdot \mathbf{W}^{-1})^{-1} \\
=&(\mathbf{W} \cdot \mathbf{W}^{-1} - \mathbf{W} \cdot e^{\mathbf{D}l} \cdot \mathbf{W}^{-1})^{-1} \\
=&(\mathbf{W} \cdot (\mathbf{I} - e^{\mathbf{D}l}) \cdot \mathbf{W}^{-1})^{-1} = \mathbf{W} \cdot e^{(\mathbf{I}-\mathbf{D}l)^{-1}} \cdot \mathbf{W}^{-1} \\
=&\mathbf{W} \cdot diag\{(1 - e^{-\lambda_1 l})^{-1}, \cdots, (1 - e^{-\lambda_N l})^{-1}\} \cdot \mathbf{W}^{-1}.
\end{aligned} \tag{12}$$

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. □

Theorem 4.6 *Let* $l > 0$ *and* $\mathbf{0} \leq \mathbf{T} \leq (\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v_{min}}))$*, then* $(\mathbf{I} - \mathbf{K})\mathbf{T} \geq \mathbf{0}$*, 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 $\mathbf{v} \leq \mathbf{v_{max}}$, we have $\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v}) \geq \mathbf{0}$ (Lemma 4.4). 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 [18]), we have $\forall t \in [t_{q-1}, t_q]$, $\mathbf{T}(t) \leq \mathbf{T}^{\infty}(\mathbf{v_{max}})$. Thus, from (3), $\mathbf{T}(t)$ can be expressed as

$$\begin{aligned}
e^{\mathbf{A}(t-t_{q-1})}\mathbf{T}^{\infty}(\mathbf{v_{max}}) + (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})\mathbf{T}^{\infty}(\mathbf{v}) \leq \mathbf{T}^{\infty}(\mathbf{v_{max}}) \\
\Rightarrow (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})(\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v})) \geq \mathbf{0}
\end{aligned} \tag{13}$$

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 [18]). Thus, we have

$$\begin{aligned}
e^{\mathbf{A}(t-t_{q-1})}\mathbf{T}^{\infty}(\mathbf{v_{min}}) + (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})\mathbf{T}^{\infty}(\mathbf{v}) \geq \mathbf{T}^{\infty}(\mathbf{v_{min}}) \\
\Rightarrow (\mathbf{I} - e^{\mathbf{A}(t-t_{q-1})})(\mathbf{T}^{\infty}(\mathbf{v}) - \mathbf{T}^{\infty}(\mathbf{v_{min}})) \geq \mathbf{0}
\end{aligned} \tag{14}$$

Since $\mathbf{v_{min}} \le \mathbf{v} \le \mathbf{v_{max}}$, we have $\mathbf{T}^{\infty}(\mathbf{v_{min}}) \le \mathbf{T}^{\infty}(\mathbf{v}) \le \mathbf{T}^{\infty}(\mathbf{v_{max}})$, which both satisfy (13) and (14) contemporarily. Thus, $(\mathbf{I}-\mathbf{K})\mathbf{T} \ge \mathbf{0}$ holds throughout our problem and $\mathbf{0} \le \mathbf{T} \le (\mathbf{T}^{\infty}(\mathbf{v_{max}}) - \mathbf{T}^{\infty}(\mathbf{v_{min}}))$. □

Theorem 5.1 *Let $\mathbb{S}(t) = \{\mathbb{I}_q : q = 1 \cdots z\}$ be a periodic multi-core schedule. Then, the stable-state temperature $\mathbf{T}_{ss}(t)$ achieved by running $\mathbb{S}(t)$ 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 (4), we have

$$\begin{aligned}\mathbf{T}_{ss}(t_0) &= \mathbf{T_0} + (\mathbf{I}-\mathbf{K})^{-1}(\mathbf{T}(t_p) - \mathbf{T_0}) \\ &= \mathbf{T_0} + (\mathbf{I}-\mathbf{K})^{-1}[\mathbf{K}\cdot\mathbf{T_0} + \mathbf{F} - \mathbf{T_0}] \\ &= (\mathbf{I}-\mathbf{K})^{-1}\cdot\mathbf{F}\end{aligned} \tag{15}$$

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 5.1 shows that when identifying the peak temperature in the stable status, we can assume $\mathbf{T_0} = \mathbf{0}$ to ease the presentation in this paper, if otherwise specified. 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 $\mathbb{I}_q = [t_{q-1}, t_q]$, (3) can be simplified as

$$\begin{aligned}\mathbf{T}(t_1) &= e^{\mathbf{A}l_1}\mathbf{T_0} + (\mathbf{I} - e^{\mathbf{A}l_1})\mathbf{T}_1^{\infty} = (\mathbf{I} - e^{\mathbf{A}l_1})\mathbf{T}_1^{\infty}; \\ \mathbf{T}(t_2) &= \sum_{q=1}^{2} e^{\mathbf{A}\sum_{\theta=q+1}^{2} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty}; \\ &\cdots \\ \mathbf{T}(t_h) &= \sum_{q=1}^{h} e^{\mathbf{A}\sum_{\theta=q+1}^{h} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty}; \\ &\cdots \\ \mathbf{T}(t_p) &= \sum_{q=1}^{z} e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I} - e^{\mathbf{A}l_q})\mathbf{T}_q^{\infty}.\end{aligned} \tag{16}$$

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

Theorem 5.3 *The peak temperature when repeating a step-up execution trace $\mathbb{S}_u(t)$ periodically from the ambient temperature occurs at the end of the period in the stable status.*

Proof. Assume step-up worst-case execution trace $\mathbb{S}_u(t)$ is of period t_p and contains z state intervals with scheduling points $t_0, t_1, \cdots, t_p$. 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 $\Delta t_x = t_x - t_{h-1}$. The following proof is to show that for $\mathbb{S}_u(t)$ the temperature at t_p is no lower than the temperature of t_x. Based on (4), we have

$$\begin{cases}\mathbf{T}_{ss}(t_p) &= \mathbf{T}(t_p) + \mathbf{K}(\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(t_p) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(t_p) \\ \mathbf{T}_{ss}(t_x) &= \mathbf{T}(t_x) + \mathbf{K}_x(\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(t_p) \\ &= (\mathbf{I}-\mathbf{K})^{-1}[(\mathbf{I}-\mathbf{K})\mathbf{T}(t_x) + \mathbf{K}_x\mathbf{T}(t_p)],\end{cases} \tag{17}$$

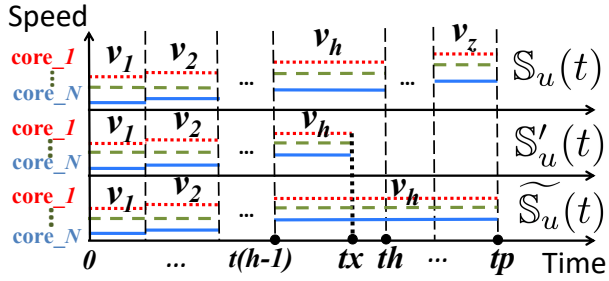


Fig. 11. Proof illustration for Theorem 5.3

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 4.5), 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

$$\mathbf{T}(t_p) - [(\mathbf{I}-\mathbf{K})\mathbf{T}(t_x) + \mathbf{K}_x\mathbf{T}(t_p)] = (\mathbf{I}-\mathbf{K}_x)(\mathbf{I}-\mathbf{K})[(\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(t_p) - (\mathbf{I}-\mathbf{K}_x)^{-1}\mathbf{T}(t_x)] \geq \mathbf{0}. \quad (18)$$

Since $(\mathbf{I}-\mathbf{K}_x)(\mathbf{I}-\mathbf{K})\mathbf{T} \geq \mathbf{0}$, if $\mathbf{T} \geq \mathbf{0}$ (Theorem 4.6), 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 (4), $(\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(t_p)$ is the stable status temperature at time t_p of schedule $\mathbb{S}_u(t)$, and $(\mathbf{I}-\mathbf{K}_x)^{-1}\mathbf{T}(t_x)$ is the stable status temperature at time t_x for the $\mathbb{S}'_u(t)$ that consists of all state intervals of $\mathbb{S}_u(t)$ within interval $[0, t_x]$.

To prove $\mathbb{S}_u(t)$ has a higher peak temperature than $\mathbb{S}'_u(t)$, we define an intermediate schedule $\widetilde{\mathbb{S}_u}(t)$ with period t_p, which consists of all the state intervals of $\mathbb{S}'_u(t)$ within interval $[0, t_{h-1}]$, and keeps constant voltage $\mathbf{v}_h$ within $[t_{h-1}, t_p]$, as shown in Figure 11. Then we need to prove $\mathbf{T}_{ss}(\mathbb{S}_u(t_p)) \geq \mathbf{T}_{ss}(\widetilde{\mathbb{S}_u}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'_u(t_x))$.

First, we prove $\mathbf{T}_{ss}(\mathbb{S}_u(t_p)) \geq \mathbf{T}_{ss}(\widetilde{\mathbb{S}_u}(t_p))$, in which $\mathbf{T}_{ss}(\mathbb{S}_u(t_p)) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(\mathbb{S}_u(t_p))$ and $\mathbf{T}_{ss}(\widetilde{\mathbb{S}_u}(t_p)) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(\widetilde{\mathbb{S}_u}(t_p))$ from (4). $\mathbf{K}$ is identical for both schedule $\mathbb{S}_u(t)$ and $\widetilde{\mathbb{S}_u}(t)$ because their periods are the same. Then, we need to prove $\mathbf{T}(\mathbb{S}_u(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}_u}(t_p))$, 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 $\mathbb{S}_u(t)$ and $\widetilde{\mathbb{S}_u}(t)$, 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 $\mathbb{S}_u(t)$ is in a step-up pattern (Definition 5.2). Then, from (16), we have

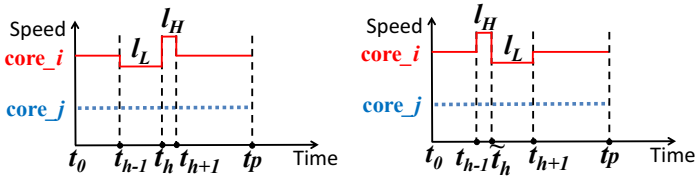
$$\begin{aligned}
&\mathbf{T}(\mathbb{S}_u(t_p)) - \mathbf{T}(\widetilde{\mathbb{S}_u}(t_p)) \\
=&\sum_{q=1}^{h} \left(e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I}-e^{\mathbf{A}l_q})(\mathbf{T}_q^\infty - \widetilde{\mathbf{T}}_q^\infty)\right) + \sum_{q=h+1}^{z} \left(e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I}-e^{\mathbf{A}l_q})(\mathbf{T}_q^\infty - \widetilde{\mathbf{T}}_q^\infty)\right) \\
=&\mathbf{0} + \sum_{q=h+1}^{z} \left(e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}(\mathbf{I}-e^{\mathbf{A}l_q})(\mathbf{T}_q^\infty - \widetilde{\mathbf{T}}_q^\infty)\right).
\end{aligned} \quad (19)$$

Since $e^{\mathbf{A}\sum_{\theta=q+1}^{z} l_\theta}$ contains all positive elements (Lemma 1 in [18]), 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 4.6).

Next, we prove $\mathbf{T}_{ss}(\widetilde{\mathbb{S}_u}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'_u(t_x))$. Starting from the same temperature, if we can prove that for each consecutive period the ending temperature of $\widetilde{\mathbb{S}_u}(t)$ is greater than that of $\mathbb{S}'_u(t)$,

we are able to claim that in the stable status $\mathbf{T}_{ss}(\widetilde{\mathbb{S}_u}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'_u(t_x))$. To prove $\mathbf{T}(\widetilde{\mathbb{S}_u}(t_p)) \geq \mathbf{T}(\mathbb{S}'_u(t_x))$ for the first execution period, we know in $[0, t_x]$, $\widetilde{\mathbb{S}_u}(t)$ and $\mathbb{S}'_u(t)$ are the same, so $\mathbf{T}(\widetilde{\mathbb{S}_u}(t_x)) = \mathbf{T}(\mathbb{S}'_u(t_x))$. Then, within $[t_x, t_p]$ the temperature of schedule $\widetilde{\mathbb{S}_u}(t)$ is monotonically non-decrease because it is a step-up pattern, so we have $\mathbf{T}(\widetilde{\mathbb{S}_u}(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}_u}(t_x))$. Then, for the second period, the starting temperature of $\widetilde{\mathbb{S}_u}(t)$ is no lower than $\mathbb{S}'_u(t)$, so we have $\mathbf{T}(\widetilde{\mathbb{S}_u}(t_x)) \geq \mathbf{T}(\mathbb{S}'_u(t_x))$. Therefore, for the second period, $\mathbf{T}(\widetilde{\mathbb{S}_u}(t)) \geq \mathbf{T}(\mathbb{S}_u(t))$. So on and so forth, we prove $\mathbf{T}_{ss}(\widetilde{\mathbb{S}_u}(t_p)) \geq \mathbf{T}_{ss}(\mathbb{S}'_u(t_x))$. □

Lemma 8.1. *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) $\widetilde{\mathbb{S}}(t)$

Fig. 12. Proof illustration for Lemma 8.1

Proof. To prove $\mathbf{T}_{ss}(\mathbb{S}(t_p)) = (\mathbf{I}-\mathbf{K})^{-1}\mathbf{T}(\mathbb{S}(t_p))$ is no smaller 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), we have

$$\mathbf{T}(\mathbb{S}(t_h)) = e^{\mathbf{A}\cdot l_L}\mathbf{T}(\mathbb{S}(t_{h-1})) + (\mathbf{I} - e^{\mathbf{A}\cdot l_L})\mathbf{T}_L^{\infty} \tag{20a}$$

$$\mathbf{T}(\mathbb{S}(t_{h+1})) = e^{\mathbf{A}\cdot l_H}\mathbf{T}(\mathbb{S}(t_h)) + (\mathbf{I} - e^{\mathbf{A}\cdot l_H})\mathbf{T}_H^{\infty}, \tag{20b}$$

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, take (20a) into (20b), for $\mathbb{S}(t)$ we have

$$\mathbf{T}(\mathbb{S}(t_{h+1})) = e^{\mathbf{A}\cdot l_H}e^{\mathbf{A}\cdot l_L}\mathbf{T}(\mathbb{S}(t_{h-1})) + e^{\mathbf{A}\cdot l_H}(\mathbf{I} - e^{\mathbf{A}l_L})\mathbf{T}_L^{\infty} + (\mathbf{I} - e^{\mathbf{A}\cdot l_H})\mathbf{T}_H^{\infty}. \tag{21}$$

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

$$\mathbf{T}(\widetilde{\mathbb{S}}(t_{h+1})) = e^{\mathbf{A}\cdot l_H}e^{\mathbf{A}\cdot l_L}\mathbf{T}(\widetilde{\mathbb{S}}(t_{h-1})) + e^{\mathbf{A}\cdot l_L}(\mathbf{I} - e^{\mathbf{A}l_H})\mathbf{T}_H^{\infty} + (\mathbf{I} - e^{\mathbf{A}\cdot l_L})\mathbf{T}_L^{\infty}. \tag{22}$$

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

$$\mathbf{T}(\mathbb{S}(t_{h+1})) - \mathbf{T}(\widetilde{\mathbb{S}}(t_{h+1})) = (\mathbf{I} - e^{\mathbf{A}l_H})(\mathbf{I} - e^{\mathbf{A}l_L})(\mathbf{T}_H^{\infty} - \mathbf{T}_L^{\infty}) \geq \mathbf{0} \tag{23}$$

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 4.6). Then, since $\mathbb{S}(t)$ and $\widetilde{\mathbb{S}}(t)$ run at same speeds within $t \in [t_{h+1}, t_p]$, we can conclude $\mathbf{T}(\mathbb{S}(t_p)) \geq \mathbf{T}(\widetilde{\mathbb{S}}(t_p))$. □

Theorem 5.4 *Given a periodic schedule $\mathbb{S}(t)$, the corresponding step-up execution trace $\mathbb{S}_u(t)$ bounds the peak temperature of $\mathbb{S}(t)$.*

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 (1) is a linear time-invariant system [2, 43], 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 8.1, Theorem 5.4 can therefore be proved. □

Theorem 6.2 *Let $\mathbb{S}_u(t) = \{\mathbb{I}_q : q = 1 \cdots z\}$ be a periodic step-up worst-case execution trace of any periodic schedule on a multi-core processor. Then, $T_{peak}(\mathbb{S}_u(m_1, t)) \geq T_{peak}(\mathbb{S}_u(m_2, t))$, if $1 \leq m_1 \leq m_2$.*

Proof. Since $\mathbb{S}_u(t)$ is a step-up worst-case execution trace, both $\mathbb{S}_u(m_1, t)$ and $\mathbb{S}_u(m_2, t)$ are in the step-up pattern with period t_p/m_1 and t_p/m_2, respectively. From Theorem 5.3, their peak temperatures must occur at relative time corresponding to $t = t_p/m_1$ and t_p/m_2, respectively, when temperature reaches the stable status.

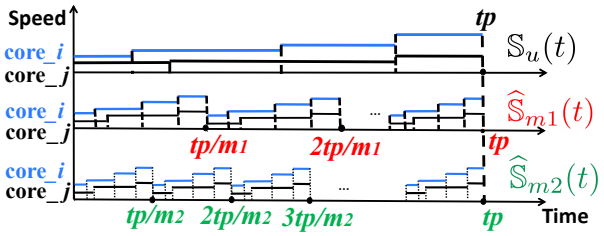


Fig. 13. Proof illustration for Theorem 6.2

Now, let's construct two new schedules $\widehat{\mathbb{S}}_{m1}(t)$ and $\widehat{\mathbb{S}}_{m2}(t)$ such that $\widehat{\mathbb{S}}_{m1}(t)$ consists of all state intervals of $\mathbb{S}_u(m_1, t)$ within $[0, t_p]$, and $\widehat{\mathbb{S}}_{m2}(t)$ consists of all state intervals of $\mathbb{S}_u(m_2, t)$ within $[0, t_p]$, as shown in Figure 13. Note that, the stable status temperature of $\mathbb{S}_u(m, t)$ at time corresponding to $t = t_p/m$ is equal to the temperature at time $t = t_p$ in the stable status.

Let $\mathbf{T_{m1}}(t_p)$ represent the temperature at the end of the first period of $\widehat{\mathbb{S}}_{m1}(t)$. Then, the stable status temperature of $\widehat{\mathbb{S}}_{m1}(t)$ at time corresponding to $t = t_p$ can be written as $(\mathbf{I} - \mathbf{K})^{-1}\mathbf{T_{m1}}(t_p)$ with $\mathbf{K} = e^{\mathbf{A}t_p}$ from equation (4). Therefore, $T_{peak}(\mathbb{S}_u(m_1, t)) = max((\mathbf{I} - \mathbf{K})^{-1}\mathbf{T_{m1}}(t_p))$. Similarly, let $\mathbf{T_{m2}}(t_p)$ be the temperature of $\widehat{\mathbb{S}}_{m2}(t)$ at the end of the first period, so we have $T_{peak}(\mathbb{S}_u(m_2, t)) = max((\mathbf{I} - \mathbf{K})^{-1}\mathbf{T_{m2}}(t_p))$. Since $(\mathbf{I} - \mathbf{K})^{-1} > \mathbf{0}$ (Lemma 4.5), to prove that $T_{peak}(\mathbb{S}_u(m_1, t)) \geq T_{peak}(\mathbb{S}_u(m_2, t))$, it is to prove that $\mathbf{T_{m1}}(t_p) \geq \mathbf{T_{m2}}(t_p)$.

[Step 1] From (16), we have

$$\mathbf{T_m}(t_p) = \sum_{\varrho=0}^{m-1} \sum_{q=\varrho z+1}^{(\varrho+1)z} \left(e^{\mathbf{A}\sum_{\theta=q+1}^{mz} \frac{l_\theta}{m}} (\mathbf{I} - e^{\mathbf{A}\frac{l_q}{m}}) \mathbf{T}_q^\infty\right) \tag{24}$$

in which $m = m_1$ for $\widehat{\mathbb{S}}_{m1}(t)$ or $m = m_2$ for $\widehat{\mathbb{S}}_{m2}(t)$ schedule.

Let $\mathbf{T_{m1}}(t_p)$ and $\mathbf{T_{m2}}(t_p)$ denote the q-th element when $m = m_1$ and $m = m_2$ in (24), respectively. To prove $\mathbf{T_{m1}}(t_p) - \mathbf{T_{m2}}(t_p) \geq \mathbf{0}$, we need to prove fro each q, there exsists $\mathbf{T_{m1}}(t_p) \geq \mathbf{T_{m2}}(t_p)$, where

$$\mathbf{T_m}(t_p) = \sum_{\varrho=0}^{m-1} e^{\mathbf{A}\sum_{\theta=\varrho z+q+1}^{mz}\frac{l_\theta}{m}}(\mathbf{I} - e^{\mathbf{A}\frac{l_{\varrho z+q}}{m}})\mathbf{T}^{\infty}_{\varrho z+q} \tag{25}$$

Since $\mathbb{S}_u(m_1, t)$ and $\mathbb{S}_u(m_2, t)$ are both periodical, for all positive integer ϱ, we have $l_{\varrho z+q} = l_q$ and $\mathbf{T}^{\infty}_{\varrho z+q} = \mathbf{T}^{\infty}_q$. Then, (25) indeed is the sum of a geometric matrices series [32], which are constituted by the first term as $e^{\mathbf{A}\sum_{\theta=(m-1)z+q+1}^{mz}\frac{l_\theta}{m}}(\mathbf{I}-e^{\mathbf{A}\frac{l_q}{m}})\mathbf{T}^{\infty}_q$, equal ratio as $e^{\mathbf{A}t_p/m}$, and totally contains m terms. Using the sum of geometric matrices series [32], we have

$$\begin{aligned}\mathbf{T_m}(t_p) =& e^{\mathbf{A}\sum_{\theta=(m-1)z+q+1}^{mz}\frac{l_\theta}{m}}(\mathbf{I} - e^{\mathbf{A}\frac{mt_p}{m}})(\mathbf{I} - e^{\mathbf{A}\frac{t_p}{m}})^{-1}(\mathbf{I} - e^{\mathbf{A}\frac{l_q}{m}})\mathbf{T}^{\infty}_q\\ =&e^{\mathbf{A}\sum_{\theta=q+1}^{z}\frac{l_\theta}{m}}(\mathbf{I} - e^{\mathbf{A}t_p})(\mathbf{I} - e^{\mathbf{A}\frac{t_p}{m}})^{-1}(\mathbf{I} - e^{\mathbf{A}\cdot\frac{l_q}{m}})\mathbf{T}^{\infty}_q\\ =&\widetilde{\mathbf{\Theta}}(m)(\mathbf{I} - e^{\mathbf{A}t_p})\mathbf{T}^{\infty}_q\end{aligned} \tag{26}$$

where

$$\widetilde{\mathbf{\Theta}}(m) = e^{\mathbf{A}\frac{\varpi}{m}}(\mathbf{I} - e^{\mathbf{A}\frac{t_p}{m}})^{-1}(\mathbf{I} - e^{\mathbf{A}\frac{l_q}{m}}). \tag{27}$$

and $\varpi = \sum_{\theta=q+1}^{z} l_\theta$.

[Step 2] To compare $\mathbf{T_{m1}}(t_p)$ and $\mathbf{T_{m2}}(t_p)$ from (26) by different m, essentially, it can be sufficiently proved by showing that each eigenvalue of (27) monotonically decreases with m. From (5), we can diagonalize $\widetilde{\mathbf{\Theta}}(m)$ as

$$\widetilde{\mathbf{\Theta}}(m) = \mathbf{W}diag\{\Upsilon_1(m), \cdots, \Upsilon_N(m)\}\mathbf{W}^{-1}. \tag{28}$$

where

$$\Upsilon_i(m) = \frac{e^{-\lambda_i\frac{\varpi}{m}}(1 - e^{-\lambda_i\frac{l_q}{m}})}{1 - e^{-\lambda_i\frac{t_p}{m}}}. \tag{29}$$

We then prove $\Upsilon_i(m)$ monotonically decreases with m as follows. 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

$$F'(\varphi) = \frac{(\sigma_2 - \sigma_1)e^{-(\sigma_1+\sigma_2)\varphi} + \sigma_1 e^{-\sigma_1\varphi} - \sigma_2 e^{-\sigma_2\varphi}}{(1 - e^{-\sigma_2\varphi})^2} > 0. \tag{30}$$

Let

$$\frac{F'(\varphi)}{F(\varphi)} = \frac{\sigma_1 e^{-\sigma_1\varphi}}{1 - e^{-\sigma_1\varphi}} - \frac{\sigma_2 e^{-\sigma_2\varphi}}{1 - e^{-\sigma_2\varphi}} = \xi(\sigma_1) - \xi(\sigma_2) \tag{31}$$

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 $\xi'(\sigma) < 0$.

□

Theorem 6.3 *Let $\mathbb{S}_u(t)$ be a periodic step-up execution trace on a multi-core platform under a peak temperature constraint T_{max}. Using the m-Oscillating approach, $\mathbb{S}_u(m_2, t)$ is able to output a service bound that is no lower than $\mathbb{S}_u(m_1, t)$, if $m_2 \geq m_1 \geq 1$.*

Proof. As shown in Figure 5, without losing generality, $\mathbb{S}_u(t)$ is assumed to be a two-speed schedule. Since $m_2 \geq m_1 \geq 1$, we have $T_{peak}(\mathbb{S}_u(m_2, t)) \leq T_{peak}(\mathbb{S}_u(m_1, t)) \leq T_{peak}(\mathbb{S}_u(t)) \leq T_{max}$ (Theorem 6.2).

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 $\mathbb{S}_u(t)$, respectively. Then, the bounded-delay function of the i-th node of $\mathbb{S}_u(m_1, t)$ is $bdf(\Delta, \rho(A, B), l(0, A))$ [26], which is defined by the slope $\rho(A, B)$ and the bounded-delay $l(0, A)$ (the distance between 0 and point A) for any interval length Δ. Similarly, $bdf(\Delta, \rho(A', B'), l(0, A'))$ is the bounded-delay function of the i-th node for $\mathbb{S}_u(m_2, t)$.

To prove that $\mathbb{S}_u(m_2, t)$ is able to output a service bound that is no lower than $\mathbb{S}_u(m_1, t)$, we need to prove $l(0, A') \leq l(0, A)$ and $\rho(A', B') \not< \rho(A, B)$. 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 $\mathbb{S}_u(m_1, t)$, we can express the slope factor $\rho(A, B)$ and bounded delay $l(0, A)$ as

$$\rho(A, B) = \frac{\rho_L t_{i,L}/m_1 + \rho_H t_{i,H}/m_1}{(t_{i,L} + t_{i,H})/m_1} = \frac{\rho_H t_{i,H} + \rho_L t_{i,L}}{t_{i,H} + t_{i,L}} \tag{32}$$

$$l(0, A) = \frac{(\rho_H - \rho_L) \cdot t_{i,H}/m_1 \cdot t_{i,L}/m_1}{(\rho_H t_{i,H} + \rho_L t_{i,L})/m_1} = \frac{(\rho_H - \rho_L) \cdot t_{i,H} \cdot t_{i,L}}{(\rho_H t_{i,H} + \rho_L t_{i,L}) \cdot m_1} \tag{33}$$

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

$$\rho(A', B') = \frac{\rho_H t_{i,H} + \rho_L t_{i,L}}{t_{i,H} + t_{i,L}} \tag{34}$$

$$l(0, A') = \frac{(\rho_H - \rho_L) \cdot t_{i,H} \cdot t_{i,L}}{(\rho_H t_{i,H} + \rho_L t_{i,L}) \cdot m_2} \tag{35}$$

It is not hard to see that $\rho(A, B) = \rho(A', B')$ and since $m_2 \geq m_1 \geq 1$, we have $l(0, A') \leq l(0, A)$. □