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

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

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 article, we analytically prove a series of fundamental properties and principles concerning the 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, as well as 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: 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. 2020. On Fundamental Principles for Thermal-Aware Design on Periodic Real-Time Multi-Core Systems. *ACM Trans. Des. Autom. Electron. Syst.* 25, 2, Article 23 (February 2020), 23 pages.
https://doi.org/10.1145/3378063

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

This work was supported in part by the Provost Research and Scholarship Grant, Wilkes University, Wilkes-Barre, PA.
Authors' addresses: S. Sha, Wilkes University, 84 West South Street, Wilkes-Barre, PA 18766; email: shi.sha@wilkes.edu; A. S. Bankar and G. Quan, Florida International University, 10555 West Flagler Street, EC3900, Miami, FL 33174; emails: {abank013, gaquan}@fiu.edu; X. Yang, University of Houston Clear Lake, 2700 Bay Area Boulevard, Houston, TX 77058; email: yangxia@uhcl.edu; W. Wen, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015; email: wuw219@lehigh.edu.

1084-4309/2020/02-ART23 $15.00
https://doi.org/10.1145/3378063

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. The soaring power and accompanied thermal crisis negatively impacts the system computational performance and 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.

Although 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 a 2D or 3D multi-core platform come not only from 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 floorplanning, 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], and allocating hot tasks to cores closer to the heat sink [29]. 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], and mathematical programming [23, 54]. Although some of these approaches (e.g., the runtime learning approach in Ge and Qiu [15]) can guarantee the temperature constraints, it could be extremely challenging when employing the preceding 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 article includes the following:

- 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 work, 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 and 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]). 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, 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 Wang and Ranka [54] and by a convex optimization method in Murali et al. [35]. However, Murali et al. [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 Chantem et al. [8], or reduce the peak temperature and energy contemporarily for video streaming in Singh et al. [49]. A genetic programming approach has been proposed to minimize the energy consumption for periodic tasks under a peak temperature constraint in Saha et al.

[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 Fan et al. [13]. These mathematical programming–based approaches can provide a solution but 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 (e.g., priority, preemption, resource sharing) cannot be easily formulated in mathematical programming. For example, a thermal-aware global scheduling algorithm for sporadic task sets has been proposed in Fisher et al. [14], the energy optimization under a given temperature constraint has been solved by a convex method in Hanumaiah and Vrudhula [22] and by a variable-sized bin packing approach in Sha et al. [44]; the throughput maximization problem has been studied in Hanumaiah and Vrudhula [21] and Hanumaiah et al. [23], and the reliability has been enhanced in Hanumaiah and Vrudhula [20]. However, since some works [14, 20–23] incorrectly assume the peak temperature only occurred at a scheduling point and others [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 Pagani et al. [39], which achieves a higher power budget than the traditional TDP. Sha et al. [45] presents 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 article, 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 the thermal-constrained design problem on multi-core platforms. The rest of this article is organized as follows. Section 3 introduces the multi-core system and thermal models. Section 4 studies the general principle of the RC-thermal model. Peak temperature bound and minimization are discussed in Section 5 and Section 6, respectively. Section 7 shows the experiment results and is followed by a conclusion in Section 8.

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 of the matrices/vectors/values are in the real number domain. The notations in Table 1 are used in the article.

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 (e.g., heat sink/spreader), we assume there is no

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 a 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 $\forall j \in [1, N_2]$.	

power consumption. In this article, 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., $[t_{q-1}, t_q]$) a *state interval*.

3.2 Thermal Model

The thermal model, similar to that in other works [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) and inactive nodes having no power/heat dissipation (e.g., heat sink). Please note that active nodes in the RC model are not limited to CPUs alone. Cache units, through-silicon vias (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 the $\mathbf{T}(t)$ vector represents node temperatures at time t. Coefficient matrix $\mathbf{A} = [A_{i,j}]_{N\times N}$ is an architectural-related constant, and thus the system is time invariant. $\mathbf{A}$ depends only on the thermal capacitance matrix $\mathbf{C} = diag\{C_1, \ldots, C_N\}$ 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 Han et al. [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 non-singular (Lemma 1 in Wang and Ranka [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 *non-singular*. 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 not only on 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 work, we take leakage-temperature dependency [18, 44] on the active processing units into account 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 Han et al. [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 a 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 is equal 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 Han et al. [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 an 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 (e.g., $-\mathbf{A}^{-1}$, $e^{\mathbf{A}l}$, or $(\mathbf{I}-e^{\mathbf{A}l})^{-1}$) 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 in the following. In this article, all of the detailed proofs can be found in the appendix, unless 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 of 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 of the eigenvalues of the system matrix are strictly negative real values. An asymptotically stable system is bounded-input, bounded-output (BIBO) stable, which means 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 staying 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 $\mathbf{D}=diag\{-\lambda_1,\ldots,-\lambda_N\}$ and $\mathbf{W}=[\vec{w_1},\ldots,\vec{w_N}]$. $\vec{w_i}$ is the independent eigenvector associated with $-\lambda_i$. The matrix exponential of $e^{\mathbf{A}l}$ can be diagonalized as

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

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

In addition, since none of the eigenvalue of $\mathbf{A}$ is 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 $\mathbf{T}^{\infty}(\mathbf{v_1})\geq\mathbf{T}^{\infty}(\mathbf{v_2})$ 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 a higher initial temperature results in a higher transient temperature.

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 Han et al. [18]. Specifically, for matrix $\mathbf{K}$, we have the following lemma and theorem.

Lemma 4.5. *All of 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 Coskun et al. [11] and 3D thermal-aware energy optimization with memory-awarenesses are proposed in Meng et al. [31]. In these works, the thermal impact of the TSVs is treated as 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], and reliability optimization [33]. Note that the thermal model in this work 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, the Intel Xeon Processor E5 v3 family, ranging from 4 to 18 cores used for embedded and server domains, 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.

A few approaches propose 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.

Some other approaches intend to find the solution analytically. For example, Pagani et al. [38] developed an analytical method to identify the transient peak temperature in a single short time by solving the differential equation array directly. This approach works for small systems and schedules, such as those 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 Pagani et al. [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, Schor et al. [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, for example, 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

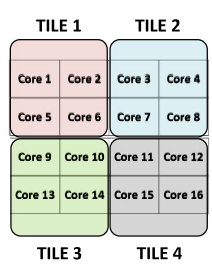
In this section, we utilize a motivation example to show that a hypothetical *step-up execution trace* can tightly bound the runtime temperature for any feasible execution trace based on a given periodic schedule.

Consider a 4×4 multi-core platform with four 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 1.5 seconds and there are five 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 an ambient temperature of 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.

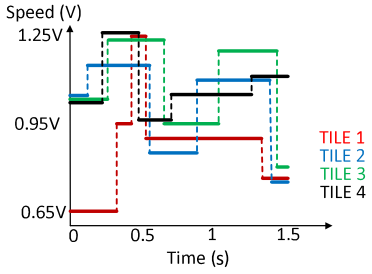
To identify the peak temperature, a common way is to build a temperature trace numerically for the worst-case execution time (WCET) schedule by checking the temperature of each small interval. The problem is that its computational time scales linearly with the period length and numerical checking granularity.

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

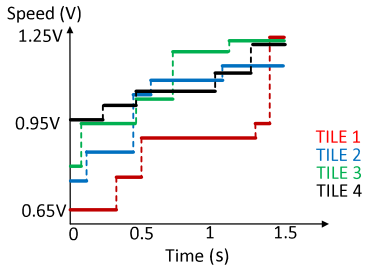
For a quick response time, some works (e.g. [36, 57]) use the highest stable state temperature in each state interval to approximate the interval-wise peak temperature, and the overall peak temperature of running a periodic schedule is selected from the highest interval-wise stable state temperatures, as $T_{peak}(\mathbb{S}(t)) = max(\mathbf{T}_q^{\infty})$, where $q = 1, \ldots, z$. This method simply assumes thermal



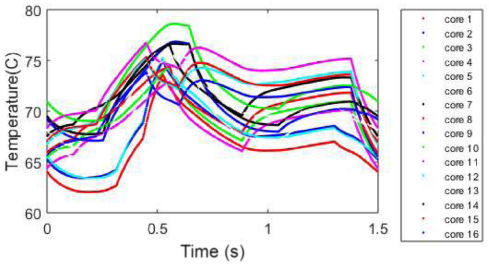
(a) A 16-core platform



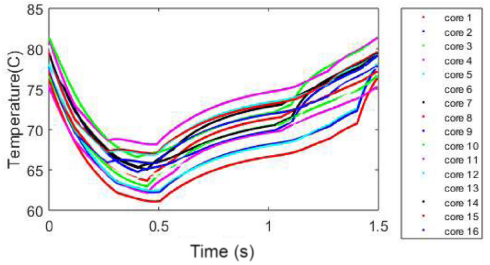
(b) A predefined periodic WCET-schedule.



(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.

Table 2. Motivation Example Speed Schedule

	Consecutive Execution Modes and Lengths on Each Tile				
Tile 1	(0.65 V, 0.315 s)	(0.95 V, 0.12 s)	(1.25 V, 0.09 s)	(0.9 V, 0.78 s)	(0.75 V, 0.195 s)
Tile 2	(1.05 V, 0.12 s)	(1.15 V, 0.435 s)	(0.85 V, 0.33 s)	(1.1 V, 0.495 s)	(0.75 V, 0.12 s)
Tile 3	(1.1 V, 0.255 s)	(1.25 V, 0.39 s)	(0.95 V, 0.405 s)	(1.2 V, 0.375 s)	(0.8 V, 0.075 s)
Tile 4	(1.05 V, 0.225 s)	(1.25 V, 0.225 s)	(0.95 V, 0.15 s)	(1.05 V, 0.78 s)	(1.1 V, 0.12 s)

Note: Each interval is represented by a (supply voltage in volts (V) and execution length in seconds (s)) pair.

nodes can immediately reach the stable state temperature. However, this method either cannot sufficiently bound the runtime peak temperature [38] or is 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 in each state interval of any schedule could not reach its state-state temperature immediately due to thermal capacitance. Therefore, the estimation using the stable state temperature can be very pessimistic. However, 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 be 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 30 ms. 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 than on 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]. Thus, it is difficult to predict where and when the peak temperature could 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 article, unless otherwise specified. In addition, since the peak temperature occurs in thermal steady status, when all of 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 $\mathbf{v}_q \leq \mathbf{v}_{q+1}, \forall q \in \{1, \ldots, z-1\}$.

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 compared 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 $\mathbb{S}(t)$.*

Theorem 5.4 can be interpreted based on the facts that the multi-core thermal model presented in (1) is an LTI system [2, 43], which follows the superposition principle: (1) the thermal impact at one time instant is the sum of the thermal impact by each core, and (2) the thermal impact of each core is the sum of the impact by each state interval in the schedule. Thus, without changing other

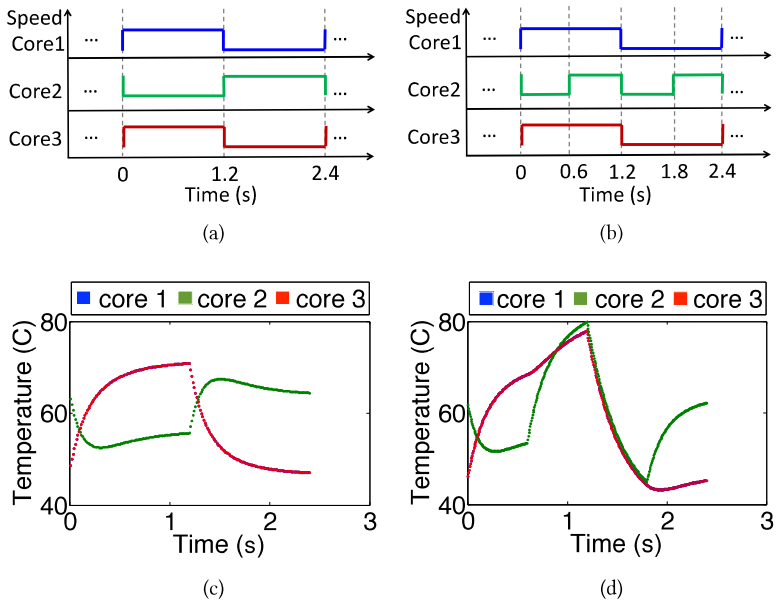


Fig. 2. (a) Core 1 and core 3 run at 1.3 V within [0, 1.2] seconds and 0.6 V within [1.2, 2.4] seconds. Core 2 runs at 0.6 V within [0, 1.2] seconds and 1.3 V within [1.2, 2.4] seconds. (b) Core 2 doubles its oscillating frequency from the schedule in Figure 2(a). (c) The stable status temperature trace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

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 Theorems 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. Today, 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 reasons. 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 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 Huang et al. [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 multi-core 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 Figure 2(a), the period is 2.4 seconds. Each core runs at equal times in two processing modes, with high-voltage $v_H = 1.3\,\text{V}$ and low-voltage $v_L = 0.6\,\text{V}$. Figure 2(c) shows the stable status temperature trace within one period, with a peak temperature of 70.83°C.

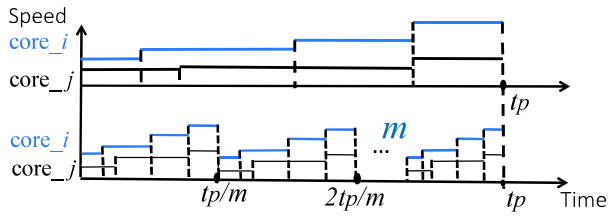


Fig. 3. Illustration of the m-Oscillating schedule [46].

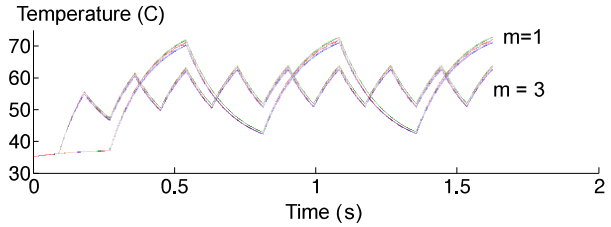


Fig. 4. Peak temperature reduction by m-Oscillating approach.

Next, we let core 2 double its oscillating frequency and core 1 and core 3 keep the same schedule, as shown in Figure 2(b). In the stable status, as shown in Figure 2(d), the peak temperature becomes 79.86°C, which is higher than in Figure 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 Theorems 5.3 and 5.4 by temporally evening 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 the 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 $t_p = 0.6$ s, starting from ambient temperature $T_{amb} = 35$°C. On each core, the high- and low-frequency modes are set to 3 and 1.8 GHz and execute 0.3 seconds for each mode within one period, respectively. In Figure 4, by applying the 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* $1 \leq m_1 \leq m_2$.

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

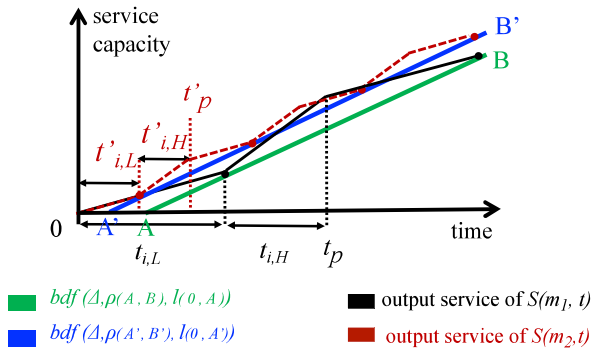


Fig. 5. Real-time calculus illustration of Theorem 6.3. The bounded delay approximation shows a higher service bound of $\mathbb{S}(m_2, t)$ than $\mathbb{S}(m_1, t)$, if $m_2 \geq m_1 \geq 1$.

function of the transition overhead as Algorithm 2 in Sha et al. [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).

Considering 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, the 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).

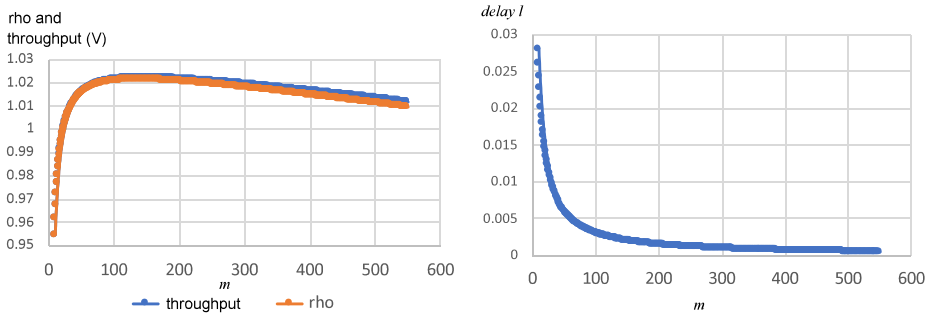
Let $t_{i,H}$ and $t_{i,L}$ denote the high-speed and low-speed length on the i-th node within one period t_p (as $t_{i,H} + t_{i,L} = t_p$) of $\mathbb{S}_u(t)$, respectively. Then, the bounded delay function of the i-th node of $\mathbb{S}_u(m_1, t)$ is $bdf(\Delta, \rho(A, B), l(0, A))$ [26], which 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.



(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.

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 an improved real-time service bound.

Let τ be the transition overhead on the given platform and $bdf(\Delta, \rho(\tilde{A}, \tilde{B}, \tau), l(0, \tilde{A}, \tau))$ 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., $(\rho_H + \rho_L)\tau$). Therefore, 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 Sha et al. [46] (Figure 6(a)). To maximize the system performance, we first find the "optimal m" that 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 and high speeds are supported by 0.6 V and 1.3 V, respectively. The maximally allowed temperature is set to 70°C. The period is 1.5 seconds, and the high and low speeds take 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 Sha et al. [45]). The system throughput is defined as the average speed within one period, which is defined in Equation (5) of Sha et al. [45].

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 rate.

7 EXPERIMENTAL RESULTS

In this section, we focus on simulating a series of peak temperature bounding methodologies on four multi-core configurations—that is, 2×3, 3×3, 3×4, and 4×4 corresponding to 6, 9, 12, and 16 cores, respectively. Each core size is $4 \times 4\,mm^2$ and follows the Alpha 21264 floorplan, and there are 15 available speed levels supported by discrete supply voltages from 0.6 V to 1.3 V with a 0.05 V 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 $T_{amb} = 35°C$, unless otherwise specified. In this article, we assume the WCET schedules are given, so they should be free of any deadline miss case.

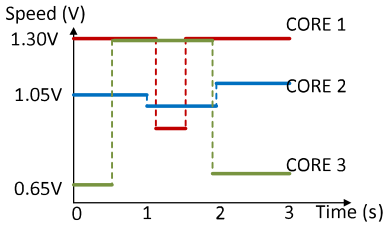
There are five approaches in this experiment. First, the numerical method (NM) splits each execution interval into small pieces and finds the peak temperature by checking each small step length (e.g., [40, 47, 50]). Second, the stable state temperature check method (TssCheck) checks the peak temperature interval by interval using a combination of the analytical and numerical approaches [18]. Third, the maximum transient temperature approach (MatEx) uses the Newton-Raphson method to solve the first-order derivative on each core in (1) as shown in Pagani et al. [38]. Fourth, the 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 Schor et al. [43]. Fifth, the last one is our proposed method—that is, 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.

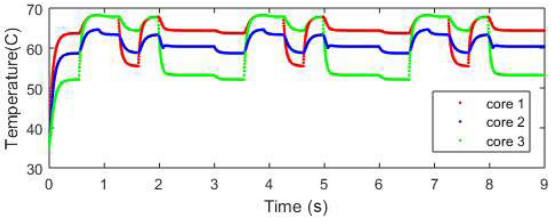
7.1 The Step-up Execution Trace can Bound the Peak Temperature

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

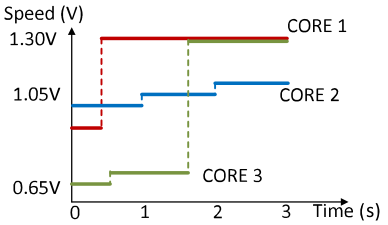
Figure 7(b) shows that the peak temperature for executing the schedule in Figure 7(a) is 68.2907°C. If we reorder the WCET intervals in Figure 7(a) following the step-up execution trace (in Definition 5.2) as shown in Figure 7(c), the peak temperature is calculated as 68.8824°C in Figure 7(d), which 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 of the peak temperatures in the step-up execution trace must occur at the end of the period, which conforms to Theorem 5.3. Meanwhile, we can also observe that the peak temperature of the step-up execution



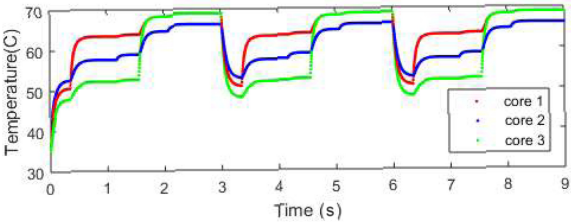
(a) A predefined periodic WCET-schedule.



(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.

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

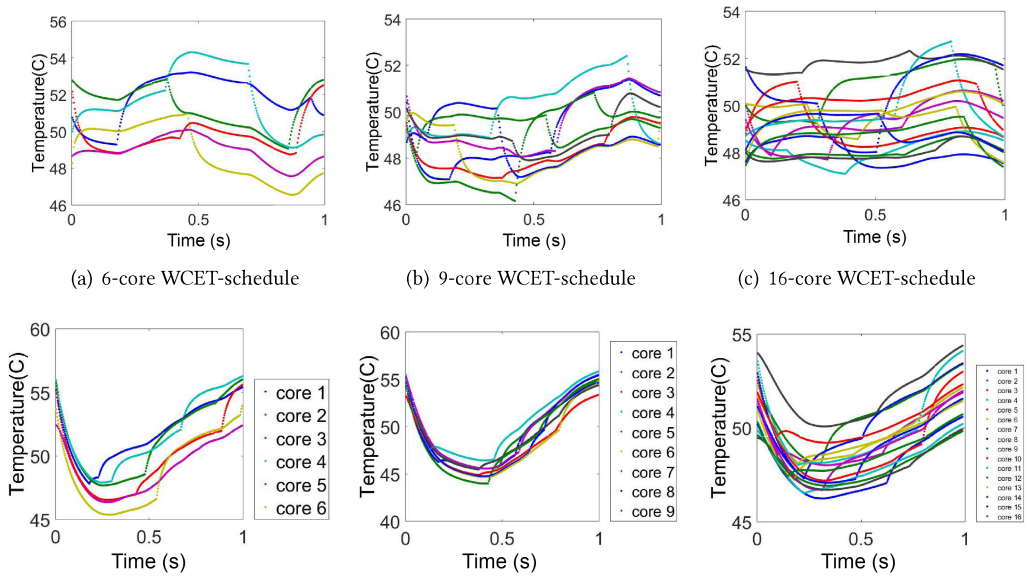
7.2 The Tightness of the Temperature Bound by the StepUp Method

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 are equal to 10 ms, 50 ms, and 100 ms, respectively. As the period becomes 0.5, 1, and 5 seconds, 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, thus causing 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 5 seconds, which is smaller than 3.12°C when the period equals 1 second. 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.

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 0.5 seconds, the error



(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 occurs at the end of its period (Theorem 5.3).

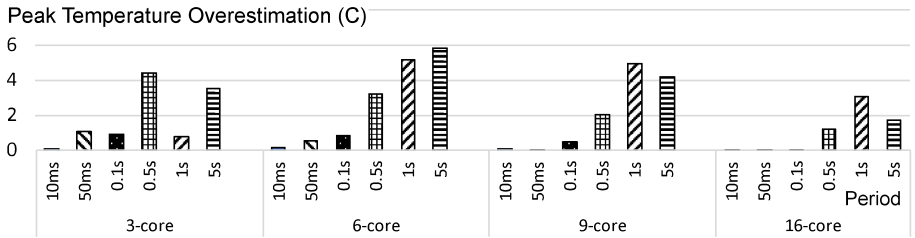


Fig. 9. The overestimation of the temperature bound by the StepUp method.

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 of 5.82°C occurs on a 6-core platform, when the period equals 5 seconds. The smallest average overestimation of 0.017°C occurs on a 16-core platform, when the period equals 100 ms.

7.3 Computational Cost Comparison of Different Approaches

In this section, we first compare the computational cost of NM, TssCheck, and MatEx approaches in Figure 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 Figure 10(b), on a 6-core platform with five 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 $O(\frac{t_p}{\text{step length}})$.

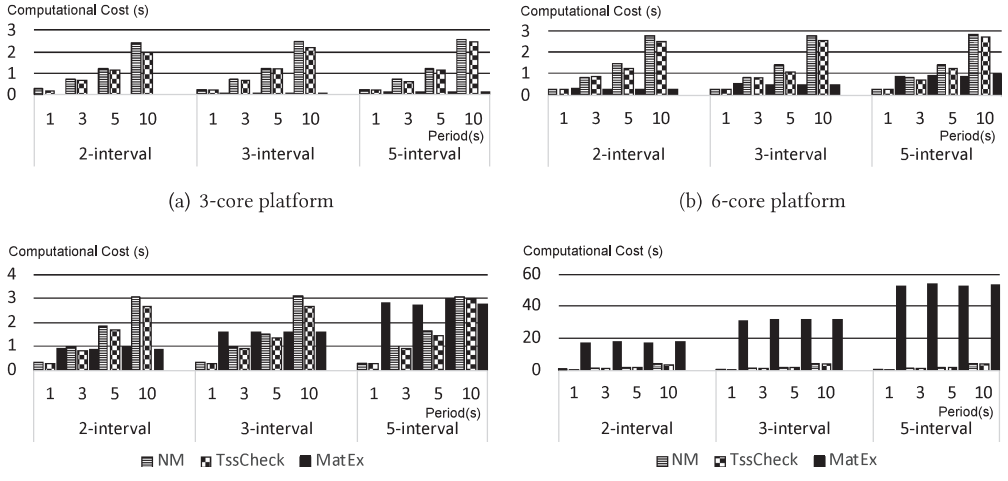


Fig. 10. Computational cost comparison of the NM, TssCheck, and MatEx approaches.

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 (lines 3–8 of Algorithm 1 in Han et al. [18]). For example, in Figure 10(a), on a 3-core platform with $t_p = 10$ and five 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% on 3-, 6-, 9-, and 16-core platforms, respectively. The TssCheck's complexity lower bound and upper bound are O(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 computationally efficient compared to the NM and TssCheck methods, as shown in Figure 10(a) and (b). In Figure 10(b), on a 6-core platform with $t_p = 10$ and three different modes on each core, the MatEx method only uses 0.464 seconds in comparison with 2.772 and 2.558 seconds of the NM and TssCheck methods, respectively. However, in Figure 10(d), on a 16-core platform with $t_p = 10$ and three different modes on each core, the MatEx method spends as much as 31.866 seconds, whereas the 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: WorstTpeak and our proposed StepUp approaches. In this experiment, we let each core have a fixed number of three different modes. The computational cost is shown in Table 3. We can see that StepUp saves orders of magnitudes of 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.

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 is computationally efficient, which can be easily applied to both proactive and reactive temperature-aware design.

Table 3. Computational Cost of the WorstTpeak and StepUp Approaches

	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.

8 CONCLUSION

As IC technology continues to progress, power density keeps increasing and results in soaring chip temperatures. Although extensive thermal-aware research approaches have been developed, we believe the great challenges of thermal problems and the quality-of-service guarantee requirement in design of real-time systems demand a more rigorous analytical study of the thermal problem. In this article, 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 *Proceedings of the 2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50)*. 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 *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS'14)*. 243–252. DOI: https://doi.org/10.1109/ECRTS.2014.15

[3] H. Anton. 2014. *Elementary Linear Algebra, Applications Version 11E with WileyPlus Card*. John Wiley & Sons. 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: https://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 *Proceedings of the 2016 International Conference on High Performance Computing Simulation (HPCS'16)*. 690–696. DOI: https://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 Transactions on Computers* 63, 5 (May 2014), 1097–1110. DOI: https://doi.org/10.1109/TC.2012.293

[7] William L. Brogan. 1985. *Modern Control Theory* (2nd 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: https://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. *Journal of Systems Architecture* 58, 10 (Nov. 2012), 387–397. DOI: https://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 *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design*. 397–402. DOI: https://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 *Proceedings of the 2009 Design, Automation, and Test in Europe Conference and Exhibition (DATE'09)*. 1410–1415. DOI: https://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 *Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC'14)*. 1–6. DOI: https://doi.org/10.1145/2593069.2593199

[13] Songchun Fan, Seyed Majid Zahedi, and Benjamin C. Lee. 2016. The computational sprinting game. *SIGOPS Operating Systems Review* 50, 2 (March 2016), 561–575. DOI: https://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. *Journal of Systems Architecture* 57, 5 (May 2011), 547–560. DOI: https://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 *Proceedings of the 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC'11)*. 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 *Proceedings of the 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 4th Annual IEEE International Workshop on Workload Characterization (WWC-4) (Cat. No. 01EX538)*. 3–14. DOI: https://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. DOI: https://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 Transactions on Embedded Computing Systems* 13, 5s (Oct. 2014), Article 151, 25 pages. DOI: https://doi.org/10.1145/2661430

[20] V. Hanumaiah and S. Vrudhula. 2011. Reliability-aware thermal management for hard real-time applications on multicore processors. In *Proceedings of the 2011 Design, Automation, and Test in Europe Conference and Exhibition (DATE'11)*. 1–6. DOI: https://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 Transactions on Computers* 61, 10 (Oct. 2012), 1484–1494. DOI: https://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 Transactions on Computers* 63, 2 (Feb. 2014), 349–360. DOI: https://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. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 11 (Nov. 2011), 1677–1690. DOI: https://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: https://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 Transactions on Embedded Computer Systems* 13, 2s (Jan. 2014), Article 70, 22 pages. DOI: https://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 48th IEEE Conference on Decision and Control (CDC) held jointly with the 2009 28th Chinese Control Conference*. 6224–6231. DOI: https://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. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 5 (May 2006), 501–513. DOI: https://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 *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*. 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: https://doi.org/10.1109/TVLSI.2014.2360802

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

[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 *Proceedings of the 2012 Design Automation Conference (DAC'12)*. 648–655.

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

[33] J. Minz, E. Wong, and Sung Kyu Lim. 2005. Reliability-aware floorplanning for 3D circuits. In *Proceedings of the 2005 IEEE International SOC Conference*. 81–82. DOI: https://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: https://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 *Proceedings of the 5th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'07)*. 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: https://doi.org/10.1109/TCSI.2008.2011589

[37] S. Pagani, J. Chen, and J. Henkel. 2015. 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: https://doi.org/10.1109/TCAD.2015.2406862

[38] Santiago Pagani, Jian-Jia Chen, Muhammad Shafique, and Jörg Henkel. 2015. MatEx: Efficient transient and peak temperature computation for compact thermal models. In *Proceedings of the 2015 Design, Automation, and Test in Europe Conference and Exhibition (DATE'15)*. 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 Transactions on Computers* 66, 1 (Jan. 2017), 147–162. DOI: https://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: https://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: https://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 *Proceedings of the 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. 41–50. DOI: https://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 *Proceedings of the 2012 IEEE 18th Real-Time and Embedded Technology and Applications Symposium (RTAS'12)*.87–96. DOI: https://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 Computing* 85 (2019), 231–242. DOI: 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 *Proceedings of the 2016 45th International Conference on Parallel Processing (ICPP'16)*. 526–535. DOI: https://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: https://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: https://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: https://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 Transactions on Architecture and Code Optimization* 1, 1 (March 2004), 94–125. DOI: https://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 Transactions on Computers* 63, 10 (Oct. 2014), 2576–2589. DOI: https://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: https://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 *Proceedings of the IEEE International Symposium on Parallel Distributed Processing (IPDPS'10)*. 1–11. DOI: https://doi.org/10.1109/IPDPS.2010.5470426

[54] Zhe Wang and S. Ranka. 2010. Thermal constrained workload distribution for maximizing throughput on multi-core processors. In *Proceedings of the 2010 International Green Computing Conference*. 291–298. DOI: https://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 *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'13)*. 242–247. DOI: https://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: https://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* 84, 1 (July 2015), 111–121. DOI: https://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: https://doi.org/10.1109/TCAD.2008.925793

Received April 2019; revised December 2019; accepted December 2019