2016 45th International Conference on Parallel Processing

Performance Maximization via Frequency Oscillation on Temperature Constrained Multi-core Processors

Shi Sha*, Wujie Wen†, Ming Fan‡, Shaolei Ren§ and Gang Quan¶
*†¶Department of Electrical and Computer Engineering
Florida International University, Miami, Florida 33174, USA
Email: ssha001, wujie.wen, gang.quan@fiu.edu
‡Broadcom Limited, San Jose, California 95131, USA
Email: mingfan@broadcom.com
§University of California Riverside, Riverside, California 92521, USA
Email: sren@ece.ucr.edu

***Abstract*—While multi-core architectures, by exploring the thread/process level parallelism, help to lower down the power/thermal barrier for single core architectures, power/thermal issues are still the primary limiting factors to achieve high computing performance. In this paper, we study the problem of how to maximize the computing performance of multi-core platforms without violating their peak temperature constraint. As different cores may exhibit different thermal behaviors, we propose to run each core with different working frequencies and develop a schedule based on two novel concepts, i.e. the *step-up schedule* and the *m-Oscillating schedule*, for multi-core platforms. We formally prove that the proposed schedule can guarantee the peak temperature constraint for a given multi-core platform. Compared with the traditional exhaustive search-based approach, our approach can reduce the computation time by orders of magnitude and improve the throughput up to 89%, with an average improvement of 11%.**

***Index Terms*—performance maximization, peak temperature minimization, temperature, thermal aware, multi-core**

I. Introduction

Today, the fast growth in computing demands has resulted in rapid increases in both software complexity and underlying hardware integration [1]. As more and more transistors are integrated into one chip, the power consumption has increased exponentially. High power density leads to high temperature, and high temperature can further increase leakage power and, thus, the overall power consumption [2]. As a result, the thermal problem becomes increasingly serious.

High temperature can degrade system performance, reliability, and even damage the chip permanently. For example, it has been reported that every $10-15^{\circ}\mathrm{C}$ temperature increment could result in 50% reduction in the device's lifespan [1] and triple the hardware failure rate [3]. Moreover, the emerging 3D SoC technology has significantly exacerbated the thermal crisis. The 3D IC technology stacks layers of cores vertically on top of each other to take advantage of shorter wires, higher data throughput, and larger memory bandwidth in comparison with 2D design [4]. However, the higher power density and longer heat removal path has made the thermal problem substantially more challenging than its 2D counterpart [5]. Even though multi-core platforms, by exploiting the thread/process level parallelism, help to lower down the power/thermal barrier of single core systems [6], the thermal problem remains as one significant bottleneck towards high-performance systems [7].

Since temperature is closely related to the power consumption, it seems that we can simply employ existing power aware strategies for temperature control, for example, strategies such as those by Isci et al. [8] to maximize computing performance under a given power budget. However, power consumption and temperature, even though they are closely related, exhibit substantially different characteristics. An optimal power minimization technique is not necessarily the best solution for temperature control. As shown by Pagani et al. [9], using traditional thermal design power (TDP) to constrain the peak temperature can result in pessimistic overall throughputs.

To deal with the heat generated by transistors, one intuitive method is to use the traditional cooling methods such as heat sinks, heat spreads, cooling fans, or other advanced cooling mechanisms (e.g. micro channel liquid cooling mechanism on 3D processor). However, designing such a heat dissipation package is uneconomical if not infeasible [10], and it is unsuitable for hand-held devices [11].

An alternative solution to alleviate the thermal stress is to rely on the *dynamic thermal management* (DTM) techniques [12] by reducing the supply voltage/frequency, powering down cores or migrating running threads to keep the chip temperature within a safe range. The problem is how to dynamically adjust the run-time schemes of multi-core platforms appropriately to maximize the computing performance without compromising their peak temperature constraints.

DTM techniques can be largely categorized as reactive (online) and proactive (offline) methods [12]. The reactive method takes actions when run-time temperature approaches or is predicted to exceed a given threshold. The "heat-and-run" schedule [13], "hot-and-cold" job swapping [14], the feedback control scheme [15], and other techniques such as those in [16], [17], [18], [19] belong to this category. These approaches make decisions online and therefore can be flexible and adaptive. However, they heavily depend on the accuracy of temperature prediction and/or run-time temperature monitoring. Due to large uncertainty of program execution, as well as other factors such as inaccuracy of temperature sensor readings, there is no guarantee of avoiding peak temperature violations or maximizing

2332-5690/16 $31.00 © 2016 IEEE
DOI 10.1109/ICPP.2016.67

throughputs. The proactive DTM techniques, on the other hand, may be less flexible or adaptive than the responsive ones, but since they are developed off-line, they can afford higher computational cost for peak temperature constraint guarantee and endure more aggressive design optimization. Our research presented in this paper belongs to this category.

In this paper, we are interested in developing a proactive DTM scheme to optimize the throughput while ensuring the peak temperature constraint. There are a few papers published [20], [21], [22], [23] with research closely related to our work. Murali et al. [20] transformed the throughput maximization problem under peak temperature constraints to a convex optimization problem, then used a 2-phase iterative approach to approximate the solution. However, this approach can be applied only for processors with working frequencies that can be instantaneously and continuously varied, which is not realistic in practice. Moreover, this approach essentially ignores the heat transfer among multiple cores and is therefore inaccurate, especially for those multi-core platforms with 3D architectures. Hanumaiah et al. [21] studied the problem of mapping a given task set to multiple cores and also setting the supply voltages and speeds of these cores to minimize the task completion time. Assuming the peak temperature always occurs at a scheduling point, i.e. the time instant when at least one core changes its running mode, they transformed the problem into a convex optimization problem. However, this assumption is not always true in practice [24]. In addition, while this approach can deal with the discrete levels of supply voltages and working frequencies, the computational cost can be prohibitive for large problem sizes. Kadin et al. [22] used machine learning techniques to set a working frequency for each core to maximize its performance, but their approach could not prevent a processor from overheating. Wang et al. [23] proposed an integer linear programming-based approach (ILP) to maximize the performance of a temperature-constrained multi-core platform. It is well known that the ILP-based approach does not scale well with the problem size.

Our approach presented in this paper is based on the traditional RC-thermal model for multi-core platforms that accounts for heat transfer and leakage/temperature dependency. We also take the discrete levels of supply voltages and working frequencies into consideration. Specifically, we have made the following contributions:

1) To identify the peak temperature for a multi-core schedule can be time consuming in design space exploration, as shown later in this paper, the peak temperature does not necessarily occur at scheduling points for random schedules. To this end, we introduce a special schedule, so called "step-up" schedule, with its peak temperature easily identified to bound the peak temperature for other *arbitrary periodic schedules*;
2) We show that a constant multi-core speed schedule minimizes the peak temperature among all periodic schedules using one or more speeds on each core and complete the same workload. If such a constant speed is not available, the one that uses the two closest neighboring speeds minimize the peak temperature among all periodic step-up schedules, completing the same workload on each core;
3) We extend the concept of the *m-Oscillating* scheme [25] from single core to multi-core platforms and show that the peak temperature of a step-up schedule monotonically decreases as m increases;
4) Based on the above analysis, we present a frequency oscillating method to maximize the throughput with a guarantee of peak temperature on a multi-core platform. Our simulation results show that, compared with the traditional exhaustive search-based approach, the overall performance improvements by our approach can be up to 89% with an average improvement of 11%, and its computation time can be reduced by several orders of magnitude.

The rest of this paper is organized as follows. Section II introduces the performance, power and thermal models, followed by a motivation example in Section III. Section IV presents key principles for our proposed algorithm. Our proposed frequency assignment algorithm is discussed in Section V. Experimental results are shown in Section VI, and Section VII concludes the paper.

II. Preliminaries

We present the models for power, thermal, performance and pertinent assumptions for our multi-core systems. The **bold characters** represent the vectors and matrices and non-bold characters are used for ordinary variables and coefficients.

A. System Model

We consider a multi-core platform $\mathfrak{N}$ with N cores, $\mathfrak{N} = \{core_i : i = 1, \cdots, N\}$. Each core is DVFS-independent. Also, each core has different running modes and each running mode is characterized by a pair of parameters (v, f), where v is the supply voltage and f is the working frequency. For an inactive core, we assume $v = f = 0$. In this paper, for ease of presentation, we use v and f interchangeably 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**.

B. Power/Thermal Model

The total power consumption (P) is composed of dynamic power and leakage power [2]. Dynamic power is proportional to the cubic of supply voltage and leakage power depends linearly on temperature T, i.e. $P_{leak} = \alpha(v) + \beta T(t)$. The total power of the i_{th} core is

$$P_i(t) = \alpha(v_i) + \beta \cdot T_i(t) + \gamma(v_i) \cdot v_i^3, \quad (1)$$

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

Due to power dissipation, the multi-core system increases its temperature. The thermal model, similar to that in [9], [26], [27], [28], [29], [30], is built upon the duality between heat

transfer and electrical phenomena. Specifically, the thermal behavior of a multi-core platform within a state interval can be formulated as

$$\frac{d\mathbf{T}(t)}{dt} = \mathbf{A}\mathbf{T}(t) + \mathbf{B}(\mathbf{v}), \tag{2}$$

where $\mathbf{T}(t)$ represents core temperatures at time t. Coefficient matrix $\mathbf{A}$ is constant and depends only on the thermal capacitances and resistances of the multi-core platform. Coefficient matrix $\mathbf{B}$ depends on not only the thermal capacitances of the multi-core platform but also the running mode of each core as well. Therefore, matrix $\mathbf{B}$ remains constant within each state interval but may change among different state intervals.

When running a multi-core processor under a constant supply voltage profile $\mathbf{v}$ long enough (i.e. $t \to \infty$), it will eventually reach a constant temperature $\mathbf{T}^{\infty} = \mathbf{T}(\infty) = -\mathbf{A}^{-1}\mathbf{B}(\mathbf{v})$ as $\frac{d\mathbf{T}(\infty)}{dt} = \mathbf{0}$. ($\mathbf{A}$ is nonsingular [27] Lemma 1). 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 [26], 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 $t_{q-1} \le t \le t_q$ 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 profile $\mathbf{v}_q$ long enough 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* [2], in which the temperature at the beginning of the period equals to that at the ending point. Specifically, for a periodic schedule $\mathbb{S}(t)$ with z state intervals and period t_p, let t_{q-1} be the starting time of the q_{th} state interval, the transient temperature in the stable status can be formulated as [26]

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

in which $\mathbf{T}(t_q)$ and $\mathbf{T}_{ss}(t_q)$ are the temperature at time t_q in the first period and in the *thermal stable status*, respectively. $\mathbf{T}(0)$ is the starting temperature. 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 $\mathbf{K} = e^{\mathbf{A}\sum_{\theta=1}^{z} l_\theta} = e^{\mathbf{A}t_p}$. $\mathbf{K}$ is a symmetric matrix and all the eigenvalue of $\mathbf{A}$ are negative real numbers.

Furthermore, we assume that a practical multi-core platform exhibits the following thermal property:

Property 1. *Consider a multi-core platform with an initial temperature $\mathbf{T}_0 \ge \mathbf{0}$ (with $\mathbf{T}_0$ normalized to ambient temperature), the temperature of each core is monotonically non-increasing, i.e. $\mathbf{T}(t_2) \le \mathbf{T}(t_1) \le \mathbf{T}_0$, $\forall\ t_2 \ge t_1 \ge 0$, when shutting down all cores at $t_0 = 0$.*

Essentially, we require that, when shutting down the power simultaneously for all the cores, the temperature of each core decreases monotonically.

C. Performance Model

We assume that a multi-core system always runs with periodic schedules and thus the system's throughput is maximized when the performance within each period is maximized. Then, the performance of the multi-core platform, i.e. the chip-wide throughput (THR), is defined as

$$THR = \frac{\sum_{q=1}^{z} THR_q}{N\sum_{q=1}^{z} l_q} = \frac{\sum_{q=1}^{z}\sum_{i=1}^{N} f_{i,q} \cdot l_q}{N\sum_{q=1}^{z} l_q}, \tag{5}$$

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

D. Problem formulation

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

Problem 1. *Given a multi-core platform $\mathfrak{N}$ and its peak temperature threshold T_{max}, set running modes and repeat the execution periodically to maximize the chip-wide throughput with the peak temperature below T_{max} all the time.*

The notations in Table I will be used in the paper.

Table I. Summary of Notations.

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

III. Motivations

Before presenting our approach, we first show a motivation example. There are some existing works [20], [21] on multi-core performance maximization problem that assume each core can run at a continuously variable speed, which is not always possible in practice. In our research, we assume that each core can only run a set of discrete modes, each of which has one dedicated supply voltage/frequency.

When only discrete processing modes are available, one approach is to round down the speeds obtained from existing work (such as [21]) to the lower available ones, namely the lower neighboring speed (**LNS**) method. This approach can be pessimistic, especially for practical processors, when the number of available speeds is very limited. In fact, when each core can only run at one mode (e.g. [20], [21]), the optimal solution can be obtained by exhaustively searching all the speed settings for the one that can maximize the performance without exceeding the temperature threshold. We call this approach exhaustive search (**EXS**), as shown in Algorithm 1.

Algorithm 1 assumes each core runs at one unique discrete mode and thus the temperature eventually reaches the constant value $\mathbf{T}^{\infty}$ in line (7). This algorithm has two limitations. First, the complexity increases exponentially with the number of cores and possible running modes. In our experiments, to exhaustively search an optimal speed profile for a platform of 9 cores with 15 speed modes, the computation time is over 24 hours. Second, each core can only run at one speed. If

Algorithm 1 Exhaustive Search Method (EXS).

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

3: $\boldsymbol{f} = [f_1, \cdots, f_N]$ and $THR_{max} = 0$;
4: **for** $f_1 = f_{lowest}$ to $f_{highest}$ **do**
5: $\cdots$
6: **for** $f_N = f_{lowest}$ to $f_{highest}$ **do**
7: $\mathbf{T}^{\infty} = -\mathbf{A}^{-1} \cdot \mathbf{B}$;
8: **if** $(max(\mathbf{T}^{\infty}) \leq T_{max})\&\&(sum(\boldsymbol{f}) \geq THR_{max})$ **then**
9: $THR_{max} \leftarrow sum(\boldsymbol{f})$;
10: $\boldsymbol{f}_{optimal} = \boldsymbol{f}$;
11: **end if**
12: **end for**
13: $\cdots$
14: **end for**

the maximum temperature is lower than T_{max}, the temperature "headroom" cannot be filled by raising the speed of any core to the next higher level, since it may violate T_{max}. Is it possible to use more than one speed on each core to achieve a better performance with temperatures staying below T_{max}?

Consider a 3-core processor with $T_{max} = 65°\text{C}$, whose power and thermal models are further detailed in Section VI. If we assume an ideal case when the continuously variable speeds are available, we can set voltages of three cores as $[1.2085, 1.1748, 1.2085]V$, respectively, without violating the T_{max}. The chip-wide performance can be as high as 1.1972. Assume there are only two running modes available with low-voltage $v_L = 0.6V$ and high-voltage $v_H = 1.3V$. Since the supply voltage of $1.2085V$ and $1.1748V$ are not available, **LNS** sets the supply voltage for all three cores to be $0.6V$ with the total performance of 0.6. With an exhaustive search, **EXS** sets voltages as $[0.6, 0.6, 1.3]V$ with an overall performance of 0.83, which is better than **LNS**. Instead of using only one speed for each core, we can use two or more speeds. For example, we use the modes with low-voltage $v_L = 0.6V$ and high-voltage $v_H = 1.3V$ to get exactly the same performance as the ideal case by setting their intervals to the ratio listed in Table II, where $ratio(v_H)$ and $ratio(v_L)$ are defined as the high-voltage's and low-voltage's execution time ratio within a given interval t_p ($t_p = 20ms$ in this example). If we run this schedule periodically, the peak temperature becomes $79.69°\text{C}$, which exceeds the temperature threshold. We can reduce the high-voltage ratio and increase the low-voltage ratio, (for example, according to the values listed in Column 1 of Table III), to ensure the peak temperature is no more than $65°\text{C}$, with the average throughput performance of 0.8725, which improves 45.42% over the **LNS** method. It is worth mentioning that, at this time, we ignore the overhead due to the voltage transitions. We discuss this problem later in this paper (section V). It is also interesting to note that in Table III, with different interval lengths (i.e. t_p), the corresponding high/low speed interval ratios and the average throughputs are different.

Table II. The execution time ratio for different cores.

	$\mathbf{core_1}$	$\mathbf{core_2}$	$\mathbf{core_3}$
$\mathbf{ratio(v_H)} =$	0.8693	0.8211	0.8693
$\mathbf{ratio(v_L)} =$	0.1307	0.1789	0.1307

* $ratio(v_H) + ratio(v_L) = 1$.

Table III. The high-speed ratio "$ratio(v_H)$" list when utilizing two speeds under T_{max}.

	original $\mathbf{t_p = 20ms}$*	**2 divisions** $\mathbf{t_p = 10ms}$	**5 divisions** $\mathbf{t_p = 5ms}$
$\mathbf{core_1}$	0.1733	0.2303	0.2713
$\mathbf{core_2}$	0.8211	0.8211	0.8211
$\mathbf{core_3}$	0.1733	0.2303	0.2713
Performance	0.8725	0.8991	0.9182

* t_p denote the period of the multi-core schedule.

The above examples clearly show that using multiple running speeds for each core can potentially improve the throughput performance significantly for a multi-core platform, due to the fact that it is more flexible to adjust the length of a speed interval rather than the speed itself to control the peak temperature. The problem is, however, how to develop the periodic schedule that can maximize the throughput performance without violating the peak temperature constraint. In what follows, we first introduce several observations and principles related to the peak temperature identification and minimization. We then present our algorithm for throughput maximization for a multi-core platform.

IV. Peak Temperature Identification and Minimization

One of the most fundamental problems for throughput maximization under a given peak temperature constraint is to find out when and where the peak temperature occurs. On a single core platform, the peak temperature always occurs at the scheduling point [31], [25]. However, on a multi-core platform, due to the heat interference between cores, it is no longer always the case [28], [29]. To identify the peak temperature on a multi-core platform, HotSpot [10] can be used to estimate temperature, but its computational time is long. Most recently, Pagani et al. [28] introduced a peak temperature identification method with computation time much faster than HotSpot. However, as the design space becomes larger and the number of power changes increase, it is still quite time consuming. Instead of capturing the exact peak temperature, a peak temperature bounding method is also proposed in [29]. However, the computational complexity is still very high and the results can be very pessimistic.

A. The Step-up Schedule

To quickly and accurately predict the peak temperature on multi-core platforms, in this section we introduce a *step-up schedule*, with its peak temperature easily identified. Also, we use step-up schedules to bound the peak temperature for more general periodic schedules.

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

According to Definition 1, the voltage for each core is monotonically non-decreasing from the first to the last state interval in a step-up schedule. For a step-up schedule, its peak

temperature always occurs at the end of the period, as stated in the following theorem. The proof is omitted due to page limit.

Theorem 1. *The peak temperature when repeating a step-up schedule $\mathbb{S}(t)$ periodically from the ambient temperature occurs at the end of the schedule when the temperature reaches the stable status.*

Based on (3) and (4), we can quickly identify the peak temperature with linear complexity. Furthermore, the peak temperature of a step-up schedule can be used to bound the peak temperature of an *arbitrary schedule*. Before we introduce this conclusion, we first introduce the following definition.

Definition 2. *Given an **arbitrary periodic schedule** $\mathbb{S}(t)$, the corresponding step-up schedule (denoted as $\mathbb{S}_u(t)$) is the periodic schedule that, for each core, the schedule consists of the same scheduling intervals as that in $\mathbb{S}(t)$, but these intervals are ordered according to a non-decreasing order of their supply voltages.*

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

Lemma 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 $\mathbf{T}_{ss}(\mathbb{S}(t_p)) \leq \mathbf{T}_{ss}(\widetilde{\mathbb{S}}(t_p))$.*

Lemma 1 indicates that, as a high-speed interval moves toward the end of a periodic schedule, it tends to increase the temperature at the end of the schedule during the stable status. With the help of the lemma, we are now ready to introduce the following theorem.

Theorem 2. *Given an arbitrary periodic schedule $\mathbb{S}(t)$ and its corresponding step-up schedule $\mathbb{S}_u(t)$ with period of t_p, let $T_{peak}(\mathbb{S}(t))$ and $T_{peak}(\mathbb{S}_u(t))$ be the peak temperature during the stable status. Then, $T_{peak}(\mathbb{S}(t)) \leq T_{peak}(\mathbb{S}_u(t))$*

From Theorem 2, the peak temperature of a periodic schedule is no more than that for its corresponding step-up schedule. 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 (2) is a linear time-invariant system [29], [32], following 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. With the assistance of Lemma 1, Theorem 2 can therefore be proved. More detailed proof is omitted due to page limit.

B. Choose Two Neighboring Running Modes

Theorem 2 can bound the peak temperature for an arbitrary periodic multi-core schedule. This helps to ensure that the peak temperature constraint is not violated. The problem now becomes how to design the periodic schedule that can maximize the performance without exceeding the given peak temperature. As our motivation example indicates, using multiple modes rather than lowering down the constant speed for each core helps to improve the throughput. The problem is, if there are more than two different modes available, which mode should be chosen to form the schedule? Some existing works [33], [23] seem to imply using more speed selections can achieve better performance under the temperature constraint. On the other hand, recall that it has been shown that in [31], a constant mode schedule minimizes the dynamic energy consumption among all other schedules accomplishing the same workload, and if such a constant mode is not available, the schedule using the two neighboring modes becomes optimal. Is it possible that similar principles can be applied here? As a matter of fact, similar principles can be established for step-up schedules to minimize the peak temperature, as shown in the following theorem.

Theorem 3. *Let $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$ be two periodic step-up schedules with period t_p, that are exactly the same except for $core_i$. For $\mathbb{S}_{u1}(t)$, $core_i$ uses a constant mode with voltage v_e throughout the period, but for $\mathbb{S}_{u2}(t)$, $core_i$ uses the mode with voltage v_L for l_L seconds followed by v_H for l_H seconds ($l_L + l_H = t_p$) such that*

$$(l_L + l_H) \cdot v_e = l_L \cdot v_L + l_H \cdot v_H. \tag{6}$$

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

proof sketch. Without loss of generality, we assume all cores, except for $core_i$, have no power consumptions. To ease the presentation, we let $t_p = 1$, $x = l_L$ and $1-x = l_H$ ($0 \leq x \leq 1$).

According to Theorem 1, we have $T_{peak}(\mathbb{S}_{u1}(t_p)) = max\big(\mathbf{T}_{ss}(\mathbb{S}_{u1}(t_p))\big)$ and $T_{peak}(\mathbb{S}_{u2}(t_p)) = max\big(\mathbf{T}_{ss}(\mathbb{S}_{u2}(t_p))\big)$. In addition, from (4), we have $\mathbf{T}_{ss}(\mathbb{S}_{u1}(t_p)) = (\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(\mathbb{S}_{u1}(t_p))$ and $\mathbf{T}_{ss}(\mathbb{S}_{u2}(t_p)) = (\mathbf{I} - \mathbf{K})^{-1}\mathbf{T}(\mathbb{S}_{u2}(t_p))$, where $\mathbf{K} = e^{\mathbf{A}t_p}$. Since $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$ are of the same period, their $\mathbf{K}$ are the same, and since $(\mathbf{I} - \mathbf{K})^{-1}$ is a positive matrix [34], we only need to prove in the first period $\mathbf{T}(\mathbb{S}_{u1}(t_p)) \leq \mathbf{T}(\mathbb{S}_{u2}(t_p))$.

From (3) and (4), we have

$$\begin{cases} \mathbf{T}(\mathbb{S}_{u1}(t_p)) = (\mathbf{I} - e^{\mathbf{A}})\mathbf{T}_e^{\infty} \\ \mathbf{T}(\mathbb{S}_{u2}(t_p)) = e^{\mathbf{A}(1-x)}(\mathbf{I} - e^{\mathbf{A}x})\mathbf{T}_L^{\infty} + (\mathbf{I} - e^{\mathbf{A}(1-x)})\mathbf{T}_H^{\infty} \\ \qquad\qquad = (e^{\mathbf{A}(1-x)} - e^{\mathbf{A}})\mathbf{T}_L^{\infty} + (\mathbf{I} - e^{\mathbf{A}(1-x)})\mathbf{T}_H^{\infty}, \end{cases} \tag{7}$$

where $\mathbf{T}_e^{\infty}$, $\mathbf{T}_L^{\infty}$ and $\mathbf{T}_H^{\infty}$ are the constant temperature when $core_i$ runs in the mode with v_e, v_L and v_H long enough while all the other cores keep idle, respectively. To prove $\mathbf{T}(\mathbb{S}_{u1}(t_p)) \leq \mathbf{T}(\mathbb{S}_{u2}(t_p))$, based on (7) and (8), we want to prove

$$\begin{aligned} &\mathbf{T}(\mathbb{S}_{u2}(t_p)) - \mathbf{T}(\mathbb{S}_{u1}(t_p)) \\ =&(\mathbf{I} - e^{\mathbf{A}}) \cdot \big[(\mathbf{I} - e^{\mathbf{A}})^{-1}(e^{\mathbf{A}(1-x)} - e^{\mathbf{A}})\mathbf{T}_L^{\infty} \\ +&(\mathbf{I} - e^{\mathbf{A}})^{-1}(\mathbf{I} - e^{\mathbf{A}(1-x)})\mathbf{T}_H^{\infty} - \mathbf{T}_e^{\infty}\big] \\ =&(\mathbf{I} - e^{\mathbf{A}}) \cdot [\boldsymbol{\rho}\mathbf{T}_L^{\infty} + (\mathbf{I} - \boldsymbol{\rho})\mathbf{T}_H^{\infty} - \mathbf{T}_e^{\infty}] \geq \mathbf{0}, \end{aligned} \tag{8}$$

where $\boldsymbol{\rho} = (\mathbf{I} - e^{\mathbf{A}})^{-1}(e^{\mathbf{A}(1-x)} - e^{\mathbf{A}})$ and $\mathbf{I} - \boldsymbol{\rho} = (\mathbf{I} - e^{\mathbf{A}})^{-1}(\mathbf{I} - e^{\mathbf{A}(1-x)})$. Note that in (8), $(\mathbf{I} - e^{\mathbf{A}})$ is *monotone* because $(\mathbf{I} - e^{\mathbf{A}})^{-1} \geq \mathbf{0}$ [34]. Therefore, we only need to prove that $[\boldsymbol{\rho}\mathbf{T}_L^{\infty} + (\mathbf{I} - \boldsymbol{\rho})\mathbf{T}_H^{\infty} - \mathbf{T}_e^{\infty}] \geq \mathbf{0}$.

From (2), we know $\mathbf{T}^{\infty}(\mathbf{v}) = -\mathbf{A}^{-1}\mathbf{B}(\mathbf{v})$ and $-\mathbf{A}^{-1}$ is a constant matrix which contains all positive elements, because in practical scenarios, without changing any factor, increasing the power (voltage) of one node cannot decrease the temperature of other nodes. Moreover, since $\mathbf{B}(\mathbf{v}) = \mathbf{C}^{-1}\mathbf{\Psi}(\mathbf{v})$, $\mathbf{\Psi}(\mathbf{v}) = [\psi(v_i)]_{N\times 1}$ and for each element $\psi_i(v_i) = \alpha + \gamma v_i^3$ is a convex function (α and γ are constants for a fixed v_i), $\mathbf{T}^{\infty}(\mathbf{v})$ is a convex function [35]. Therefore, given the condition in (6), we have $v_e^3 \le x\cdot v_L^3 + (1-x)\cdot v_H^3$ and $\mathbf{T}_e^{\infty} \le x\cdot\mathbf{T}_L^{\infty} + (1-x)\cdot\mathbf{T}_H^{\infty}$. Next we need to prove

$$x\mathbf{T}_L^{\infty} + (1-x)\mathbf{T}_H^{\infty} \le \boldsymbol{\rho}\mathbf{T}_L^{\infty} + (\mathbf{I}-\boldsymbol{\rho})\mathbf{T}_H^{\infty} \Longrightarrow x\cdot\mathbf{I} \ge \boldsymbol{\rho}. \tag{9}$$

Recall that matrix $\mathbf{A}$ has N negative eigenvalues [28], i.e. $\{-\lambda_i : i = 1,\cdots,N\}$, $\lambda_i \ge 0$. Let matrix $\mathbf{W} = [w_{i,j}]_{N\times N}$ be composed of $\mathbf{A}$'s eigenvectors and let $\mathbf{W}^{-1} = [u_{i,j}]_{N\times N}$. We therefore can diagonalize matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{W}\mathbf{D}\mathbf{W}^{-1}$, where $\mathbf{D} = diag\{-\lambda_i : i = 1,\cdots,N\}$. Because $e^{\mathbf{A}t} = \sum_{n=0}^{\infty}\frac{t^n}{n!}\mathbf{A}^n$, we have $e^{\mathbf{A}t} = \sum_{n=0}^{\infty}\frac{t^n}{n!}(\mathbf{W}\mathbf{D}\mathbf{W}^{-1})^n = \mathbf{W}(\sum_{n=0}^{\infty}\frac{t^n}{n!}\mathbf{D}^n)\mathbf{W}^{-1} = \mathbf{W}e^{\mathbf{D}t}\mathbf{W}^{-1}$. So, $e^{\mathbf{D}t}$ is a diagonal matrix and $e^{\mathbf{D}t} = diag\{e^{-\lambda_i t} : i = 1,\cdots,N\}$. Then, (9) leads to

$$\begin{aligned}
&x\cdot\mathbf{I} \ge \mathbf{W}diag\{\frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1-e^{-\lambda_i}}\}\mathbf{W}^{-1}\\
\Longrightarrow&\mathbf{W}^{-1}x\cdot\mathbf{W} \ge \mathbf{W}^{-1}\mathbf{W}diag\{\frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1-e^{-\lambda_i}}\}\mathbf{W}^{-1}\mathbf{W}\\
\Longrightarrow&x\cdot\mathbf{I} \ge diag\{\frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1-e^{-\lambda_i}}\}\\
\Longrightarrow&x \ge \frac{e^{-\lambda_i(1-x)} - e^{-\lambda_i}}{1-e^{-\lambda_i}} \Longrightarrow \frac{1-e^{-\lambda_i(1-x)}}{1-e^{-\lambda_i}} - (1-x) \ge 0.
\end{aligned} \tag{10}$$

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

Theorem 3 indicates that using two speeds for a step-up schedule leads to a higher peak temperature in the stable status than using a constant speed. Furthermore, we prove that using two neighboring speeds benefits the peak temperature for a step-up schedule as follows.

Theorem 4. *Let $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$ be two periodic step-up schedules that are exactly the same except for $core_i$ during interval $[t_{h-1}, t_{h+1}]$. Assume that in $\mathbb{S}_{u1}(t)$, $core_i$ uses two modes with voltages $v_{i,h}$ and $v_{i,(h+1)}$, while in $\mathbb{S}_{u2}(t)$, $core_i$ uses $v'_{i,h}$ and $v'_{i,(h+1)}$ such that (i) $core_i$ completes the same workload in both $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u2}(t)$; (ii) $v'_{i,h} \le v_{i,h} \le v_{i,(h+1)} \le v'_{i,(h+1)}$. Then we have $T_{peak}(\mathbb{S}_{u1}(t)) \le T_{peak}(\mathbb{S}_{u2}(t))$.*

Proof. As shown in Fig. 1, within interval $[t_{h-1}, t_{h+1}]$ on $core_i$ we define a third same throughput schedules $\mathbb{S}_{u3}(v_{i,h}, v'_{i,(h+1)})$ and let the h_{th} interval of $\mathbb{S}_{u1}(t)$, $\mathbb{S}_{u2}(t)$ and $\mathbb{S}_{u3}(t)$ change their modes at t_{h1}, t_{h2} and t_{h3}, respectively. Then we have $t_{h1} \le t_{h3} \le t_{h2}$. Note that $\mathbb{S}_{u1}(t)$ and $\mathbb{S}_{u3}(t)$ are of the same modes within interval $[0, t_{h1}]$; however, $\mathbb{S}_{u3}(t)$ uses two modes to complete the tasks within $[t_{h1}, t_{h+1}]$, while $\mathbb{S}_{u1}(t)$ use a constant mode. From Theorem 3, we can conclude that $\mathbf{T}(\mathbb{S}_{u3}(t_{h+1})) \ge \mathbf{T}(\mathbb{S}_{u1}(t_{h+1}))$. Then in the following intervals, the temperature of $\mathbb{S}_{u3}(t)$ will always be higher than $\mathbb{S}_{u1}(t)$. Thus, we can conclude $T_{peak}(\mathbb{S}_{u3}(t)) \ge T_{peak}(\mathbb{S}_{u1}(t))$. Similar method can also be applied to prove $T_{peak}(\mathbb{S}_{u2}(t)) \ge T_{peak}(\mathbb{S}_{u3}(t))$. Therefore, $T_{peak}(\mathbb{S}_{u2}(t)) \ge T_{peak}(\mathbb{S}_{u1}(t))$. □

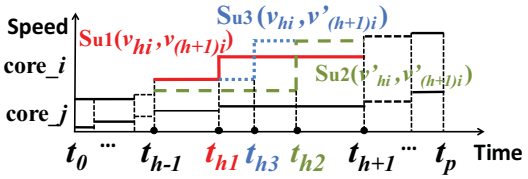


Figure 1. Illustration for Theorem 4.

From Theorem 3 and 4, when constructing a multi-core schedule to maximize the throughput, it is highly desirable to use a single mode with a constant voltage if it is available. Otherwise, we should choose the two neighboring modes, which are most likely to achieve a better throughput than other choices.

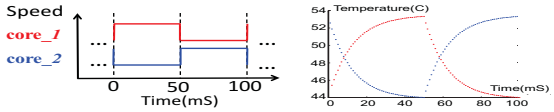
C. m-Oscillating Schedule on Multi-core Platforms

After the running modes are chosen, the problem now becomes how frequently we should oscillate the modes with high and low voltages to maximize the throughput. Recall that, our motivation example shows that different periods leads to different throughput performance. For single processor platforms, Huang et al. [25] proposed a strategy called the *m-Osillating method* for a two-voltage schedule. According to this method, each scheduling interval is evenly divided into m sections, and the processor interleaves the high and low-voltage sections. They further showed that the larger the m, the lower the peak temperature is.

To study if the m-Osillating method works for multi-core platforms, we conducted the following example. We set up a schedule on a two-core platform with a period of $100ms$, each core running at equal times using two processing modes, with high-voltage $v_H = 1.3V$ and low-voltage $v_L = 0.6V$, as shown in Fig. 2(a). Fig. 2(b) shows the stable status temperature trace within one period, with a peak temperature of $53.3°\mathrm{C}$. Next, we let $core_1$ double its oscillating frequency and $core_2$ keep the same schedule, as shown in Fig. 2(c). In the stable status, as shown in Fig. 2(d), the peak temperature becomes $54.6°\mathrm{C}$, which is higher than the previous one.

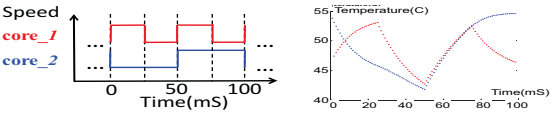
This example clearly shows that the frequency oscillation scheme performed only on one core does not necessarily reduce the peak temperature in a multi-core platform. Can peak temperature be reduced if we simultaneously scale the oscillating frequencies for all cores? In regard to this, we first formally define the m-Oscillating schedule for a multi-core platform as follows.

Definition 3. *Let $\mathbb{S}(t)$ be a 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 by m times without changing its voltage levels.*



(a) $core_1$ runs at 1.3V and 0.6V within $(0, 50)ms$ and $(50, 100)ms$, respectively. $core_2$ uses 0.6V for the first interval and 1.3V for the second.

(b) Stable status temperature trace for schedule in Fig. 2(a).



(c) $core_1$ doubles its oscillating frequency for schedule in Fig. 2(a).

(d) Stable status temperature trace for schedule in Fig. 2(c).

Figure 2. m-Oscillating for one core does not always reduce the peak temperature.

Then, for multi-core m-Oscillating schedules, we found oscillating all cores helps to reduce the peak temperature. The proof is rather long and thus omitted due to page limit.

Theorem 5. *Let $\mathbb{S}(t) = \{\mathbb{I}_q : q = 1 \cdots z\}$ be a periodic step-up schedule with period of t_p on multi-core processor. Then, $T_{peak}(\mathbb{S}(m, t)) \geq T_{peak}(\mathbb{S}(m + 1, t))$.*

V. Throughput Maximization Using Frequency Oscillation

With the principles and observations on thermal characteristics presented above, we are ready to introduce our approach for throughput maximization on multi-core platforms. Our approach consists of three steps: we first calculate the ideal constant voltage/frequency for each core that can maximize the throughput of the given multi-core platform. If such a voltage/frequency is not available, we choose two neighboring discrete modes to form the "step-up" schedule (Theorem 4) and then use m-Oscillating scheme to lower the peak temperature (Theorem 5). We further adjust the execution time ratio of different modes when necessary for each core to meet the temperature constraint. The detailed approach is explained below.

As the starting point of our approach, we use a similar method to Hanumaiah et al. [21] to find the single constant mode (with $\mathbf{v}_{const}$) on each core to maximize the throughput. Specifically, we assume the stable state temperature for each core equals to T_{max}, i.e. $\mathbf{T}^{\infty}(\mathbf{v}_{const}) = [T_{max}]_{N\times 1}$. The power consumption therefore can be calculated by letting $\frac{d\mathbf{T}}{dt} = \mathbf{0}$ and $\mathbf{T} = [T_{max}]_{N\times 1}$ in (2), and the optimal voltage for each core can be calculated as $v_i = \sqrt[3]{(P_i - \alpha(v_i) - \beta T_{max})/\gamma(v_i)}$. With the knowledge of the single constant mode of v_i defined for the i_{th} core, the available high-voltage $v_{i,H}$ and low-voltage $v_{i,L}$ and their execution time ratios $r_{i,H}$ and $r_{i,L}$ that maintain the same throughput are also defined as

$$\begin{cases} v_{i,H} \cdot r_{i,H} + v_{i,L} \cdot r_{i,L} = v_i \\ r_{i,H} + r_{i,L} = 1. \end{cases} \tag{11}$$

Next, we need to find the proper value of m that can make the best tradeoff between temperature reduction and accumulated transition overhead, which is unfavorable for throughput maximization. Without considering the transition overhead, the multi-core peak temperature monotonically decreases as m increases. However, in practical scenario, each DVFS transition stalls the program execution for a small time interval. Assume that the clock will be halted for interval τ during the voltage transition by DVFS. For each transition on $core_i$, it causes $(v_{i,H} + v_{i,L})\tau$ performance loss. To keep the same throughput requires extending the high-voltage mode and reducing the low-voltage mode for $\delta_i = \frac{(v_{i,H}+v_{i,L})\tau}{v_{i,H}-v_{i,L}}$ seconds. However, there is an upper bound $M_i = m_{i,max}(\tau)$ because low-voltage interval $t_{i,L}$ of $core_i$ should be large enough to cover the voltage transitions as $M_i = \lfloor \frac{t_{i,L}}{\delta_i + \tau} \rfloor$. Then, the chip-wide upper bound of $M = min\{M_i | core_i \in \mathfrak{N}\}$. Thanks to the fact that the peak temperature of a step-up schedule can be easily calculated, it is affordable to search the optimal m using the linear sequential search method.

Once the optimal m is found, the resulting m-Oscillating schedule is likely to have a peak temperature higher than T_{max} (Theorem 3). Therefore, we need to further adjust the different modes' execution time ratios to meet the temperature constraint. To do this, we first order cores by their peak temperatures. Then, we select the core with the highest peak temperature to lower its temperature. Note that, due to the linearity of the system, we can reduce the high-voltage mode execution time ratio for any core to reduce its peak temperature. To find the core that can most effectively reduce the peak temperature (i.e. $core_i$) with the minimum throughput loss, we define a metric called *temperature performance tradeoff index* for $core_i$, denoted as $\mathbf{TPT}_{core_i}$. Specifically, $\mathbf{TPT}_{core_i}(j) = \frac{\Delta T_i}{|v_{j,H}-v_{j,L}|\times t_{unit}}$ is the ratio of temperature reduction at $core_i$ to the throughput loss at $core_j$ when changing the high-voltage interval to the low-voltage interval for one unit of time, i.e. t_{unit}, on $core_j$. We modify the schedule for the core with the highest $\mathbf{TPT}_{core_i}$, and this procedure continues until the temperature constraint is satisfied. The overall algorithm is illustrated in Algorithm 2, with complexity $O(M + \frac{t_p}{t_{unit}}N)$.

Note that in our approach, we require each m-Oscillating schedule to be a step-up schedule. This decision is really a double-edged sword. A step-up schedule allows us to quickly determine the highest temperature in a schedule to ensure peak temperature constraint is guaranteed. In the meantime, however, since temperature varies with power density, the schedules that can interleave the intervals with high and low-voltage modes, not only temporally but also spatially, lead to peak temperature lower than a step-up schedule. We study the potential impacts of this observation with experiments in the following section.

VI. Experimental Results

In this section, we validate our theorems and study the performance of our proposed approaches through simulations.

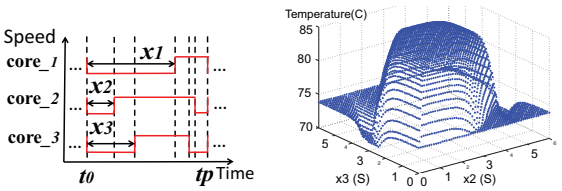
Since we study the system-level temperature-related problem, the processing cores consume most part of the power consumption. We therefore simplify the floor-plan to the core-level and adopt the parameters of thermal capacitance and resistance from HotSpot-5.02 [10] at $65nm$ technology node. Power parameters are abstracted from the McPAT simulator [36].

In our simulations, we used different multi-core configurations, i.e. 2×1, 3×1, 3×2 and 3×3 layout, with $4\times4mm^2$ core

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

Input: Multi-core platform $\mathfrak{N} = \{core_i : i = 1 \cdots N\}$;
Transition overhead parameters: τ, δ_i;
Temperature threshold: T_{max};
Unit time: t_{init}
Output: The m-Oscillating schedule $\mathbb{S}(m^{opt}, t)$ and throughput THR (equation 5)

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



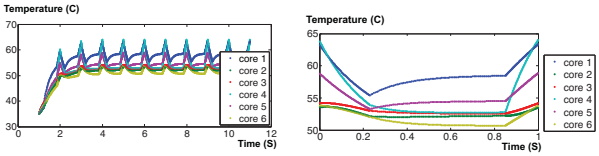
(a) A 3-core schedule with shifting the high-speed mode. (b) Peak temperature varies when shifting its high-speed modes.

Figure 3. Peak temperatures varies when shifting its high-speed modes.

size. We assumed that the available supply voltages for each core are in the range of $[0.6V, 1.3V]$ with $0.05V$ step size. The ambient temperature was set to be $T_{amb} = 35\,^{\circ}\mathrm{C}$.

A. Step-up schedule can bound the peak temperature of the corresponding arbitrary schedules

As an experiment, we ran a large number of tests for schedules, shown in Fig. 3(a), on a 3-core platform. We set the period as 6 seconds and let running modes with high-voltage $1.3V$ and low-voltage $0.6V$ execute 3 seconds on each core. Let $core_1$'s x_1 equal to the length of its low-voltage mode, then we change x_2 of $core_2$ and x_3 of $core_3$ by a 0.1 second step size each time. Fig. 3(b) shows the peak temperature changes with different high-speed starting time of $core_2$ and $core_3$, i.e. x_2 and x_3. The highest peak temperature can reach $84.13^{\circ}\mathrm{C}$, when $x_2 = x_3 = 3$ seconds; and the lowest peak temperature is $71.22^{\circ}\mathrm{C}$, when $x_2 = 0.6$ and $x_3 = 4.2$ seconds. We are able to see that the step-up schedule bounds the peak temperature of its corresponding *arbitrary schedules*.



(a) Temperature trace for a 6 core step-up schedule, starting from the stable-status. $T_{amb} = 35^{\circ}\mathrm{C}$. (b) Temperature trace within one period in the stable-status.

Figure 4. A step-up schedule temperature trace on a 6-core processor.

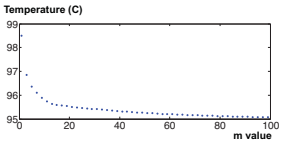


Figure 5. The peak temperature of a 9-core m-Oscillating schedule monotonically decreases with m.

B. Thermal Characteristics for Step-up Schedule and m-Oscillating Schedule

We verified the thermal characteristics for the *step-up schedule* and the m-Oscillating schedule identified in Theorem 1 and Theorem 5.

By randomly selecting period and creating non-decrease speed levels within one period, we generate a large set of random step-up schedules and collected the temperature traces using HotSpot-5.02 [10], and the results confirm that the peak temperature of a step-up schedule always occurs at the end of the period in the stable status. Fig. 4 shows the temperature trace for a step-up schedule on a 6-core platform. The period of the schedule is set to be 1 second and each core has up to 3 different intervals. As we can see from Fig. 4(a), when starting from the ambient temperature, the temperature of each core monotonically increases and reaches its peak at the end of the period, which conforms to Theorem 1.

We also verified Theorem 5 with our experiments. Fig. 5 shows the maximum temperature changes for the m-Oscillating schedule on a 9-core platform. The original multi-core schedule was randomly generated with a period of 9.836 seconds, each core with up to 5 intervals. As shown in Fig. 5, the peak temperature monotonically decreases when m increases, exactly as predicted by Theorem 5.

C. Performance Comparison of Different Approaches

Table IV. Different numbers of modes with different voltages.

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

Next, we studied the performance of our proposed approach. We generated different multi-core configurations and obtained the corresponding thermal model from HotSpot-5.02 [10] using the matrix modeling method presented in [23]. The voltage

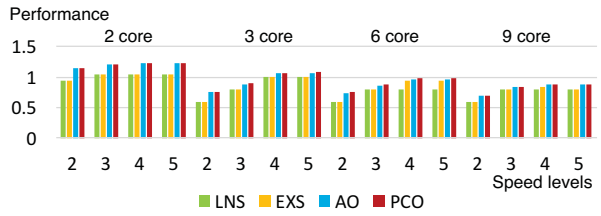


Figure 6. Performance comparisons with different numbers of cores and voltage levels.

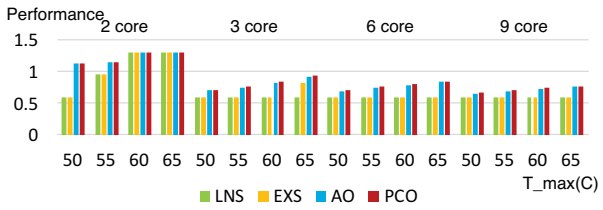


Figure 7. Performance comparisons with different numbers of cores and different T_{max}.

switching overhead was set to $5us$. The temperature threshold was set to be $T_{max} = 55\,^{\circ}\mathrm{C}$.

We compared four approaches in our study. The first one *lower neighboring speed* method (**LNS**) uses the lower neighboring modes when the continuous voltage/frequency levels are not available. The second is *exhaustive search approach* (**EXS**), as illustrated in Algorithm 1. The third approach, *aligned oscillation* (**AO**), is our proposed approach, depicted in Algorithm 2. As mentioned in section V, **AO** approach requires each candidate schedule to be a step-up schedule and thus may not optimize the performance. To evaluate the potential impacts of this factor, we developed another schedule *phase-conscious oscillation* (**PCO**) that interleaves the high/low-speed intervals among multiple cores spatially. Specifically, after the optimal value of m in **AO** is defined, we shift the schedule for each core individually and search the best high-speed mode starting time to interleave the high/low-voltage modes differently for each core, and use it in the **PCO** schedule.

Fig. 6 shows the performance comparison for four approaches on multi-core platforms with 2, 3, 6, 9 cores and different numbers of available voltage/frequency levels. The voltage levels were defined in Table IV. From Fig. 6, we can observe that **AO** and **PCO** always outperform **EXS** and **LNS**. It is interesting to see from Fig. 6 that, the fewer the available voltage levels, the greater the improvement that can be achieved by **AO** and **PCO** over **EXS** and **LNS**. This is because **LNS** and **EXS** allow only one constant voltage/frequency for each core, which can be much lower than the optimal one if fewer voltage choices are available. In addition, **EXS** searches all the combinations of voltages/frequencies, which deeper explored the design space than **LNS**. As shown in the figure, for 2 voltage levels, the average performance improvement by **AO** and **PCO** over **EXS** is 55.2%; and the improvement becomes 24.8% when the number of available voltage levels is 5.

It is also interesting to see that the performance of **AO** and **PCO** are very close. Even though our simulation study in section VI-A shows large differences for schedules with different high-speed mode starting time, this happens only when the period of the schedule is long, e.g. 6 seconds, for the results in section VI-A. As both **AO** and **PCO** adopt the m-Oscillating schemes, the scheduling periods are significantly reduced and therefore the differences become really insignificant, as shown in Fig. 6.

The same conclusions can be drawn from our experimental results by changing the temperature threshold, as shown in Fig. 7. Specifically, we set the temperature threshold T_{max} range from $50\,^{\circ}\mathrm{C}$ to $65\,^{\circ}\mathrm{C}$, with $5\,^{\circ}\mathrm{C}$ step size, and we used only 2 levels of voltages, as shown in Table IV. From Fig. 7, it is no surprise to see that, the higher the T_{max}, the larger the throughput by each approach. Note that for a 2-core platform, when the temperature threshold is larger than $55^{\circ}\mathrm{C}$, all three approaches have the same performance. This is because all of the cores can run at the highest voltage/frequency without violating the temperature constraints. For a 6-core platform, **AO** and **PCO** can improve over **EXS** by 40.4% when $T_{max} = 65\,^{\circ}\mathrm{C}$.

D. Computation Time Comparison

We also compared the computational efficiency of our approach **AO** and **PCO** with **EXS**, since **EXS** has a better performance than **LNS**. In this study, we set $T_{max} = 65\,^{\circ}\mathrm{C}$ and 2, 3, 4, 5 speed levels for each core. We randomly generated up to 100 test cases under each configuration. The average CPU time for each approach under each case was collected, as shown in Table V. From Table V, we can see that as the number of cores and voltage levels increase, the computation time for each approach increases in general. However, the computation time for **EXS** increases at a much faster pace than **AO** and **PCO**. When the number of cores and speed levels is small, for example, for 2 or 3 cores, **EXS** takes less time than **AO** since **AO** needs to adjust high/low-speed ratios. However, as core numbers and voltages/frequencies increase, the computation time for **EXS** increases dramatically. For example, for 9 cores with 4 voltages, the **EXS** method takes as long as 581.14 seconds, while the **AO** takes only 5.12 second. For 9 core with 5 voltages, **EXS** takes over 2 hours, while **AO** method takes only 14.87 seconds. In all the cases, **AO** shows shorter computation time than **PCO**, since **PCO** needs to search the best high-speed starting time to further minimize peak temperature based on **AO**; then **PCO** adjusts high/low-speed ratio again to fill the peak temperature "headroom". In summary, the experimental results clearly demonstrate that our approach **AO** is both effective and computationally efficient in maximizing the throughput under a given peak temperature constraint.

VII. Conclusions

As IC technology continues to scale, the thermal problem is becoming a more and more serious concern in the design of high-performance computing systems. In this paper, we present a novel technique to maximize the throughput of a multi-core platform under a given peak temperature constraint. Our

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

	Scheme	2 levels	3 levels	4 levels	5 levels
2 cores	**AO**	0.13	0.08	0.08	0.16
	PCO	0.27	0.17	0.16	0.30
	EXS	0.01	0.01	0.01	0.01
3 cores	**AO**	8.97	7.27	5.24	3.97
	PCO	16.82	16.79	15.73	16.65
	EXS	0.01	0.01	0.02	0.03
6 cores	**AO**	10.55	8.88	5.15	5.93
	PCO	19.25	18.64	15.64	19.87
	EXS	0.13	1.36	8.01	28.22
9 cores	**AO**	19.33	15.03	5.12	14.87
	PCO	110.67	104.55	90.10	106.10
	EXS	1.53	43.36	581.14	>2-hours

techniques are built upon two novel concepts, i.e. the *step-up schedule* and the *m-Oscillating schedule*, and their interesting thermal characteristics for multi-core platforms. The significance of this work lies in the fact that our proposed methods not only reduce the computation time from the exhaustive search by orders of magnitude but also improves the throughput up to 89%, with an average improvement of 11%. More importantly, the fundamental principles established in this paper are general enough to be readily used for other thermal related research.

ACKNOWLEDGMENT

This work is supported in part by NSF under project CNS-1565474.

REFERENCES

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

[2] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 329–339, 2010.

[3] C. Zweben, "Ultrahigh-thermal-conductivity packaging materials," in *STHERM*, pp. 168–174, 2005.

[4] G. H. Loh and Y. Xie, "3d stacked microprocessor: Are we there yet?," *IEEE Micro*, vol. 30, no. 3, pp. 60–64, 2010.

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

[6] S. Sha, J. Zhou, C. Liu, and G. Quan, "Power and energy analysis on intel single-chip cloud computer system," in *IEEE Southeastcon*, pp. 1–6, 2012.

[7] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. 31, no. 4, pp. 6–15, 2011.

[8] C. Isci, A. Buyuktosunoglu, C. y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *MICRO*, pp. 347–358, 2006.

[9] S. Pagani, H. Khdr, W. Munawar, J. J. Chen, M. Shafique, M. Li, and J. Henkel, "Tsp: Thermal safe power - efficient power budgeting for many-core systems in dark silicon," in *CODES+ISSS*, pp. 1–10, 2014.

[10] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, 2004.

[11] S. S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *ICCAD*, pp. 281–288, 2007.

[12] H. F. Sheikh, I. Ahmad, Z. Wang, and S. Ranka, "An overview and classification of thermal-aware scheduling techniques for multi-core processing systems," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 3, pp. 151 – 169, 2012.

[13] M. Gomaa, M. D. Powell, and T. N. Vijaykumar, "Heat-and-run: Leveraging smt and cmp to manage power density through the operating system," in *ASPLOS*, pp. 260–270, 2004.

[14] J. Zhou, T. Wei, M. Chen, J. Yan, S. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2015.

[15] T. Ebi, H. Amrouch, and J. Henkel, "Cool: Control-based optimization of load-balancing for thermal behavior," in *CODES+ISSS*, pp. 255–264, 2012.

[16] B. Yun, K. G. Shin, and S. Wang, "Predicting thermal behavior for temperature management in time-critical multicore systems," in *RTAS*, pp. 185–194, 2013.

[17] A. Coskun, T. Rosing, K. Whisnant, and K. Gross, "Static and dynamic temperature-aware scheduling for multiprocessor socs," *IEEE Transactions on VLSI Systems*, vol. 16, no. 9, pp. 1127–1140, 2008.

[18] G. L. Liu, M. Fan, and G. Quan, "Neighbor-aware dynamic thermal management for multi-core platform," in *DATE*, pp. 187–192, 2012.

[19] R. Rao and S. Vrudhula, "Efficient online computation of core speeds to maximize the throughput of thermally constrained multi-core processors," in *ICCAD*, pp. 537–542, 2008.

[20] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli, "Temperature-aware processor frequency assignment for mpsocs using convex optimization," in *CODES+ISSS*, pp. 111–116, 2007.

[21] V. Hanumaiah, S. Vrudhula, and K. S. Chatha, "Performance optimal online dvfs and task migration techniques for thermally constrained multi-core processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1677–1690, 2011.

[22] M. Kadin and S. Reda, "Frequency planning for multi-core processors under thermal constraints," in *ISLPED*, pp. 213–216, 2008.

[23] Z. Wang and S. Ranka, "Thermal constrained workload distribution for maximizing throughput on multi-core processors," in *IGCC*, pp. 291–298, 2010.

[24] T. Chantem, X. S. Hu, and R. P. Dick, "Online work maximization under a peak temperature constraint," in *ISLPED*, pp. 105–110, 2009.

[25] H. Huang, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," in *DAC*, pp. 363–368, 2011.

[26] T. Wang, M. Fan, G. Quan, and S. Ren, "Heterogeneity exploration for peak temperature reduction on multi-core platforms," in *ISQED*, pp. 107–114, 2014.

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

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

[29] L. Schor, I. Bacivarov, H. Yang, and L. Thiele, "Worst-case temperature guarantees for real-time applications on multi-core systems," in *RTAS*, pp. 87–96, 2012.

[30] Q. Han, M. Fan, O. Bai, S. Ren, and G. Quan, "Temperature-constrained feasibility analysis for multi-core scheduling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[31] V. Chaturvedi, H. Huang, S. Ren, and G. Quan, "On the fundamentals of leakage aware real-time dvs scheduling for peak temperature minimization," *Journal of Systems Architecture*, vol. 58, no. 10, pp. 387–397, 2012.

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

[33] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *Journal of the ACM*, vol. 54, no. 1, p. 3, 2007.

[34] G. D. Poole, "Generalized m-matrices and applications," *Mathematics of Computation*, vol. 29, no. 131, pp. 903–910, 1975.

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

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