Journal of
Low Power Electronics
Vol. 8, 378–393, 2012

Leakage Aware Scheduling on Maximum Temperature Minimization for Periodic Hard Real-Time Systems

Huang Huang*, Vivek Chaturvedi, Guanglei Liu, and Gang Quan

Department of Electrical and Computer Engineering, Florida International University, Miami, FL, 33174, USA

(Received: 28 January 2012; Accepted: 8 June 2012)

Over the years, the chip power density has been increased exponentially due to the increasingly complicated circuit architecture as well as the continuous miniaturization of the transistor feature size. High power consumption has directly translated to high chip temperature which adversely affects the system performance/reliability and increases the cooling/packaging costs. Moreover, high chip temperature also elevates the leakage power consumption, which further augments the overall power consumption and thus the operating temperature. In this paper, we incorporate the leakage/temperature dependency as well as the nonnegligible transition overhead into analysis and present a novel real-time speed scheduling algorithm, namely M-Oscillating, that can reduce the peak temperature of a system when executing a hard real-time periodic task set. We analytically prove the correctness of the proposed algorithm based on a processor model that can effectively account for the leakage/temperature relationship. We validate the effectiveness of the proposed algorithm by comparing it with the existing work on two platforms. The first platform is a C/Matlab based chip-level thermal/power simulator, and the second platform is a more practical one built based on a desktop computer running SPEC CPU2000 benchmark programs. The experimental results obtained from both platforms demonstrate the superiority of the proposed M-Oscillating scheme over the existing approach in peak temperature reduction and feasibility improvement.

Keywords: Peak Temperature Reduction, Real-Time Systems, Scheduling, Leakage, Thermal Aware.

1. INTRODUCTION

The human's pursuing of high performance computing systems has driven the semiconductor technology into the deep sub-micron era. The increasingly complicated circuit architecture together with the continuous shrinking of transistor feature size has resulted in an exponential increase of power density. The escalated power consumption of IC circuits has directly translated to an elevated temperature which not only raises the packaging/cooling costs, but also degrades the performance/reliablity of the computing system and shortens its life span.[34, 40] Moreover, due to the strong leakage/temperature dependency, the soaring chip temperature drastically increases the leakage power, which is becoming a major contributor of the overall power consumption in the deep sub-micron domain. According to Ref. 23, the leakage power can increase by 38% when chip temperature rising from 65 °C to 110 °C. High power consumption leads to high temperature which in turn, hikes up the leakage power and thus the overall power consumption. Consequently, a thermal aware design technique will become ineffective if the interdependency between leakage and temperature is not properly addressed in the deep sub-micron domain.

Previous researches have studied the temperature/leakage interdependency in depth. Based on the circuit-level analysis, a complex relationship between the leakage and temperature is established by Refs. [23, 43], where the leakage current is formulated as

$$I_{\text{leak}} = I_s \cdot (\mathscr{A} \cdot T^2 \cdot e^{((\alpha \cdot V_{dd}+\beta)/T)} + \mathscr{B} \cdot e^{(\gamma \cdot V_{dd}+\delta)}) \qquad (1)$$

where I_s is the leakage current at certain reference temperature and supply voltage, T is the operating temperature, V_{dd} is the supply voltage, $\mathscr{A}, \mathscr{B}, \alpha, \beta, \gamma, \delta$ are empirically determined technology constants. By using these relations, many practical power and thermal analysis tools were developed, for example the "HotSpot,"[2] which was developed to simulate and study the processor thermal phenomena at the architectural-level. Though Eq. (1) is suitable for developing tools like "HotSpot," it

*Author to whom correspondence should be addressed.
Email: hhuan001@fiu.edu

is fairly complexto be used for system-level analysis, such as real-time feasibility analysis or scheduling technique development.

In this paper, we investigate how to apply the real-time scheduling techniques to solve the peak temperature minimization problem. We present a novel real-time scheduling technique, i.e., the *M-Oscillating* algorithm, that oscillates the high and low processor speeds to minimize the peak temperature when executing a periodic task set. We have formally proved the effectiveness of the *M-Oscillating* algorithm based on a processor power model that can capture the leakage/temperature dependency with reasonable accuracy, and at the same time, is simple enough and thus suitable for the system-level analysis. Moreover, we extend the ideal *M-Oscillating* algorithm into non-ideal scenarios by incorporating a more realistic non-negligible transition overhead model into the algorithm. To evaluate the effectiveness of the proposed algorithm, we compare our approach with an existing scheduling scheme on two different platforms. The first platform is a simulation platform built based on a theoretic chip-level thermal model and a linear leakage/temperature dependency model. To make our thermal analysis more concrete, we develop a more practical hardware platform based on a desktop computer running the Linux OS. The run-time temperature can be captured by reading the on-chip thermal sensor when executing various benchmark programs, i.e., SPEC CPU2000 benchmark.[1] The experimental results obtained from both platforms confirm the superiority of the proposed *M-Oscillating* over existing scheduling methods in terms of peak temperature minimization and feasibility improvement. Moreover, our work clearly shows that a more rigorous and analytical system level analysis of leakage/temperature relationship is not only possible but necessary.

The rest of the paper is organized as follows. Section 2 discusses the related work. The system models we used in this paper are described in Section 3. The proposed *M-Oscillating* scheduling algorithm is introduced in Section 4. Section 5 introduces the non-negligible transition overhead model and extends the proposed *M-Oscillating* scheduling into the non-ideal scenarios. Experimental results obtained from both the chip-level C/Matlab simulator and the real platform are presented in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORK

In this paper, we study the problem on how to apply the real-time scheduling technique to minimize the peak temperature when executing a period hard real-time task set. As related work, there are a number of papers published on either minimizing the overall energy consumption (e.g., Refs. [5, 6, 11, 17, 21, 38]) or maximizing the system throughput (e.g., Refs. [8, 15, 16, 19, 31, 35, 36, 42]) under the maximum temperature/energy constraint. Specifically, Bao et al.[6] proposed to distribute the idle interval wisely when scheduling a task graph such that the temperature of the processor can be effectively "cooled down" and thus reduce the leakage power consumption. In this approach, the leakage power is modeled using a piecewise linear function of temperature under different supply voltage levels. Yang et al.[38] proposed a quadratic leakage model to simplify the leakage/temperature dependency. Based on this model, they proposed a "pattern-based" scheduling approach to periodically switch the processor between the active and dormant modes and reduce the energy. Huang and Quan[17] derived a closed-form energy calculation equation based on which they proposed an energy minimization scheduling method by extending the concept of *M-Oscillating* approach in Ref. [9]. Wang et al.[35, 36] studied the maximum delay when scheduling periodic tasks based on a two-speed scheduling policy, i.e., using the highest processor speed to run the tasks until the temperature reaches the acceptable maximum temperature, and then using an equilibrium processor speed to maintain the temperature. In contrast, by separating the silicon die and the package in the RC thermal model, Rao et al. 31 analytically derived an exponential speed throttling curve to maintain the processor die temperature at the given upper bound value once reached. Chantem et al.[8] proposed to run real-time tasks by frequently switching between the two speeds which are neighboring to a constant speed whose stable temperature is the given peak temperature. Hanumaiah et al.[16] proposed a *zero-slack* policy to maximize the performance of a multi-core platform. The binary search is used to find the optimal speeds and voltage levels. A simplified *Hotspot*-like thermal model is derived to reduce the computation time so that the algorithm can be effectively implemented during run-time. By exploiting the frequency scaling together with the task migration, Hanumaiah et al.[15] further proposed an online algorithm to maximize the throughput of a multi-core real-time system under a given thermal constraint. Zhang and Chatha[42] developed several algorithms to maximize the throughput of a real-time system by sequencing the execution of a task set consisting of tasks with heterogeneous power/thermal characteristics for processors with and without dynamic voltage/frequency scaling (DVFS) capability.

There are extensive work published recently on how to optimize the operational temperature by using real-time scheduling techniques. Specifically, in Refs. [4, 10], the researchers aimed to identify the upper bound of the maximum temperature. Some others (e.g., Refs. [3, 4, 7, 13, 32, 39]) intended to minimize the peak temperature or to guarantee the given maximum temperature constraints when scheduling a task set or a single copy of a task graph. In Ref. [24], a thermal-aware scheduling algorithm with stochastic workloads is presented to effectively avoids thermal emergencies by reducing peak temperature. Kumar et al.[22] derived a stop-and-go algorithm to schedule a task graph with *just enough* idle period so that the peak temperature is minimized while ensuring the makespan constraint. Jayaseelan et al.[20] studied how to appropriately order the execution sequence of a task set consisting of tasks with different thermal profiles to minimize the peak temperature. Ahmed et al.[3] investigated how to minimize the peak temperature when running a sporadic task system scheduled according to *EDF*. They assume that the processor can work only in two running modes: active and idle modes, and they proposed to run tasks by periodically setting the processor to active mode and idle mode. Our research differs from these work not only by real-time task models, but more importantly, the leakage/temperature dependency formulations.

As we discussed in Section 1, the leakage/temperature dependency plays an important role in the study of thermal aware scheduling problem. Therefore, the techniques (such as those in 4) assuming no or constant leakage power consumption become ineffective. Some other researchers (e.g., Ref. [5, 41]) applied Eq. (1) directly to capture the leakage/temperature dependency for the scheduling analysis. There are also a number of other approaches that formulate the temperature-constrained problem as a convex optimization problem.[7, 26, 27] Even though the leakage/temperature dependency (Eq. (1)) may be incorporated into the convex optimization formulation,[26] its computational complexity is very high. Therefore, these approaches can only work at a system level when the design solution space is small.

Efforts have been made to simplify the leakage/temperature dependency model. Liu et al.[25] showed that using linear models is an effective way for accurate leakage estimation over the operating temperature ranges in real-time ICs. A number of researches (such as Refs. [8, 12, 14] adopt a simple temperature/leakage dependency model that assumes the leakage current changes linearly *only* with temperature. However, leakage power changes not only with temperature but also supply voltage as well (i.e., Eq. (1)). As evidenced in Ref. [18], the leakage model ignoring the effect of supply voltage can lead to results deviated far away from the actual values. Quan et al.[29] introduced a leakage/temperature model that is more practical. According to their model, a processor has different running modes, and leakage varies at different rates with temperature when running at different modes. Based on this model, they presented several conditions to verify the feasibility of a *given* real-time schedule. However, how to develop a feasible and effective schedule for a given periodic task set under the maximum temperature constraint remains the problem. In what follows, we develop a novel scheduling technique that can effectively reduce the maximum temperature when considering the interdependency between the leakage, temperature and the supply voltage levels.

3. PRELIMINARIES

The real-time system we considered in this paper consists of a number of real-time tasks with the same period (such as the MPEG decoder). We can thus simplify this model by assuming that the real-time system has only one periodic task. The period of the task is denoted as p and its worst-case workload is c. We further assume that the deadline of the task equals its period.

We use the RC thermal model that has been widely used in similar research (e.g., Refs. [8, 12, 17, 29]). Specifically, assuming a fixed ambient temperature (T_{amb}), let $T(t)$ be the temperature at time t, then we have

$$RC\frac{dT(t)}{dt} + T(t) - RP(t) = T_{\text{amb}} \tag{2}$$

where $P(t)$ denotes the power consumption (in $Watt$) at time t, and R, C denote the thermal resistance (in ${}^oC/W$) and thermal capacitance (in $J/{}^oC$). We can then scale T such that T_{amb} is zero and get

$$\frac{dT(t)}{dt} = aP(t) - bT(t) \tag{3}$$

where $a = 1/C$ and $b = 1/RC$.

The processor considered in this paper is assumed to be able to run in n different modes, with each mode characterized by (v_i, f_i), $i = 0, 1, \ldots, n-1$. where v_i is the supply voltage and f_i is the working frequency in mode i. We assume that $v_i < v_j$, if $i < j$. We also assume that the processor speed is in proportion to the supply voltage. In what follows, we use processor speed and supply voltage interchangeably.

Given a supply voltage level v, the power consumption is composed of dynamic power P_{dyn} and leakage power P_{leak}, i.e., $P = P_{\text{dyn}} + P_{\text{leak}}$. According to Ref. [23], the leakage power consumption can be effectively estimated as,

$$P_{\text{leak}} = N_{\text{gate}} \cdot I_{\text{leak}} \cdot v \tag{4}$$

where N_{gate} represents the number of gate, v is the voltage level, and I_{leak} can be formulated by using Eq. (1). As leakage current changes super linearly with temperature,[25] we can simplify P_{leak} and define the leakage power when the processor running in mode k as

$$P_{\text{leak}}(k) = C_0(k)v_k + C_1(k)Tv_k \tag{5}$$

where $C_0(k)$ and $C_1(k)$ are constants. As we can see from Eq. (5), the leakage power depends on both the supply voltage and the temperature.

The dynamic power consumption is independent of the temperature and can be formulated as $P_{dyn} = C_2 v_k^{\xi} (\xi > 0)$. We choose $\xi = 3$[30] in this paper.[a] Hence the total power consumption at processor mode k is

$$P(k) = C_0(k)v_k + C_1(k) \cdot Tv_k + C_2 v_k^3 \tag{6}$$

[a]Choosing other values will not change the conclusions in this paper.

Based on Eqs. (3) and (6), when the processor running in mode k, the temperature dynamics can be formulated as

$$\frac{dT(t)}{dt} = A(k) - BT(t) \tag{7}$$

where

$$A(k) = a(C_0(k)v_k + C_2 v_k^3) \tag{8}$$

$$B(k) = b - aC_1(k)v_k \tag{9}$$

Given an interval $[t_0, t_e]$, let the starting temperature be T_0, by solving Eq. (7), the ending temperature can be formulated as below:

$$\begin{aligned} T_e &= \frac{A(k)}{B(k)} + \left(T_0 - \frac{A(k)}{B(k)}\right)e^{-\mathrm{B(k)(t_e - t_0)}} \\ &= G(k) + (T_0 - G(k))e^{-\mathrm{B(k)(t_e-t_0)}} \end{aligned} \tag{10}$$

where

$$G(k) = \frac{A(k)}{B(k)} \tag{11}$$

In what follows, we use A_k, B_k and G_k to denote $A(k)$, $B(k)$ and $G(k)$, respectively when there is no confusion. Equations (6) to (10) form the basis of our system level thermal analysis with leakage/termpature interdependency taken into account.

To ease our presentation, we define a typical two-speed schedule, which will be used and referenced in the following discussions.

Definition 3.1 *A two-speed schedule $\hat{S}(S_1, S_2)$ within an interval $[t_0, t_p]$ is a feasible schedule that uses speed S_2 during interval $[x_1, x_2](t_0 \leq x_1 < x_2 \leq t_p)$, and S_1 for the rest of the interval $[t_0, t_p]$, or vice versa.*

Figure 1 shows a few variations of the two-speed schedules, based on which we formulate several important theorems.

Theorem 3.2 *Given a hard real-time periodic task τ, the maximal temperature when the processor reaches its thermal steady state does not depend upon the initial temperature.*

PROOF. Let us consider a two-speed schedule shown in Figure 2(A), where S_2 and S_1 denotes the high speed and low speed, respectively. t_1 and t_2 are the duration of S_1 and S_2 in the first period.

Based on Eq. (10), the temperature at $t = x$ and $t = t_p$ can be formulated as

$$T_x = G_2 + (T_0 - G_2)e^{-Bt_2}, \quad T_{t_p} = G_1 + (T_x - G_1)e^{-Bt_1}$$

From Ref. [29], the maximal temperature at the steady state can be formulated as

$$T_{\max} = \max(T_x^{\infty}, T_{tp}^{\infty})$$

where,

$$T_x^{\infty} = \frac{G_2(1 - e^{-\mathrm{B}t_2})}{1 - e^{-\mathrm{B}t_2}} = G_2 \tag{12}$$

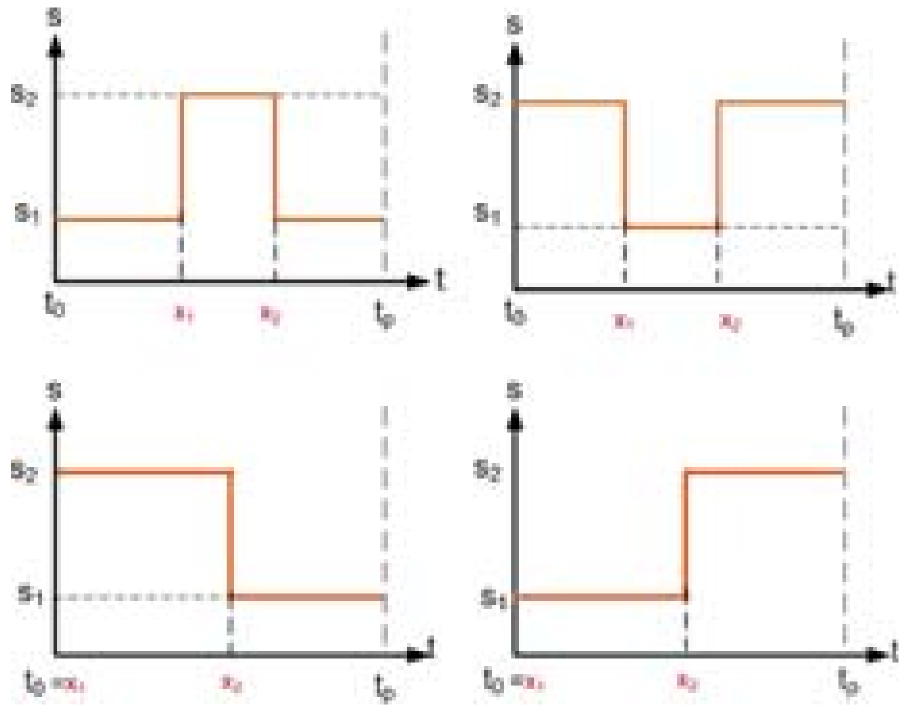

Fig. 1. Different two-speed schedules.

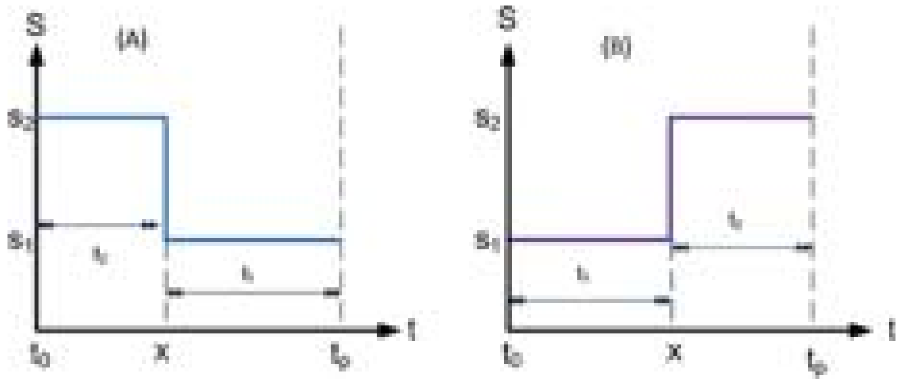

Fig. 2. Two-speed schedules within a given period p.

$$T_{tp}^{\infty} = \frac{G_1(1-e^{-\mathrm{B}t_1}) + G_2(1-e^{-\mathrm{B}t_2})e^{-\mathrm{B}t_1}}{1-e^{-\mathrm{B}(t_1+t_2)}} \tag{13}$$

As can be seen from Eqs. (12) and (13), no matter the maximal temperature occurs at $t = x$ or $t = t_p$, it does not depend upon the initial temperature T_0. Similar conclusion can be drawn by using other two-speed schedules. □

Based on Theorem 3.2, we can show that the peak temperatures of any two-speed schedule at the thermal steady state are the same.

Theorem 3.3 *Given a real-time periodic task τ and two processor speeds, the maximal temperatures at the thermal steady state with any two-speed schedules using the same speed levels and interval lengths are the same.*

Proof. Consider a periodic two-speed schedule as shown in Figure 3(A). From Ref. [29], we can calculate the steady state temperature or the peak temperature as:

$$T_{\max} = \frac{G_2(1-e^{-\mathrm{B}t_2}) + G_1(1-e^{-\mathrm{B}t_1})e^{-\mathrm{B}t_2})}{1-K} \tag{14}$$

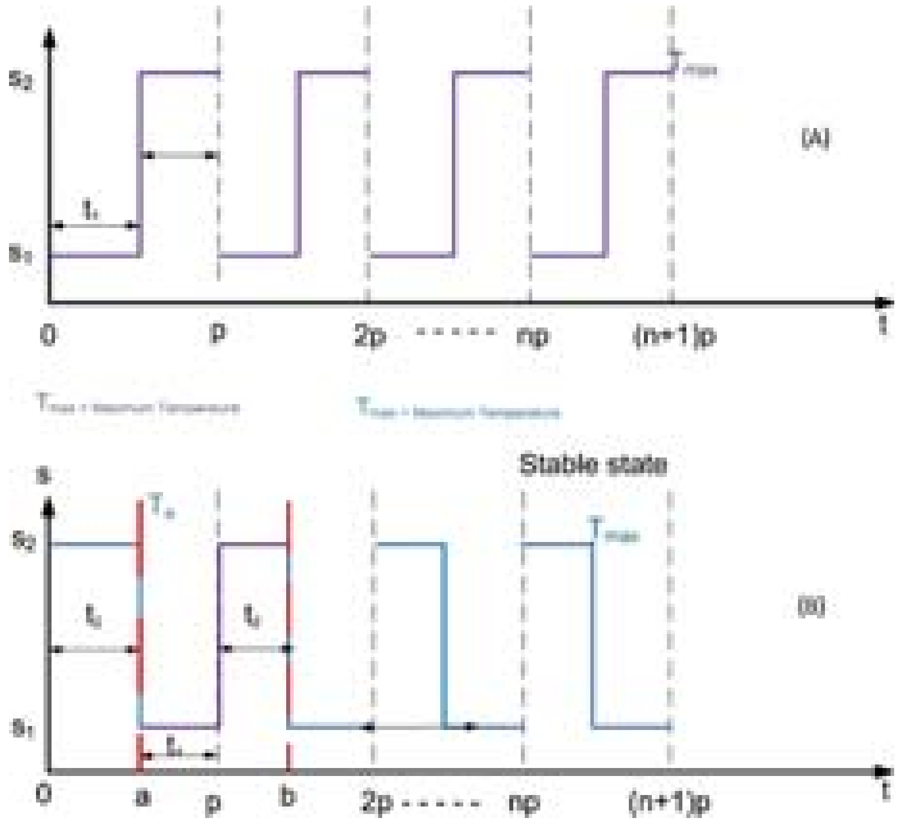

Fig. 3. Stable temperature for two-speed schedule.

where

$$K = e^{-\mathrm{B}(t_1+t_2)}$$

From Figure 3(B), we can see that the periodic two-speed schedule is the same as the one shown in Figure 3(A) but with an initial temperature T_a. Similarly all other periodic two-speed schedules can be considered as the one shown in Figure 3(A) with an initial shift. Therefore, they all have the identical peak temperature but different initial temperature. Since we have already shown in Theorem 3.2 that the steady state temperature does not depend on the starting temperature, therefore, the conclusion in Theorem 3.3 is also proved. □

4. PEAK TEMPERATURE MINIMIZATION

In this section, we present a novel scheduling approach that can effectively reduce the peak temperature when scheduling a periodic task set. Since high power consumption directly translates to high temperature, one intuitive idea is to employ existing power minimization scheduling techniques to solve the peak temperature minimization problem. However, thermal-aware scheduling problems have distinct characteristics in comparison with the power-aware scheduling problem as illustrated below.

Consider a simple two-speed schedule, as illustrated in Figure 4, that can finish a real-time job at its deadline. Note that, the dynamic energy consumption by the two-speed schedule remains constant as long as the length of each individual speed level keeps unchanged, i.e., t_1 and t_2 are constants. However, the temperature at $t = 1$ varies with the value of x. The following theorem captures this relationship.

Theorem 4.1 *Given a two-speed schedule as shown in Figure 4, and letting $S_2 > S_1$, if for any $S_2 > S_1$, we have $G_2 > G_1$ and $B_1, B_2 > 0$ (with G_k, B_k defined in Eqs. (11) and (9), respectively), then the temperature at $t = 1$, i.e., T_e, is a monotonically increasing function of x.*

Proof. Based on Eq. (10), let T_i and T_a be the temperature at time instants i and a, respectively. For simplicity, let the starting temperature be zero, then we have

$$\begin{aligned} T_i &= G_1(1 - e^{-\mathrm{B}_1 x}) \\ T_a &= G_2 + (T_i - G_2)e^{-\mathrm{B}_2 t_2} \end{aligned} \tag{15}$$

and the ending temperature is:

$$T_e = G_1 + (T_a - G_1)e^{-\mathrm{B}_1(t_1 - x)} \tag{16}$$

Replacing T_a with Eq. (15) we get,

$$T_e = G_1(1 - e^{-(\mathrm{B}_1 t_1 + \mathrm{B}_2 t_2)}) + (G_2 - G_1)(1 - e^{-\mathrm{B}_2 t_2})e^{-\mathrm{B}_1(t_1 - x)} \tag{17}$$

Therefore,

$$\begin{aligned} \frac{d(T_e)}{dx} &= \frac{d}{dx}(G_1(1 - e^{-(\mathrm{B}_1 t_1 + \mathrm{B}_2 t_2)}) \\ &\quad + (G_2 - G_1)(1 - e^{-\mathrm{B}_2 t_2})e^{-\mathrm{B}_1(t_1 - x)}) \\ &= (G_2 - G_1)(1 - e^{-\mathrm{B}_2 t_2})B_1 e^{-\mathrm{B}_1(t_1 - x)} \end{aligned} \tag{18}$$

From Eq. (18), it is not difficult to see that if $G_2 > G_1$ and $B_1, B_2 > 0$, the first order derivative, i.e., $d(T_e)/dx$, is always greater than zero which indicates that T_e is a monotonically increasing function of x. □

For the two-speed schedule illustrated in Figure 5, we have similar conclusion which is summarized in Theorem 4.2.

Theorem 4.2 *Given a two-speed schedule as shown in Figure 5, and letting $S_2 > S_1$, if for any $S_2 > S_1$, we have $G_2 > G_1$ and $B_k > 0$ (with G_k, B_k defined in Eq. (11) and (9), respectively), then the temperature at $t = 1$, i.e., T_e, is a monotonically decreasing function of x.*

Proof. Based on Eq. (10), let T_i and T_a be the temperature at points i and a, respectively. For simplicity let the starting temperature be zero, we have

$$\begin{aligned} T_i &= G_2(1 - e^{-\mathrm{B}_2 x}) \\ T_a &= G_1 + (T_i - G_1)e^{-\mathrm{B}_1 t_2} \end{aligned} \tag{19}$$

the ending temperature is given by:

$$T_e = G_2 + (T_a - G_2)e^{-\mathrm{B}_2(t_1 - x)} \tag{20}$$

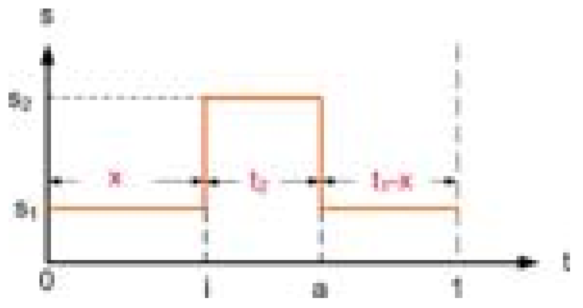

Fig. 4. A two-speed schedule that uses speed s_1 for t_1 time units and speed s_2 for t_2 time units. $t_1 + t_2 = 1$.

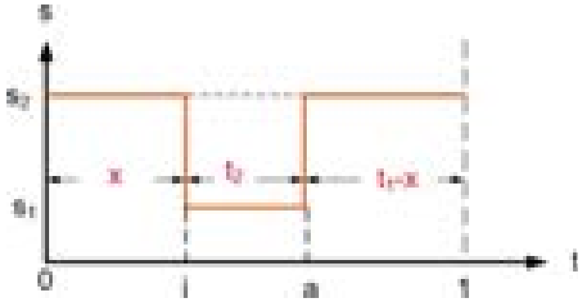

Fig. 5. A two-speed schedule that uses speed S_2 for t_1 time units and speed s_1 for t_2 time units. $t_1 + t_2 = 1$.

Again, replacing T_a with Eq. (19) and simplifying we have,

$$T_e = G_2(1-e^{-(\mathrm{B_2t_1+B_1t_2})}) - (G_2-G_1)(1-e^{-\mathrm{B_1t_2}})e^{-\mathrm{B_2(t_1-x)}} \tag{21}$$

Therefore,

$$\begin{aligned}\frac{d(T_e)}{dx} &= \frac{d}{dx}(G_2(1-e^{-(\mathrm{B_2t_1+B_1t_2})}) \\ &\quad -(G_2-G_1)(1-e^{-\mathrm{B_1t_2}})e^{-\mathrm{B_2(t_1-x)}}) \\ &= -(G_2-G_1)(1-e^{-\mathrm{B_1t_2}})B_2e^{-\mathrm{B_2(t_1-x)}}\end{aligned} \tag{22}$$

if $G_2 > G_1$ and $B_k > 0$, we have $d(T_e)/dx$ always less than zero and thus T_e monotonically decreases with x. □

Theorems 4.1 and 4.2 indicate that the temperature at the end of a schedule depends on the locations where different running modes are applied. They help to reduce the temperature at the end of a schedule, but do not necessarily reduce the maximum temperature within the entire interval. In addition, Theorems 4.1 and 4.2 are applied for a single job rather than a periodic task set. In what follows, we introduce a novel scheduling algorithm, i.e., the *M-Oscillating* algorithm, to minimize the peak temperature for a periodic hard real-time task. We assume that when a processor runs a periodic task, the temperature will not run away and eventually reach a steady state. The temperature steady state is defined as follows.

Definition 4.3 *When running a periodic task with period p, the temperature of a processor is called to be stable if for a given threshold, i.e.,* $0 < \varepsilon \ll 1$,

$$|T((n+1)p) - T(np)| < \varepsilon, \tag{23}$$

where $n \geq 0, n \in Z$, *and* $T(t)$ *is the temperature at* t.

Our *M-Oscillating* algorithm works as follows: given a two-speed schedule, we divide the high speed interval and the low speed interval evenly into m sections and run the processor with the low speed and high speed alternatively. Apparently, an *M-Oscillating* schedule will complete the same workload as the original two-speed schedule in one period and thus guarantee the deadline. At the same time, the maximum temperature can be significantly reduced as stated in the following theorem.

Theorem 4.4 *Let* $S(t)$ *be a two-speed schedule and* $\tilde{S}(m,t)$ *be the corresponding M-Oscillating schedule. Also let* $T_{\max}(S)$ *represent the maximum temperature that a processor can reach when running schedule S. If for any* $v_2 > v_1$, *we have* $G_2 > G_1$ *and* $B_i > 0, i = 1, 2$, *then*
- $T_{\max}(\tilde{S}(m,t)) \leq T_{\max}(S(t))$,
- $T_{\max}(\tilde{S}(n,t)) \leq T_{\max}(\tilde{S}(m,t))$ *if* $m \leq n$.

Proof. From the conclusion presented in Theorem 3.3, we know that when the processor reaches the thermal steady state, the peak temperature achieved by any two-speed schedule is the same. Therefore, to prove this theorem, we consider a step-up schedule i.e., the case shown in Figure 6.

For $\tilde{S}(m,t)$ shown in Figure 6, base on Eq. (10), the temperature at $t = x$ and $t = y$ can be formulated as

$$T_x = G_1(1-e^{-\mathrm{B_1t_1/m}}), \quad T_y = G_2 + (T_x - G_2)e^{-\mathrm{B_2t_2/m}}$$

From 29, when the temperature reaches the thermal steady state, we have

$$T_{max}(\tilde{S}(m,t)) = T_y^\infty = T_y + \frac{T_y}{1-K_y}K_y$$

where

$$K_y = e^{-((B_1t_1+B_2t_2))/m}$$

Expand T_y^∞, we have

$$T_y^\infty = (G_2-G_1)\frac{1-e^{-(B_2t_2)/m}}{1-e^{-(B_1t_1+B_2t_2)/m}} + G_1$$

Let $B_2t_2 = m(m+1)p$ and $B_1t_1 = m(m+1)q$, $p, q > 0$ and let

$$f(m) = \frac{1-e^{-mp}}{1-e^{-m(p+q)}}$$

Then,

$$T_{\max}(\tilde{S}(m,t)) = (G_2-G_1)f(m+1) + G_1$$
$$T_{\max}(\tilde{S}(m+1,t)) = (G_2-G_1)f(m) + G_1$$

To show that $f(m+1) > f(m)$, we only need to note that

$$f(m) = \frac{1-e^{-p}}{1-e^{-(p+q)}} \cdot \frac{\sum_{i=0}^{m-1} e^{-ip}}{\sum_{i=0}^{m-1} e^{-i(p+q)}}$$

Also,

$$\frac{\sum_{i=0}^{m-1} e^{-ip}}{\quad} < \frac{\sum_{i=0}^{m} e^{-ip}}{\quad}$$

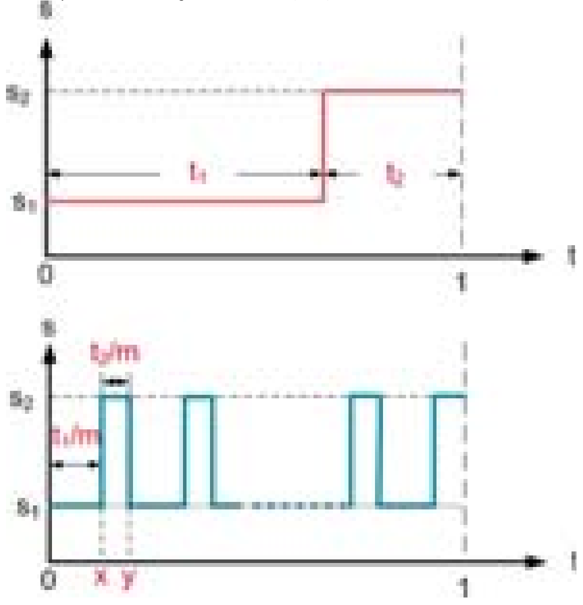

Fig. 6. A two-speed schedule and its corresponding m-oscillation schedule.

Since $i \leq m$, $e^{(m-i)q} \geq 1$, we have $f(m+1) > f(m)$, and therefore

$$T_{\max}(\tilde{S}(m,t)) > T_{\max}(\tilde{S}(m+1,t)) \quad \square \tag{24}$$

Theorem 4.4 implies that, by dividing the high speed interval and the low speed interval each into m equal sections and running them alternatively, an *M-Oscillating* schedule can always reduce the maximum temperature when a processor reaches its thermal steady state. The larger the m is, the lower the maximum temperature becomes.

Note that the conclusion in Theorem 4.4 and its proof are contingent upon two important assumptions, i.e., (*i*) $G_2 > G_1$ for any $v_2 > v_1$ and (*ii*) $B_i > 0, i = 1, 2$. It is difficult, however, to analytically validate these two assumptions since the temperature invariants C_0 and C_1 in Eqs. (9) and (11) depend on the technology parameters. In addition, C_0 and C_1 are obtained through curve-fitting rather than from a closed analytical formula. In Section 6, we validate these assumptions empirically. Moreover, it is worth mentioning that Theorem 4.4 ignores the voltage transition overhead which can be very significant in certain scenarios. When the overhead is non-negligible, conceivably, there exists an optimal value of m to balance the impact of the transition overhead and the potential of *M-Oscillating* algorithm in peak temperature reduction. How to identify this optimal value by incorporating transitional model such as that proposed in Ref. [8] is an interesting problem and will be discussed in the following section.

5. THE *M-OSCILLATING* CONSIDERING TRANSITION OVERHEAD

In this section, we introduce how to incorporate the non-negligible transition overhead into the *M-Oscillating* technique. In the real world scenario, when the processor speed switches, there is a short time interval (on the order of hundreds of microseconds[8]) during which the clock is halted. As a result, the amount of time the processor spends on useful computation is reduced. To compensate the performance loss during clock halting, one intuitive way is to increase the high or low processor speeds. However, this is not always feasible (e.g., when the high speed is the maximum processor speed). Furthermore, it increases the dynamic energy consumption and makes the tradeoff analysis more complicated. In our case, we simply change the duration of high and low speed subject to the workload constraint, i.e., extend the high speed interval and reduce the low speed interval. This sets an upper bound of m, beyond which the predefined workload cannot be completed.

We assume that the clock will be halted for a short interval τ during each speed transition. In contrast to the ideal two-speed schedule shown in Figure 6, the non-ideal two-speed schedule is plotted in Figure 7(A). Apparently, in the non-ideal scenario, a performance loss of $(S_1 + S_2)\cdot\tau$ is imposed due to *one* speed transition. Thus, in order to counteract this performance loss, one have to extend the high speed interval while reduce the low speed interval by the same time frame δ as shown in Figure 7(B). The amount of time δ taken from low speed can be calculated as

$$\delta = \frac{(S_1 + S_2)\cdot\tau}{S_2 - S_1} \tag{25}$$

From Eq. (25), one can easily notice that δ is a positive number. Therefore, the number of transition m cannot be arbitrarily increased since the low speed duration t_1 in the ideal two speed schedule has to be sufficiently large to accommodate m speed transitions. Then, the maximum allowable m can be calculated by letting $m\cdot(\delta+\tau) \leq t_1$. Thus,

$$m \leq \frac{t_1}{\delta+\tau}$$

Evidently, the term $t_1/(\delta+\tau)$ is not necessarily an integer. Therefore, in order to ensure the workload constraint, we set the upper bound of m as

$$m_{\text{Max}} = \lfloor \frac{t_1}{\delta+\tau} \rfloor$$

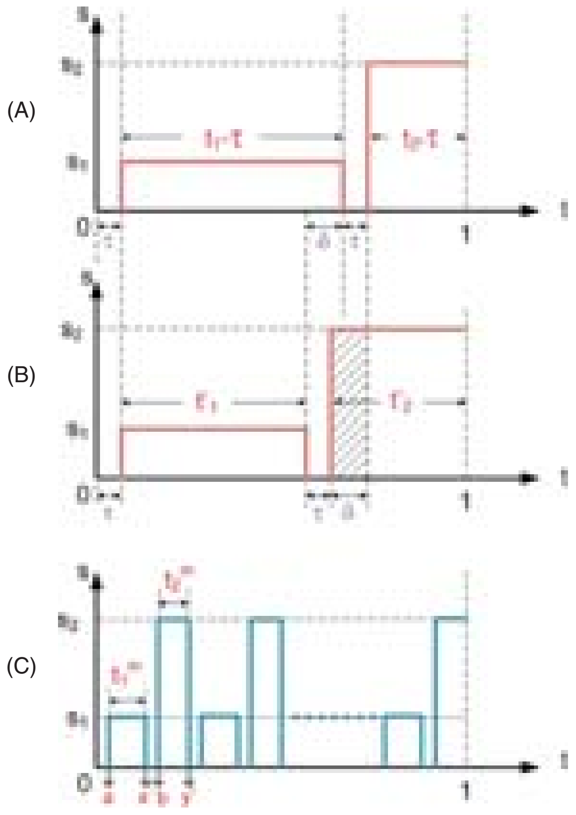

Fig. 7. Non-ideal two-speed schedule and its corresponding *M-Oscillating* schedule.

Based on Eq. (25), the modified non-ideal two-speed schedule can be found and the reduced low speed interval t_1' and extended high speed interval t_2' are calculated by

$$t_1' = t_1 - \tau - \delta$$
$$t_2' = t_2 - \tau + \delta$$

By adopting the similar concept, we can find the corresponding non-ideal *M-Oscillating* schedule for a given ideal two-speed schedule. The procedure is shown the Lemma 5.1.

Lemma 5.1 *Given an ideal two speed schedule with low speed S_1 and high speed S_2 running for t_1 and t_2, respectively, for any value of m from 1 to m_{Max}, the corresponding non-ideal M-Oscillating schedule can guarantee the same workload if the adjusted low speed sub-interval t_1^m and high speed sub-interval t_2^m are calculated by Eqs. (26) and (27).*

$$t_1^m = \frac{t_1}{m} - \tau - \delta \tag{26}$$

$$t_2^m = \frac{t_2}{m} - \tau + \delta \tag{27}$$

Now the problem becomes how to identify the optimal value of m that leads to the lowest peak temperature. Consider the example illustrated in Figure 7(C). Based on Ref. [29], when the temperature reaches the thermal steady state, the maximal temperature of a non-ideal *M-Oscillating* schedule $T_{\max}^{\text{NI}}(\tilde{S}(m,t))$ can be expressed as

$$T_{\max}^{\text{NI}}(\tilde{S}(m,t)) = T_{y'}^{\infty} = T_{y'} + \frac{T_{y'}}{1-K_{y'}}K_{y'} \tag{28}$$

where $T_{y'}$ can be expressed as a function of m based on Eq. (10) and can be calculated iteratively once the temperature of previous speed transition points, i.e., T_a, $T_{x'}$ and T_b, are available. Specifically,

$$\begin{aligned} T_{y'} &= G_2 + (T_b - G_2)e^{-B_2 t_2^m} \\ T_b &= G_0 + (T_{x'} - G_0)e^{-B_0\tau} \\ T_{x'} &= G_1 + (T_a - G_1)e^{-B_1 t_1^m} \\ T_a &= G_0 + (1 - e^{-B_0\tau}) \end{aligned} \tag{29}$$

where t_1^m and t_2^m are obtained from Eqs. (26) and (27). And $K_{y'}$ can be computed by

$$K_{y'} = e^{-(\mathrm{B_1 t_1^m + B_2 t_2^m + 2B_0\delta})} \tag{30}$$

where δ is defined in Eq. (25).

Once the formula of the maximal temperature is available, one straightforward way to find optimum m is to set the first order derivative of Eq. (28) equal to zero and solve for m. However, this method can be challenging not only because the complexity of the equation, but also for the discrete nature of Eq. (28), since the solution of the optimum m is not necessarily an integer. Instead, we adopt a searching algorithm[28] that can locate optimum m in linear time based on Eq. (28).

The algorithm is summarized in Algorithm 1. Initially, the *interval of uncertainty* is set between 1 and the upper bound of m. We choose two points, i.e., Lb and Ub in step 4 and 5, respectively, that equally divide the initial interval to evaluate the function $T_{\max}^{\text{NI}}(\tilde{S}(Lb,t))$ such that after each iteration the *interval of uncertainty* can be reduced by at least one third. Thus, new *interval of uncertainty* bound L and R are updated and new evaluation points are calculated during each iteration. The loop will stop until the *interval of uncertainty* is small enough to locate single integer value which is the optimum value of m.

Algorithm 1 Searching for optimum m

```
1: Input: the upper bound of m: $m_{Max}$;
2: Initialize uncertainty interval: $L = 1$, $R = m_{Max}$;
3: while (1) do
4:     $Lb = \lfloor L + 1/3 \times (R - L) \rfloor$;
5:     $Ub = \lfloor L + 2/3 \times (R - L) \rfloor$;
6:     if $T_{max}^{NI}(\tilde{S}(Lb,t)) < T_{max}^{NI}(\tilde{S}(Ub,t))$; then
7:         $L = L$;
8:         $R = Ub$;
9:     else if $T_{max}^{NI}(\tilde{S}(Lb,t)) < T_{max}^{NI}(\tilde{S}(Ub,t))$; then
10:        $L = Lb$;
11:        $R = R$;
12:    else if $T_{max}^{NI}(\tilde{S}(Lb,t)) == T_{max}^{NI}(\tilde{S}(Ub,t))$; then
13:        $L = L$;
14:        $R = Ub$;
15:    end if
16:    if $R - L \le 2$ then
17:        Break;
18:    end if
19: end while
```

6. EMPIRICAL STUDIES

In this section, we use experiments to test the effectiveness of the proposed *M-Oscillating* technique. Two platforms were developed for this purpose. The first one was implemented in C/Matlab based on a chip-level power/thermal model we introduced in Section 3 while the second one was developed based on a real desktop PC platform.

6.1. Experimental Results from Chip Level Simulator

In this subsection, we discuss several experiments that we have performed on a simulation environment that uses system models discussed in Section 3. First, we validated the processor model as well as the assumptions upon which the algorithms are developed. We then evaluated the performance of both ideal and non-ideal *M-Oscillating* by comparing it with an existing work, i.e., the two-speed scheduling method 36, in terms of the feasibility and peak temperature.

6.1.1. Verification of the Processor Model and Assumptions

To verify the processor model and assumptions made in Theorems 4.1 to 4.4, we built our processor models based on the work by Ref. [23] using the 65 nm technology from the UC Berkeley's BSIM device model. Specifically, we used Eq. (1) to compute the leakage currents for temperature from 40 °C to 110 °C with a step size of 5 °C, and supply voltage from 0.60 Volt to 1.30 Volt with a step size of 0.05 V. These results were used to determine the temperature invariants $C_0(k)$ and $C_1(k)$ in Eq. (5) through curve-fitting.

We set the frequency for each mode according to the formula[23]

$$f = \frac{1}{\text{delay}} = \frac{(v - v_t)^{\mu}}{vT^{\eta}} \times 4.2824 \times 10^{14} \qquad (31)$$

with $\mu = 1.19$, $\eta = 1.2$, $v_t = 0.3$, and T was set to the highest temperature as 100 °C, and we also normalized the frequency with the highest equal to 1.0. To obtain the leakage power consumption, we set N_{gate} in Eq. (4) to be 10^6. The dynamic power consumption (and thus constant C_2) was determined based on experimental results reported in Ref. [23] on a common benchmark *gcc*. For thermal constants, we considered two different options, the first one is the conventional air cooling with $R_{\text{th}} = 0.8$ K/W, $C_{\text{th}} = 340$ J/K[34] and the second one is the water spray-cooling with $R_{\text{th}} = 0.067$ K/W, $C_{\text{th}} = 340$ J/K.[33] The ambient temperature was set to 25 °C.

Figure 8 compares the estimated leakage power consumptions by using two different leakage/dependency models (namely, *Models 1 and 2*) with the "*actual leakage*," which is calculated based on Eq. (1). *Model 1* is the leakage/temperature model used in this paper that assumes the leakage varies with both temperature and supply voltage. *Model 2*, used in Refs. [8, 12, 14], assumes that the leakage changes with the temperature linearly but not with the supply voltage. As we can see from Figure 8, the linear approximated leakage power consumptions obtained by *Model 1* match very closely to that calculated based on Eq. (1), with the maximum relative error no more than 5.5%. On the other hand, when using *Model 2*, the leakage approximation errors can be very significant: the actual leakage power consumption can be as high as 4.5 times or as low as 29% of the estimated results, depends on the applied supply voltage level.[18]

We have also examined the assumptions made in Theorems 4.1 to 4.4. Figure 9 and Figure 10 plot the characteristics of function G_k and B_k under different supply

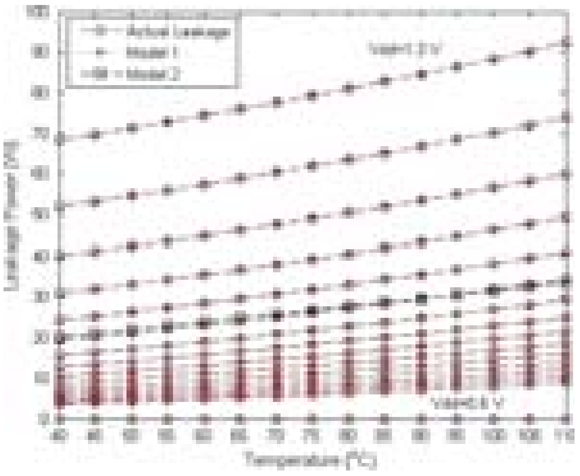

Fig. 8. Linear approximation of leakage power consumptions.

voltages and thermal constants. As illustrated in these two figures, we can clearly see that, under both representative cooling options (i.e., $R_{\text{th}} = 0.8$ J/°C and $R_{\text{th}} = 0.067$ J/°C), function G_k is a positive and monotonically increasing function of the supply voltage, and function B_k is also a positive function for the given settings. These results validate assumptions made in Theorems 4.1 to 4.4.

6.1.2. Performance Evaluation

We next study the performance of *M-Oscillating* schedule by comparing with the existing approach with and without transition overhead consideration. The proactive scheduling method introduced in Ref. 12 intends to minimize the task response time under the given maximum temperature constraint. However, it is developed based on a processor model with continuously changeable speed, and to extend the proposed scheduling technique to a more practical processor model (i.e., with discrete supply voltages) as we used in this paper is far from a trivial and straightforward effort. Therefore, we compare our approach with a more general one, i.e., the reactive two-speed scheduling approach introduced in Ref. [36]. The reactive two-speed schedule[36] works as follows. For a given maximum temperature constraint, the processor works at the highest speed until it reaches the maximum temperature. Then, it runs at an equilibrium speed to maintain the temperature.

First, we want to investigate the feasibility of the two scheduling policies, i.e., the *M-Oscillating* schedule and the reactive two-speed schedule, without transition-overhead being considered, under the same maximum temperature constraint and workload. Note that for a given maximum temperature and a processor with discrete speeds, the equilibrium speed is not necessarily one of the available speeds. We therefore fixed the equilibrium speed to one of the available speeds of the processor, and then used the stable temperature as the maximum temperature

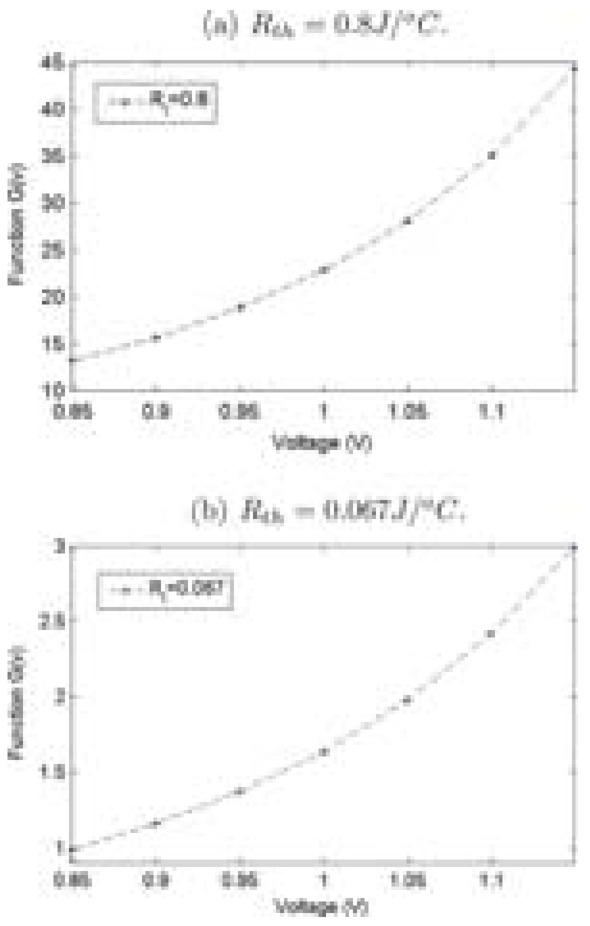

Fig. 9. Function $G(v)$ with different temperatures and thermal resistances.

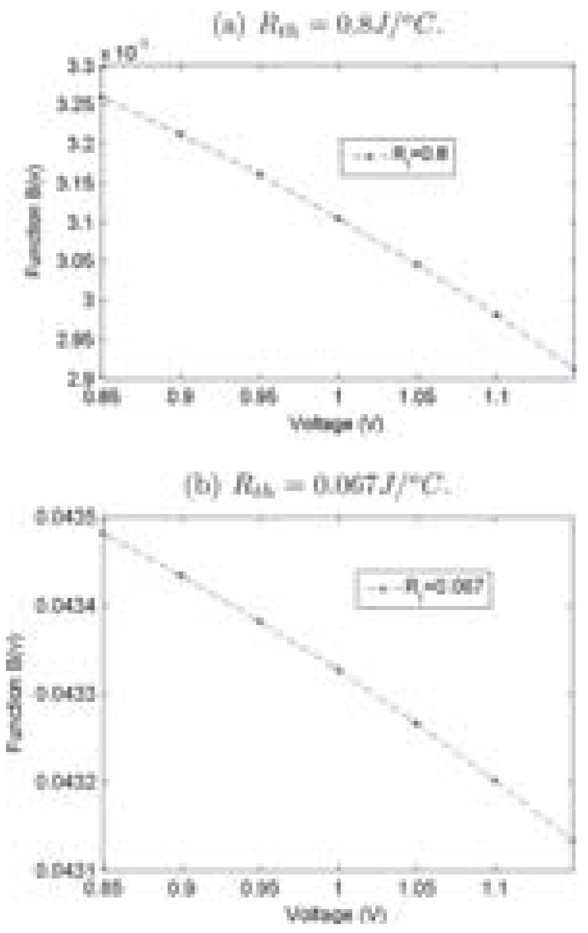

Fig. 10. Function $B(v)$ with different temperatures and thermal resistances.

constraint to test both scheduling policies. This experimental setting is clearly favorable to the reactive two-speed scheduling approach.

We randomly generated real-time tasks with period of 2000 seconds and workload evenly distributed within the range of [0, 100%], with 100% indicating that the processor has to run at the maximum speed all the time (i.e., 100%) to complete the workload. We divided the task workload into 10 equal intervals, i.e., 0–10%, 10–20% and so on, and 100 random tasks were generated within each interval. We assume that the processor has five active modes i.e., from 0.9 V to 1.3 V, with step size of 0.1 V and one shut-down mode. The equilibrium voltages were set to be 0.9 V and 1.0 V, and the corresponding maximum temperature, calculated using Eq. (7), were set as the maximum temperature constraint. Table I lists the values of the equilibrium voltages and their corresponding maximum temperatures. For *M-Oscillating* schedule, we first calculated the constant speed that can guarantee the workload. Then, the two neighboring speeds were used to construct our *M-Oscillating* schedule algorithm described in Section 4.

Figure 11 presents the feasibility differences between reactive two-speed scheduling and *M-Oscillating* schedule with $m = 1$, 2, 5, 10 and 15. When the randomly generated workload is very low, all schedules are feasible; and when the workload is high, none of the can successfully schedule the task. Therefore Figure 11 only depicts the workload regions where differences exist in terms of feasibility among different scheduling choices. From Figure 11, we can see that *M-Oscillating* schedule shows higher feasibility as compared to reactive two-speed schedule. The larger the m is, the higher the feasibility can be.

Table I. The equilibrium speeds and the corresponding maximum temperatures.

$V_{\text{Equil}}(V)$	$T_{\max}$(°C)
0.90	41.9
1.0	49.3

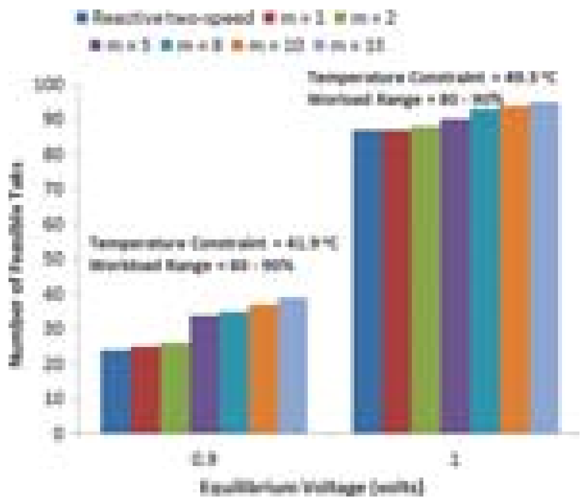

Fig. 11. Feasibility comparison between the *M-Oscillating* scheme and the reactive two-speed scheme under different maximum temperature constraints.

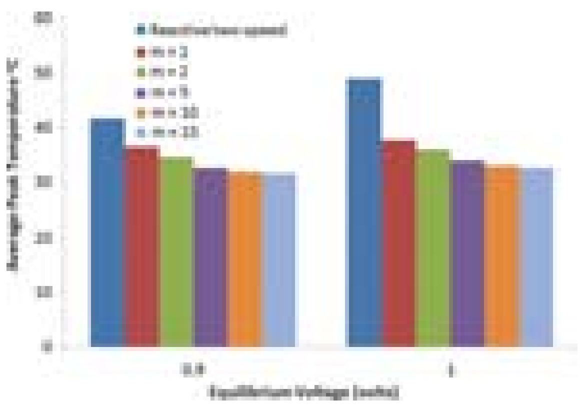

Fig. 12. Average maximum temperature comparison between the *M-Oscillating* scheme and the reactive two-speed scheme.

At the equilibrium voltage of 0.9 V, the feasibility achieved by the reactive two-speed scheduling policy is very close to the *M-Oscillating* scheduling algorithm. However, when m is increased to 15, the feasibility improves, (e.g., 25% when $m = 1$ and 39% when $m = 15$). At the equilibrium voltage 1.0 V, we can observe the feasibility improvement of about 10%, by *M-Oscillating* schedule algorithm to the two-speed scheduling algorithm.

Even though a task can be feasibly scheduled, a higher peak temperature is not desirable. We therefore collected the maximum temperatures of all feasible tasks under different scheduling policies and compared their averaged maximum temperatures as shown in Figure 12. Figure 12 demonstrates that the *M-Oscillating* schedule algorithm is very effective in reducing the peak temperature. Note that at the equilibrium voltage of 0.9 V, the average maximum temperature of reactive two-speed schedule is 41.7 °C, and is reduced to 37.6 °C when $m = 1$ for the *M-Oscillating* scheduling algorithm. It is further reduced to 31.5 °C for $m = 15$. At equilibrium voltage 1.0 V, the average maximum temperatures is reduced by 11.3 °C.

We next performed the same performance evaluation experiments as discussed above, but with transition overhead being considered. We assume that the transition overhead are 0.01, 0.1 and 1.0 second, respectively. All the other settings were similar to the previous experiments. Figure 13 presents the feasibility results for the reactive two-speed and the *M-Oscillating* schedule with $m = 1$, 2, 5, 10 and 15, for different transition overheads. In Figure 13, when the overhead time = 0.01 s, the feasibility results are almost similar to the results that we obtained without considering transition overhead. When the overhead time = 0.1 s, the feasibility for every schedule decreases, but the trend of feasibility remains the same i.e. the larger the m, the higher the feasibility is. However, when the overhead time = 1.0 s, the *M-Oscillating* schedule fails to follow the trend. The feasibility increases from $m = 1$ to 5, but for higher values, it start to decrease. For instance, when $m = 1$ the feasibility is 24% and when $m = 5$ the feasibility increases to 26%. But as we further increase the value of m, the feasibility starts to decrease due to the excessive transition overhead. To further verify this behavior, the equilibrium voltage and temperature constraint were set to 1.0 V and 49.3 °C, respectively. In Figure 14, we can see that the feasibility trend is the same as observed in Figure 13. The feasibility increases up-to 84% for $m = 2$ and it decreases to 54% when $m = 15$.

We also collected the maximum temperatures of all feasible tasks under different scheduling policies, by assuming that the transition overhead time is 1 s. From Figure 15, we observe that even after considering the transition overhead, the *M-Oscillating* algorithm is very effective in reducing the peak temperature. At the equilibrium voltage of 0.9 V, the average maximum temperature of reactive two-speed schedule is 41.7 °C, and is reduced to 37.8 °C

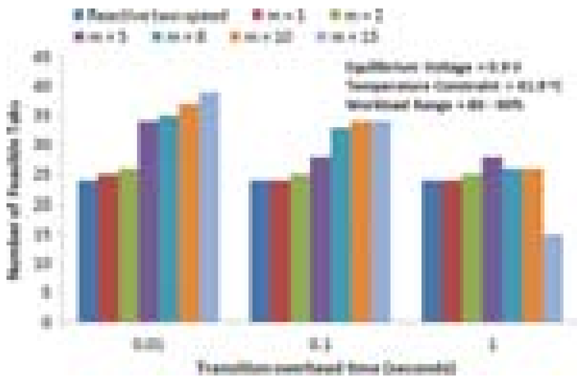

Fig. 13. Feasibility comparison between the *M-Oscillating* scheme and the reactive two-speed scheme under different Transition-overhead at Maximal temperature constraint 41.9 °C.

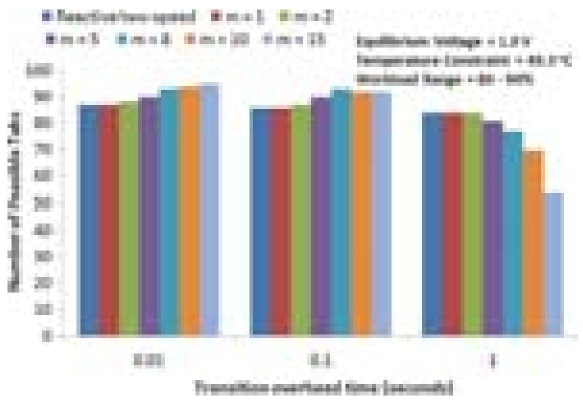
Fig. 14. Feasibility comparison between the *M-Oscillating* scheme and the reactive two-speed scheme under different Transition-overhead at Maximal temperature constraint 49.3 °C.

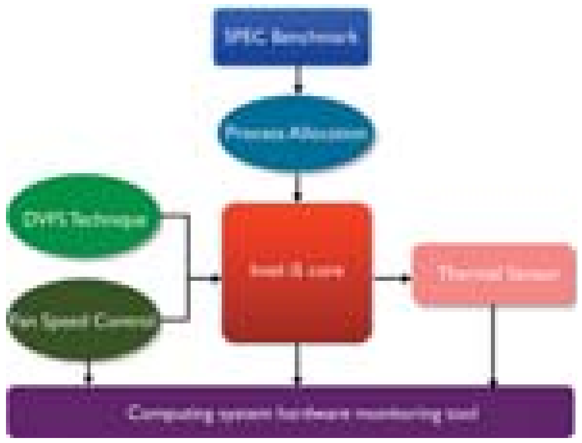

Fig. 16. Overall structure of hardware platform.

when $m = 1$ for the *M-Oscillating* scheduling algorithm. It is further reduced to 31.6 °C for $m = 15$. At equilibrium voltage 1.0 V, the average maximum temperatures is reduced by 11 °C.

6.2. Experimental Results from Real Hardware Platform

To study the effectiveness of the *M-Oscillating* scheduling algorithm, we tested it under a more practical scenario. In this subsection, we first introduce the setup of our hardware platform. We then compare the *M-Oscillating* scheduling algorithm with the reactive two-speed schedule on this platform.

6.2.1. Testbed

Our hardware platform is built based on a Dell Precision T1500 Desktop workstation with an Intel i5 750 quad core microprocessor running Linux operating system with kernel version of 2.6.23. The overall structure of our platform is shown in Figure 16. The i5 microprocessor is integrated with the DVFS capability and can adjust the working frequency of each individual core. The fan speed is set to a fixed value to guarantee the cooling constant unchanged during the experiments. The processor temperature information were collected from the on-chip *Digital Thermal Sensor* (DTS). A system hardware monitoring tool *Lm-sensors* has been used to monitor the frequency, the fan speed, and the die temperature during run-time.

To implement the DVFS, we adopted the *CPUfreq* kernel package which is a Linux kernel subsystem that provides an interface between the user level frequency-controlling policy and the underlying frequency-controlling mechanisms. The DVFS transition overhead is approximately 20 μs which is acceptable for most computing systems.[37]

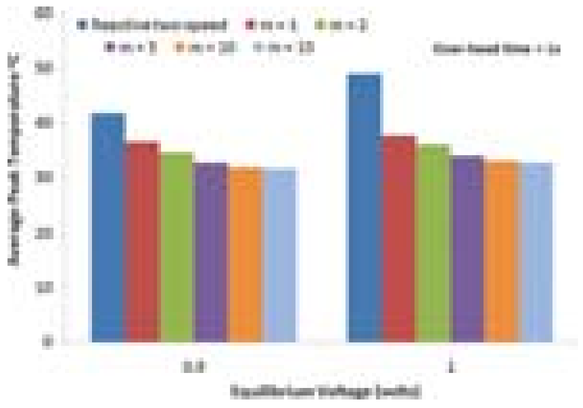
Fig. 15. Average maximum temperature comparison between the *M-Oscillating* scheme and the reactive two-speed scheme including Transition-overhead.

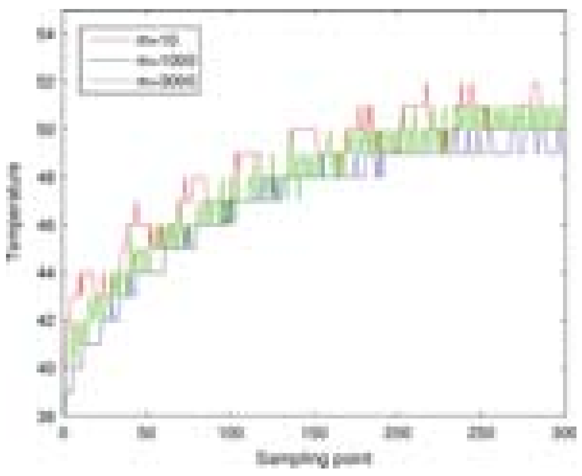
Fig. 17. Different temperature curves of *M-Oscillating* with different *m*.

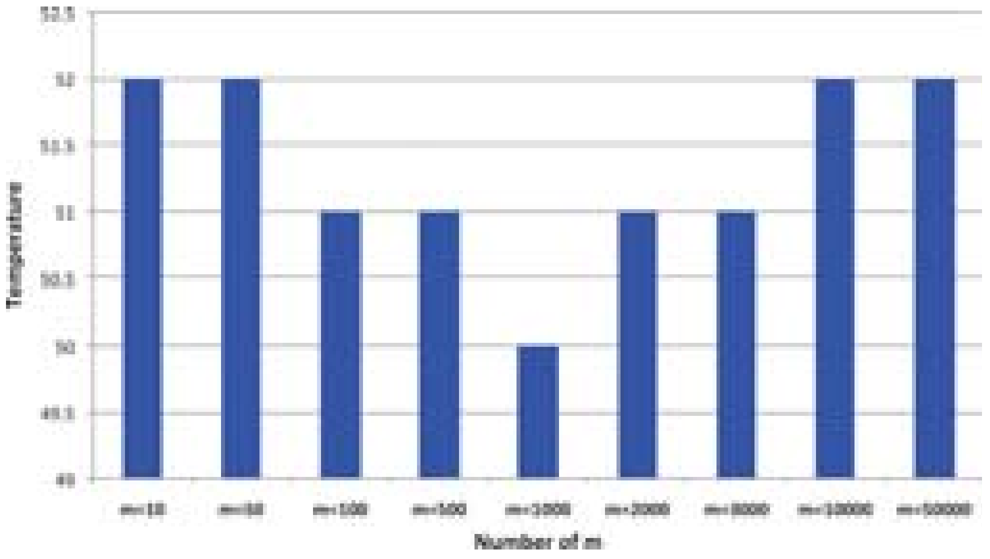

Fig. 18. Peak temperature of the *M-Oscillating* with different number of *m*.

The run-time temperature information is obtained based on the on-chip digital thermal sensors which store the temperature information in the *Model Specific Register* (MSR). The resolution of our on-chip thermal sensor is 1 °C, and the time for operating system to reflect a change of 1 °C is around 1 second and thus our temperature sampling period is set to 1 second.

6.2.2. Performance Evaluation

We first want to verify that when the transition overhead is not negligible, there exists a minimal peak temperature of the *M-Oscillating* schedule when different *m* is applied (i.e., the non-ideal scenario). We used the SPEC CPU2000 benchmark suite in our experiments to get the credible and comparable experiment results. We selected benchmarks *gzip*, *vpr*, *gcc*, *mcf*, *bzip2* from integer operation and *win*, *mgrid*, *applu*, *mesa* and *galgel* from floating operation. Each set of experiment is executed at the same initial temperature. The experimental results from different benchmark programs are very close in terms of the relationship between peak temperature and the number of speed transition *m*. Therefore, we only show the experiment result from benchmark *galgel*. We can see from Figure 18 that initially when *m* is small, the peak temperature drops as we increase the *m*. Then, as we continue to increase *m*, the peak temperature starts to rise which conforms to our theoretical conclusion.

We next studied if the *M-Oscillating* schedule can outperform the existing reactive two-speed schedule[36] in the practical test platform we developed. We assume the real-time task is to run benchmark *galgel* alone. The execution time was obtained by running *galgel* at the highest speed. To study the algorithm performance under different workload densities, we set the ratio of execution time and deadline to be varied from 50% to 100%. The peak temperature limit is set to 57 °C which is the steady state temperature of running *galgel* at 2.5 GHz. Table II lists the experimental results by the *M-Oscillating* algorithm and the reactive two-speed scheduling method. Note that the *M-Oscillating* algorithm can always meet the deadline. It fails when the temperature constraint is violated. On the other hand, the reactive two-speed schedule can always meet the temperature constraints. It fails when it cannot complete the workload by the deadline.

From Table II, we can see that the reactive two-speed schedule reaches and maintains the peak temperature under each test case. However, it fails 3 in 11 test cases. On the other hand, the *M-Oscillating* algorithm fails only one time among the total 11 test cases. The only one scenario that the *M-Oscillating* fails to guarantee the peak temperature constraint is when the workload density equals 100% which leaves no slack for us to apply the *M-Oscillating* approach. At the same time, we can see that *M-Oscillating*

Table II. Feasibility comparison of two schedules.

		Reactive two-speed		*M-Oscillating*	
Density	Deadline	Feasible?	Peak temp.	Feasible?	Peak temp.
100%	301.779s	N	57 °C	N	59 °C
95%	317.662s	N	57 °C	Y	57 °C
90%	335.310s	N	57 °C	Y	56 °C
85%	355.034s	Y	57 °C	Y	54 °C
80%	377.224s	Y	57 °C	Y	52 °C
75%	402.372s	Y	57 °C	Y	50 °C
70%	431.113s	Y	57 °C	Y	49 °C
65%	464.275s	Y	57 °C	Y	47 °C
60%	502.965s	Y	57 °C	Y	46 °C
55%	548.689s	Y	57 °C	Y	45 °C
50%	603.558s	Y	57 °C	Y	44 °C

algorithm can feasibly schedule the "task" with much lower peak temperature, as much as 13 °C lower than the reactive two-speed. Table II demonstrates clearly that *M-Oscillating* algorithm can outperform the reactive two-speed schedule.

7. CONCLUSIONS

With the continuous scaling of IC technology, the interdependency of temperature and leakage exacerbates not only the power/energy minimization problem but also the thermal management problem. In this paper, we incorporate the leakage/temperature dependency and non-negligible transition overhead into the real-time scheduling analysis that aims at minimizing the peak temperature. We develop a new scheduling technique, i.e., the *M-Oscillating*, that can effectively reduce the peak temperature when executing a hard real-time periodic task set. We validate the effectiveness of the proposed algorithm by comparing with an existing approach on two platforms, i.e., a software simulation platform and a more practical desktop platform. The experimental results obtained from both platforms demonstrate that the proposed *M-Oscillating* scheme can greatly outperform the existing approaches in terms of peak temperature reduction and feasibility improvement.

References

1. Spec2000 benchmarks, http://www.spec.org.
2. Hotspot 4.2 temperature modeling tool, University of Virgina, page http://lava.cs.virginia.edu/HotSpot (**2009**).
3. M. Ahmed, N. Fisher, S. Wang, and P. Hettiarachchi, Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources. *Sustainable Computing: Informatics and Systems (SCIS)* 1, 226 (**2011**).
4. N. Bansal, T. Kimbrel, and K. Pruhs, Speed scaling to manage energy and temperature. *Journal of the ACM* 54, 1 (**2007**).
5. M. Bao, A. Andrei, P. Eles, and Z. Peng, On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration, DAC (**2009**), pp. 490–495.
6. M. Bao, A. Andrei, P. Eles, and Z. Peng, Temperature-aware idle time distribution for energy optimization with dynamic voltage scaling, DATE (**2010**), pp. 21–27.
7. T. Chantem, R. P. Dick, and X. S. Hu, Temperature-aware scheduling and assignment for hard real-time applications on mpsocs, DATE (**2008**), pp. 288–293.
8. T. Chantem, X. S. Hu, and R. Dick, Online work maximization under a peak temperature constraint, ISLPED (**2009**), pp. 105–110.
9. V. Chaturvedi, H. Huang, and G. Quan, Leakage aware scheduling on maximal temperature minimization for periodic hard real-time systems, ICESS (**2010**), pp 1802–1809.
10. J. Chen, C. Hung, and T. Kuo, On the minimization fo the instantaneous temperature for periodic real-time tasks, RTAS (**2007**), pp. 236–248.
11. J.-J. Chen, H.-R. Hsu, and T.-W. Kuo, Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems, RTAS (**2006**), pp. 408–417.
12. J.-J. Chen, S. Wang, and L. Thiele, Proactive speed scheduling for real-time tasks under thermal constraints, RTAS (**2009**), Vol. 0, pp. 141–150.
13. A. Cohen, F. Finkelstein, A. Mendelson, R. Ronen, and D. Rudoy, On estimating optimal performance of cpu dynamic thermal management. *IEEE Computer Architecture Letter* 2, 6 (**2003**).
14. N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, Thermal-aware global real-time scheduling on multicore systems, RTAS (**2009**), pp. 131–140.
15. V. Hanumaiah, R. Rao, S. Vrudhula, and K. S. Chatha, Throughput optimal task allocation under thermal constraints for multi-core processors, *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, ACM, New York, NY, USA (**2009**), pp. 776–781.
16. V. Hanumaiah, S. Vrudhula, and K. Chatha, Maximizing performance of thermally constrained multi-core processors by dynamic voltage and frequency control, *Computer-Aided Design—Digest of Technical Papers, IEEE/ACM International Conference on*, November (**2009**), pp. 310–313.
17. H. Huang and G. Quan, Leakage aware energy minimization for real-time systems under the maximum temperature constraint, DATE (**2011**), pp. 1–6.
18. H. Huang, G. Quan, and J. Fan, Leakage temperature dependency modeling in system level analysis, ISQED (**2010**), pp. 447–452.
19. H. Huang, G. Quan, J. Fan, and M. Qiu, Throughput maximization for periodic real-time systems under the maximal temperature constraint, DAC (**2011**), pp. 363–368.
20. R. Jayaseelan and T. Mitra, Temperature aware task sequencing and voltage scaling, ICCAD (**2008**), pp. 618–623.
21. R. Jejurikar, C. Pereira, and R. Gupta, Dynamic slack reclamation with procrastination scheduling in real-time embedded systems, DAC (**2005**), pp. 111–116.
22. P. Kumar and L. Thiele, Thermally optimal stop-go scheduling of task graphs with real-time constraints, ASP-DAC (**2011**), pp. 123–128.
23. W. Liao, L. He, and K. Lepak, Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 1042 (**2005**).
24. S. Liu and M. Qiu, Thermal aware scheduling for peak temperature reduction with stochastic workloads, RTAS(WiP) (**2010**), pp. 53–56.
25. Y. Liu, R. P. Dick, L. Shang, and H. Yang, Accurate temperature-dependent integrated circuit leakage power estimation is easy, DATE (**2007**), pp. 1526–1531.
26. Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang, Thermal vs energy optimization for dvfs-enabled processors in embedded systems, ISQED (**2007**), pp. 204–209.
27. S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. D. Micheli, Temperature-aware processor frequency assignment for mpsocs using convex optimization, CODES+ISSS (**2007**), pp. 111–116.
28. R. W. Pike, Optimization for Engineering Systems, Van Nostrand Reinhold Company (**2001**), ISBN: 0442275811.
29. G. Quan and Y. Zhang, Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks, ECRTS (**2009**), pp. 207–216.
30. J. Rabaey, A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd edn., Prentice Hall Electronics and VLSI Series, Pearson Education, Upper Saddle River, NJ (**2003**).
31. R. Rao and S. Vrudhula, Performance optimal processor throttling under thermal constraints, *Proceedings of the 2007 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '07 (**2007**), pp. 257–266.
32. R. Rao, S. Vrudhula, C. Chakrabarti, and N. Chang, An optimal analytical solution for processor speed control with thermal constraints, ISLPED (**2006**), pp. 292–297.
33. M. Shaw, J. R. Waldrop, S. Chandrasekaran, B. Kagalwala, X. Jing, E. Brown, V. Dhir, and M. Fabbeo, Enhanced thermal management by direct water spray of high-voltage, high power devices in a three-phase, ITHERM (**2002**), pp. 1007–1014.

34. K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, Temperature-aware microarchitecture, ICSA (**2003**), pp. 2–13.

35. S. Wang and R. Bettati, Delay analysis in temperature-constrained hard real-time systems with general task arrivals, RTSS (**2006**), pp. 323–334.

36. S. Wang and R. Bettati, Reactive speed control in temperature-constrained real-time systems, ECRTS (**2006**), pp. 161–170.

37. X. Wang, K. Ma, and Y. Wang, Adaptive power control with online model estimation for chip multiprocessors. *Parallel and Distributed Systems, IEEE Transactions on* 22, 1681 (**2011**).

38. C.-Y. Yang, J.-J. Chen, L. Thiele, and T.-W. Kuo, Energy-efficient real-time task scheduling with temperature-dependent leakage, DATE (**2010**), pp. 9–14.

39. J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, Dynamic thermal management through task scheduling. *International Symposium on Performance Analysis of Systems and Software* (**2008**), pp. 191–201.

40. L.-T. Yeh and R. C. Chu, Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices, ASME Press, New York, NY (**2002**).

41. L. Yuan and G. Qu, ALT-DVS: Dynamic voltage scaling with awareness of leakage and temperature for real-time systems, *Adaptive Hardware and Systems, NASA/ESA Conference on* (**2007**), pp. 660–670.

42. S. Zhang and K. S. Chatha, Thermal aware task sequencing on embedded processors, DAC (**2010**), pp. 585–590.

43. Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan, Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. *University of Virginia Dept. of Computer Science Technical Report* (**2003**).

Huang Huang

Huang Huang *received his B.S. degree in electrical engineering from Northwest University, Xi'an, China in 2007 and the Ph.D. degree from Florida International University, Miami, FL, in 2012. His research interests include power and thermal-aware scheduling for real-time computing systems, advanced computer architecture and low-power and power-aware design for VLSI.*

Vivek Chaturvedi

Vivek Chaturvedi *received the M.S. degree from the Department of Electrical Engineering, Syracuse University, Syracuse, NY, in 2008. He is currently working towards the Ph.D. degree at the Electrical and Computer Engineering Department, Florida International University, Miami. During his studies at Syracuse University, he also worked as an Engineering Intern with the Micro-Electronics Group in SUN Microsystems, Burlington, MA. His research interests include real-time systems, operating systems, scheduling techniques, and computer architecture.*

Guanglei Liu

Guanglei Liu *received his bachelor degree from the Department of Electrical Engineering, Harbin University, Harbin, China, in 2006. He is currently working towards the Ph.D. degree at the Electrical and Computer Engineering Department, Florida International University, Miami. His research interests include Thermal/power aware computing and embedded realtime operating system design.*

Gang Quan

Gang Quan *is currently an Associate Professor with the Electrical and Computer Engineering Department, Florida International University (FIU), Miami. He received the B.S. degree from the Tsinghua University, Beijing, China, the M.S. degree from the Chinese Academy of Sciences, Beijing, and the Ph.D. degree from the University of Notre Dame, Notre Dame, IN. Before joining FIU, he was an assistant professor at the Department of Computer Science and Engineering, University of South Carolina. His research interests includes real-time system, power/thermal aware design, embedded system design, advanced computer architecture and reconfigurable computing. Professor Quan received the NSF CAREER award in 2006 and the Best Paper Award from the Design Automation Conference (DAC).*