2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010)

Leakage Aware Scheduling On Maximum Temperature Minimization For Periodic Hard Real-Time Systems

Vivek Chaturvedi Huang Huang Gang Quan

Electrical & Computer Engineering Department
Florida International University
Miami, Fl, USA, 33174
Email:{vchat001, hhuan001, gang.quan}@fiu.edu

Abstract

Thermal management is becoming a critical issue in computing system design as the processor power continues to grow exponentially. Since high power consumption leads to high temperature, and high temperature in turn dramatically increases the leakage power consumption, a thermal management technique becomes ineffective if this temperature/leakage relation is not properly addressed in the deep sub-micron domain. This paper incorporates the leakage/temperature dependency into real-time scheduling analysis and presents a novel real-time scheduling method that can reduce the peak temperature when scheduling a hard real-time periodic task set. We formally prove the correctness of the proposed algorithm based on a processor model that can effectively account for the leakage/temperature relationship. The experimental results validate the assumptions of our scheduling method and also demonstrate its effectiveness in terms of feasibility improvement and peak temperature reduction.

I. Introduction

As semiconductor technology continues to scale down, the chip temperature increases rapidly due to the exponentially growing power consumption. The escalating heat has directly led to high packaging and cooling costs, and threaten to significantly degrade the performance, life span, and reliability of future computing systems [18], [22]. Hence, the thermal issues have become increasingly prominent in the design of modern computing systems.

When dealing with the power consumption and thermal constraints in the deep sub-micron domain, the leakage plays a critical role. Based on the UC Berkeley's BSIM device model, Liao et al. [10] showed that the leakage power consumption can be 2-3 times higher than the dynamic power consumption for processors using the 65nm technology. Furthermore, they showed that when changing the temperature from 65^oC to 110^oC, the leakage power can increase as much as 38% [10]. High power consumption causes high temperature, and high temperature increases leakage power and thus the overall power consumption. Evidently, a thermal-conscious or power-conscious technique becomes ineffective if this temperature/leakage relation is not properly addressed in the deep sub-micron domain.

Researchers have already studied in depth the complex relationship between the leakage and temperature at the circuit and micro architecture level [10], [25], where the leakage current can be formulated as

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

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, $\mathcal{A}, \mathcal{B}, \alpha, \beta, \gamma, \delta$ are empirically determined technology constants. In addition, the variation of circuit delay (or the maximal processor speed that can be used) with temperature is also formulated as well [10]. Based on these results, a temperature modeling tool called "HotSpot" [1] was developed, which can be effectively used to simulate and study the processor thermal phenomena at the architecture level. However, such a model is too complex and cumbersome to be used for the purpose of system level analysis, such as real-time analysis and scheduling technique development.

In this paper, 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 for periodic task sets. We formally proved the correctness of this algorithm based on a processor

978-0-7695-4108-2/10 $26.00 © 2010 IEEE
DOI 10.1109/CIT.2010.312

power model that can capture the leakage/temperature dependency in reasonable accuracy yet simple enough and thus suitable for system level analysis. Furthermore, we validated the effectiveness of our algorithm based on the technology parameters derived from the 65nm technology. The experimental results demonstrate that our proposed scheduling technique can greatly reduce the peak temperature and, as a result, significantly improve the feasibility when scheduling periodic task sets under the maximum temperature constraints. Our work clearly shows that a more vigorous 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 II discusses the related work. System models are described in Section III. The *m-oscillating scheduling* algorithm is introduced in Section IV. Empirical results are presented in Section V, and Section VI concludes the paper.

II. Related Work

There have been an increasing number of research results published on thermal aware real-time scheduling, for both single and multiple processor platforms (e.g. [2], [6], [15], [24]). Some approaches (e.g. [2], [6]) try to identify the upper bound of the maximum temperature. Some others (e.g. [2], [8], [4], [21]) intend to minimize the peak temperature or to guarantee the given maximum temperature constraints when scheduling a job set or a single copy of a task graph. While it is a common practice to repeat a real time schedule developed for jobs within the first hyperperiod of a periodic task set, as indicated in [7], [15], this approach is not applicable anymore if the temperature constraint is taken into consideration.

For periodic task sets, Wang et al. [19], [20] studied the maximum delay for *periodic* tasks when scheduling real-time tasks based on a two-speed scheduling policy. Zhang et al. [24] proposed to guarantee the temperature feasibility of a periodic system by forcing the temperature at the end of its first hyperperiod to be equal or less than the starting temperature. Quan et al. [15] developed a closed formula for the feasibility analysis under the maximum temperature constraint. None of these researches has taken the temperature/leakage dependency into consideration.

Some researches, such as that by Bao et al. [3], have applied equation (1) directly to capture the leakage/temperature dependency for the scheduling analysis. However, due to the non-linear and high-order magnitude terms in equation (1), such a model or tool can be too complex and cumbersome to be used for more rigorous real-time analysis and scheduling technique development. For example, Yuan et al. [23] also studied how to directly incorporate equation (1) into scheduling decisions. However, due to the complexity of equation (1), their approach can only be applied for soft real-time systems. There are also a number of other approaches formulate the temperature-constrained problem as a convex optimization problem [12], [13], [4]. The leakage/temperature dependency (equation (1)) may be incorporated into the optimization formulation [12]. The problem is that the computational complexity of the convex optimization problem is very high. Therefore these approaches can only work at system level when the design solution space is small.

A number of recent researches try to simplify the leakage/temperature dependency model. Liu et al. observed that the leakage current changes super linearly with temperature [11]. Based on this observation, a number of researches (such as [7], [9], [5] adopt a simple temperature/leakage dependency model that assumes the leakage current changes linearly *only* with temperature. However, as can be seen from equation (1), leakage varies not only with temperature but also supply voltage as well. Quan et al. [14] 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, with leakage/temperature dependency in mind, we develop a novel and scheduling technique that can effectively reduce the maximum temperature.

III. The system models

In this section, we introduce the system models that are used in this paper.

The real-time model The real-time system we consider 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 assume that the deadline of the task equals its period.

The thermal model We use the RC thermal model that has been widely used in the similar research (e.g. [5], [7], [13], [14]). Specifically, assuming a fixed ambient temperature (T_{amb}), let $T(t)$ be the temperature at time t. Then we have

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

where $P(t)$ denotes the power consumption (in $Watt$) at time t, and R, C denote the thermal resistance (in $J/^oC$)

and thermal capacitance (in $Watt/^oC$). 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 and its power model The processor can run in n different modes, with each mode as (v_i, f_i), $i = 0, 1, ..., n-1$. 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 proportional to the supply voltage. In what follows, we use processor speed and supply voltage interchangeably.

Given a voltage level v, the power consumption is composed of dynamic P_{dyn} and leakage P_{leak}, i.e. $P = P_{dyn} + P_{leak}$. According to Liao et al. [10], the leakage power consumption can be estimated by the following,

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

where N_{gate} represents the number of gate, v is the voltage level, and I_{leak} can be formulated using equation (1). As leakage current changes super linearly with temperature [11], we can simplify P_{leak} and define the leakage power for the processor running in mode k as

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

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

The dynamic power consumption is independent of temperature, and can be formulated $P_{dyn} = C_2 v_k^{\xi} (\xi > 0)$. We choose $\xi = 3$ [16] in this paper[1]. Hence the total power consumption at processor mode k is

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

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

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

where

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

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

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

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

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

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. Equation (6) to (10) form the basis of our system level thermal analysis with leakage/termpature interplay taken into account.

IV. Scheduling for peak temperature reduction

In this section, we study how to minimize the maximum temperature when scheduling a periodic task set. 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 1, that can finish a real-time job at its deadline. Note that, the dynamic energy consumption by the two-speed schedule shown in Figure 1 remains the same as long as the length for each individual speed keeps the same, 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 characteristic.

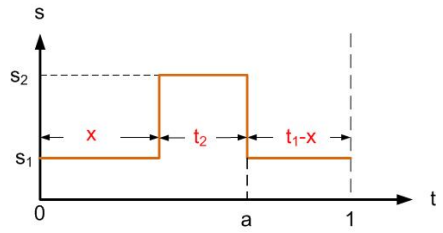


Fig. 1. A two-speed schedule that uses speed s_1 for t_1 time units and speed s_2 for t_2 time units. $t_1 + t_2 = 1$.

Theorem 1: Given a two-speed schedule as shown in Figure 1, 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 equation (11) and (9), respectively), then the temperature at $t = 1$, i.e.T_e is a monotonically increasing function of x.

Proof sketch: Based on equation (10), let T_a be the temperature at point a, then we have

$$T_e = G_1 + (T_a - G_1)e^{-B_1(t_1 - x)} \tag{12}$$

Therefore,

$$\begin{aligned} \frac{d(T_e)}{dx} &= ... \\ &= (G_2 - G_1)(1 - e^{-B_2 t_2})B_1 e^{-B_1(t_1 - x)}. \end{aligned} \tag{13}$$

So T_e decreases with decreasing x if $G_2 > G_1$ and $B_1, B_2 > 0$. □

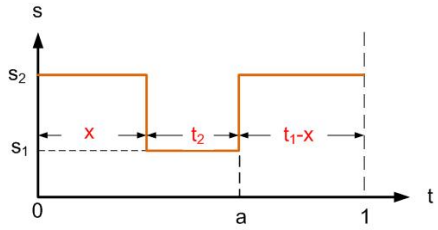


Fig. 2. A two-speed schedule that uses speed s_2 for t_1 time units and speed s_1 for t_2 time units. $t_1 + t_2 = 1$.

Similarly, for the two-speed schedule illustrated in Figure 2, we have Theorem 2.

Theorem 2: Given a two-speed schedule as shown in Figure 2, 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 equation (11) and (9), respectively), then the temperature at $t = 1$, i.e.T_e is a monotonically decreasing function of x.

Proof sketch: Based on equation (10), T_a be the temperature at a, then we have

$$T_e = G_2 + (T_a - G_2)e^{-B_2(t_1 - x)} \tag{14}$$

Therefore,

$$\begin{aligned} \frac{d(T_e)}{dx} &= \ldots \\ &= -(G_2 - G_1)(1 - e^{-B_1 t_2})B_2 e^{-B_2(t_1 - x)}. \end{aligned} \tag{15}$$

So T_e decreases with increasing x if $G_2 > G_1$ and $B_k > 0$. □

Theorem 1 and 2 indicate that temperature at the end of a schedule depends on locations where different running modes are applied. They help to reduce the temperature at the end of schedule, but do not necessarily reduce the maximum temperature within the entire interval. In addition, Theorem 1 and 2 are applied for a single job rather than a periodic task set. In what follows, we introduced a novel scheduling algorithm (we call it 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 stable status. The temperature stable status is defined below.

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

$$|T((n+1)p) - T(np)| < \varepsilon, \tag{16}$$

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

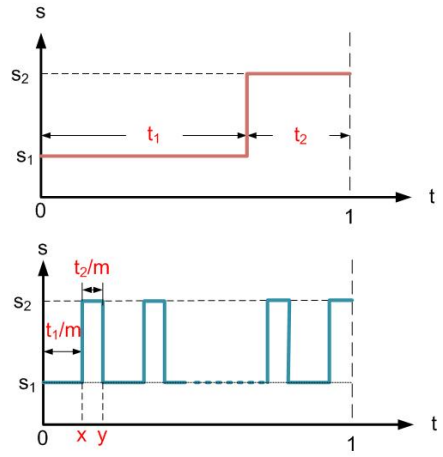


Fig. 3. A two-speed schedule and its corresponding m-oscillating schedule.

Our *m-oscillating* algorithm works as follows: given a two-speed schedule, we can 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-oscillation schedule will complete the same workload as the original 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 3: 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

- $T_{max}(\tilde{S}(m,t)) \leq T_{max}(S(t))$;
- $T_{max}(\tilde{S}(m,t)) \leq T_{max}(\tilde{S}(m,t))$ if $m \leq n$.

Proof sketch: Here we only present the partial proof, i.e. for the case shown in Figure 3.

For $\tilde{S}(m,t)$ shown in Figure 3, base on equation (10), the temperature at $t = x$ and $t = y$ can be formulated as

$$T_x = G_1(1 - e^{-B_1 t_1/m}), \quad T_y = G_2 + (T_x - G_2)e^{-B_2 t_2/m}$$

From [14], when the temperature reaches the stable status, we have

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

where

$$K_y = e^{-\frac{(B_1 t_1 + B_2 t_2)}{m}}.$$

Expand T_y^{∞}, we have

$$T_y^{\infty} = (G_2 - G_1)\frac{1 - e^{-\frac{B_2 t_2}{m}}}{1 - e^{-\frac{(B_1 t_1 + B_2 t_2)}{m}}} + G_1$$

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

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

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,

$$\begin{aligned} & \frac{\sum_{i=0}^{m-1} e^{-ip}}{\sum_{i=0}^{m-1} e^{-i(p+q)}} < \frac{\sum_{i=0}^{m} e^{-ip}}{\sum_{i=0}^{m} e^{-i(p+q)}} \\ \Longleftarrow \quad & e^{-m(p+q)} \cdot \sum_{i=0}^{m-1} e^{-ip} < e^{-mp} \cdot \sum_{i=0}^{m-1} e^{-i(p+q)} \\ \Longleftarrow \quad & e^{-mq} \cdot \sum_{i=0}^{m-1} e^{-ip} < \sum_{i=0}^{m-1} e^{-i(p+q)} \\ \Longleftarrow \quad & \sum_{i=0}^{m-1} e^{-ip} < \sum_{i=0}^{m-1} e^{-ip} \cdot e^{(m-i)q}. \end{aligned}$$

With $i \le m$, $e^{(m-i)q} \ge 1$, therefore $f(m+1) > f(m)$, and so

$$T_{max}(\tilde{S}(m,t)) > T_{max}(\tilde{S}(m+1,t)). \tag{17}$$

□

Theorem 3 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 stable status. The larger the m is, the lower the maximum temperature becomes.

Note that the conclusion in Theorem 3 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 equation (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 V, we validate these assumptions empirically. Moreover, it is worth mentioning that Theorem 3 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 [5] is an interesting problem and will be our future work.

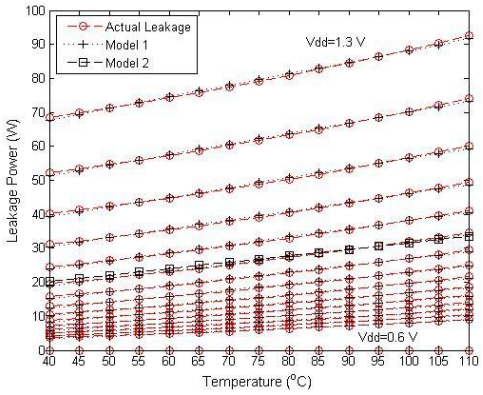


Fig. 4. Linear approximation of leakage power consumptions.

V. Experiments and results

In this section, we use experiments to examine the m-oscillating scheduling algorithm. First, we validate the processor model as well as the assumptions that the algorithm is built upon. We then evaluate its performance by comparing it with a previous work, i.e. the two-speed scheduling method [20], in terms of the feasibility and peak temperature.

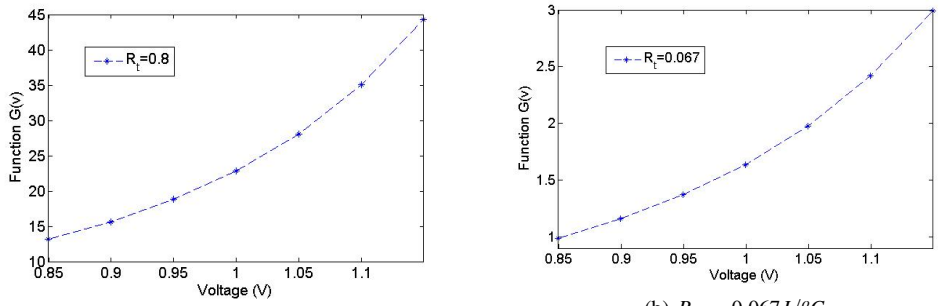
A. Verification of the processor model and assumptions

To verify the processor model and assumptions made in Theorem 1 to 3, we built our processor models based on the work by Liao et al. [10] using the 65nm technology from the UC Berkeley's BSIM device model. Specifically, we used equation (1) to compute the leakage currents for temperature from 40^oC to 110^oC with a step size of 5^oC, and supply voltage from 0.60 Volt to 1.30 Volt with a step size of 0.05V. These results were used to determine the temperature invariants $C_0(k)$ and $C_1(k)$ in equation (5) through curve-fitting.

We set the frequency for each mode according to the formula [10]

$$f = \frac{1}{delay} = \frac{(v-v_t)^\mu}{vT^\eta} \times 4.2824 \times 10^{14} \tag{18}$$

with $\mu = 1.19$, $\eta = 1.2$, $v_t = 0.3$, and T is set to the highest temperature as 100^oC, and we also normalized the frequency with highest equal to 1.0. To obtain the leakage power consumption, we set N_{gate} in equation (4) to be 10^6. The dynamic power consumption (and thus constant C_2) was determined based on experimental results reported in [10] on a common benchmark *gcc*. For



(a) $R_{th} = 0.8J/^oC$. (b) $R_{th} = 0.067J/^oC$.

Fig. 5. Function $G(v)$ with different temperatures and thermal resistances.

thermal constants, we considered two different options, first is the conventional air cooling with $Rth = 0.8K/W$, $Cth = 340J/K$ [18] and second is the water spray-cooling with $Rth = 0.8K/W$, $Cth = 340J/K$ [17]. The ambient temperature was set to 25^oC.

Figure 4 compares the estimated leakage power consumptions using two different leakage/dependency models (namely,*Model 1 and Model 2*) with the "*actual leakage*", which is calculated based on equation (1). Model 1 is the leakage/temperature model used in this paper (i.e. section III), assuming that the leakage varies with both temperature and supply voltage. Model 2, used in [7], [9], [5], assumes leakage changes with the temperature linearly but not with the supply voltage. As we can see from Figure 4, the linear approximated leakage power consumptions based on Model 1 match very closely to that calculated based on equation (1), with the maximum relative error no more than 7%. 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 supply voltage that is applied.

We also examined the assumptions made in Theorem 1 to 3. Figure 5 and Figure 6 plot the characteristics of function G_k and B_k under different supply voltages and thermal constants. As illustrated in these two figures, we can clearly see that, under both representative cooling options (i.e. $R_{th} = 0.8J/^oC$ and $R_{th} = 0.067J/^oC$), 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 Theorem 1 to 3.

B. Performance evaluation

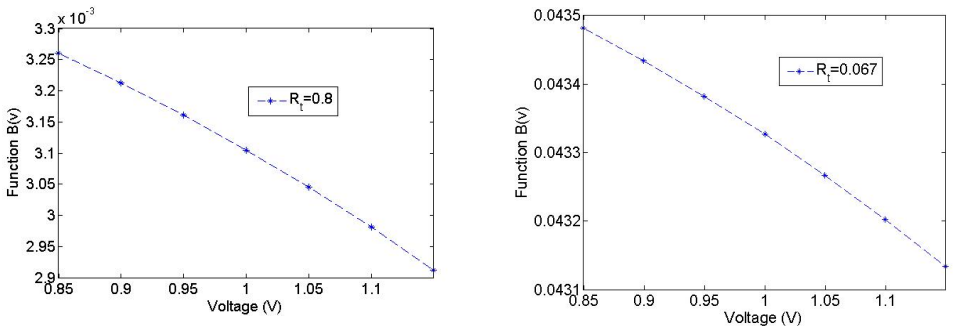
We next study the performance of *m-oscillating scheduling* by comparing with the existing approaches. The proactive scheduling method introduced in [7] intends to minimize the task response time under given maximum temperature constraints. However, it is developed based on a processor model with continuously 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 straight forward effort. Therefore, we compare our approach with a more general one, i.e. the reactive two-speed scheduling approach introduced in [20]. The reactive two-speed schedule [20] work 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.

TABLE I. The equilibrium speeds and the corresponding maximum temperatures

$V_{Equil}(V)$	$T_{max}(^oC)$
0.80	33.99
0.90	38.88
1.0	46.16

First, we want to investigate the feasibility of the two scheduling policies, i.e. the *m-oscillating schedule* and the reactive two-speed schedule, under the same maximum temperature constraints and workloads. 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 constraint to test both scheduling policies.

We randomly generated real-time tasks with period of 2000 seconds and workload evenly distributed within 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. The equilibrium voltages were set to be 0.8V, 0.9V and 1.0V, and the corresponding stable temperature were set as the



(a) $R_{th} = 0.8J/^oC$. (b) $R_{th} = 0.067J/^oC$.

Fig. 6. Function $B(v)$ with different temperatures and thermal resistances.

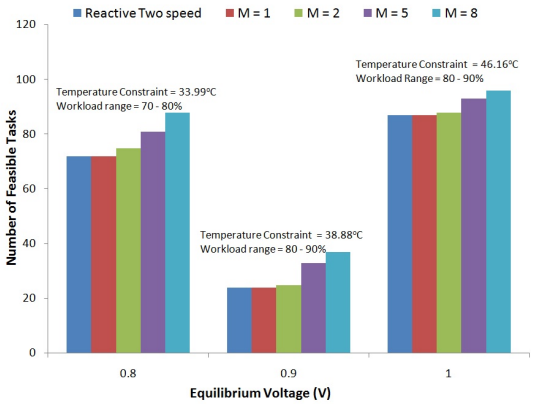


Fig. 7. Feasibility comparison between the m-oscillation scheme and the reactive two-speed scheme under different maximum temperature constraints

maximum temperature constraint. Table I lists the values of the equilibrium voltages and their corresponding stable temperatures. For *m-oscillating schedule*, we first calculated the constant speed that will guarantee workload. Then the two neighboring speeds were used to construct our *m-oscillating schedule* algorithm described in section IV.

Figure 7 presents the feasibility differences between active two-speed scheduling and m-oscillating scheduling with m=1, 2, 5, and 8. When the randomly generated workload is very low, all above scheduling policies can schedule the task feasibly; and when the workload is high, none four scheduling policies can make the task feasible. Therefore Figure 7 only depicts the workload regions that there exist differences in terms of feasibility among different scheduling choices. From Figure 7, we can clearly see that *m-oscillating scheduling* shows higher feasibility as compared to reactive two-speed schedule. The larger the m is, the higher the feasibility can be. At the equilibrium voltage of 0.8V, the feasibility by the active two-speed scheduling policy is very close to the m-oscillating scheduling algorithm. However, when m is increased to 5, the feasibility is improved over 13%, and up to 20% when $m = 8$. At the equilibrium voltage of 0.9V and 1.0V, we can see the feasibility improvement of 35% and 10%, respectively, by m-oscillating algorithm to the two-speed scheduling algorithm.

Even though a task can be feasibly scheduled, a higher peak temperature is not desirable since it increases packaging and cooling costs, degrade the performance, life span, and reliability of a computing system. We therefore collected the maximum temperatures of all feasible tasks under different scheduling policies and compared their average maximum temperatures as shown in Figure 8. Figure 8 clearly demonstrates that the m-oscillating algorithm is very effective in reducing the peak temperature. Note that at the equilibrium voltage of 0.8V, the average maximum temperature of reactive two-speed schedule is 31.84^oC, and is reduced to 28.64^oC when $m = 1$ for the m-oscillating scheduling algorithm. It is further reduced to 27.85^oC for m = 5. At equilibrium voltage of 0.9V and 1.0V, the average maximum temperatures are reduced by 7.78^oC and 14.0^oC, respectively.

VI. Summary

As semiconductor technology continues to scale down, the positive feedback loop between 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 into the real-time scheduling analysis that aims at minimizing the maximum temperature. We presented and proved a number of theorems and exhibit the distinct characteristics of thermal aware real-time scheduling. We also

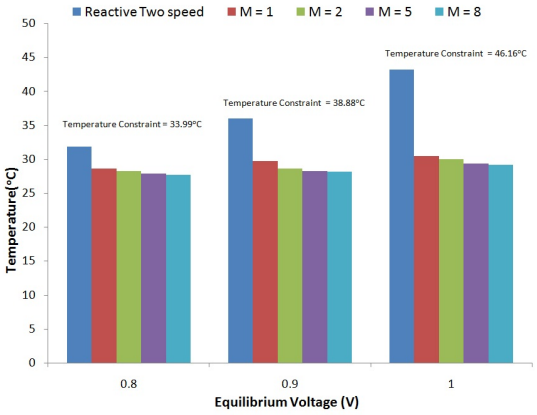


Fig. 8. Average maximum temperature comparison between the m-oscillation scheme and the reactive two-speed scheme

proposed a new scheduling technique, i.e. the *m-oscillating scheduling* that can effectively reduce the peak temperature when executing a hard real-time periodic task set. These theorems and techniques form a solid basis for further leakage-aware temperature-constrained researches in design and development of practical real-time systems. Our future research will be based on the theorems presented in this paper and extended in a number of ways, including more complex real-time system models, processors with non-trivial transition overhead, and multiple-core type of architectures.

Acknowledgement

This work is supported in part by NSF under projects CNS-0545913 and CNS-0917021.

References

[1] Hotspot 4.2 temperature modeling tool. *University of Virgina*, page http://lava.cs.virginia.edu/HotSpot, 2009.

[2] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1):1–39, 2007.

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

[4] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. In *DATE*, pages 288–293, 2008.

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

[6] J. Chen, C. Hung, and T. Kuo. On the minimization fo the instantaneous temperature for periodic real-time tasks. *RTAS*, pages 236–248, 2007.

[7] J.-J. Chen, S. Wang, and L. Thiele. Proactive speed scheduling for real-time tasks under thermal constraints. *RTAS*, 0:141–150, 2009.

[8] 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(1):6–9, 2003.

[9] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. Thermal-aware global real-time scheduling on multicore systems. *RTAS*, 0:131–140, 2009.

[10] 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(7):1042 – 1053, 2005.

[11] Y. Liu, R. P. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *DATE*, pages 1526–1531, 2007.

[12] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang. Thermal vs energy optimization for dvfs-enabled processors in embedded systems. In *ISQED*, pages 204–209, 2007.

[13] 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*, pages 111–116, 2007.

[14] G. Quan and Y. Zhang. Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks. *ECRTS*, pages 207–216, 2009.

[15] G. Quan, Y. Zhang, W. Wiles, and P. Pei. Guaranteed scheduling for repetitive hard real-time tasks under the maximal temperature constraint. *ISSS+CODES*, 2008.

[16] J. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 2003.

[17] 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*, pages 1007–1014, 2002.

[18] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. *ICSA*, pages 2–13, 2003.

[19] S. Wang and R. Bettati. Delay analysis in temperature-constrained hard real-time systems with general task arrivals. *RTSS*, pages 323–334, 2006.

[20] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. *ECRTS*, pages 161–170, 2006.

[21] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin. Dynamic thermal management through task scheduling. In *International Symposium on Performance Analysis of Systems and Software*, pages 191–201, 2008.

[22] 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.

[23] 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*, 0:660–670, 2007.

[24] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *ICCAD*, pages 281–288, 2007.

[25] 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.