Leakage Aware Feasibility Analysis for Temperature-Constrained Hard Real-Time Periodic Tasks

Gang Quan Yan Zhang
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29063
Email: gquan,zhangy@cse.sc.edu

Abstract

As semiconductor technology continues to evolve, the chip temperature increases rapidly due to the exponentially growing power consumption. In the meantime, the high chip temperature increases the leakage power, which is becoming the dominate part in the overall power consumption for sub-micron IC circuits. A power/thermal-aware computing technique becomes ineffective if this temperature/leakage relation is not properly addressed in the sub-micron domain.

In this paper, we study the feasibility problem for scheduling a hard real-time periodic task set under the peak temperature constraint, with the interaction between temperature and leakage being taken into consideration. Three analysis techniques are developed to guarantee the schedulability of periodic real-time task sets under the maximal temperature constraint. Our experiments, based on technical parameters from a processor using the 65nm technology, show that the feasibility analysis without considering the interactions between temperature and leakage can be significantly overoptimistic.

1. Introduction

According to Borkar et al. [1] from Intel, there are more than 10 billion transistors integrated into single $300mm^2$ die today, and the number is growing rapidly toward 100B by the middle of the next decade. The power consumed by these transistors is also tremendous, reaching 300W in early next decade. The exponentially increased power consumption has posted significant challenges not only on how to provide sufficient power source to an electronic system but also on how to dissipate the heat generated by the system.

The thermal issues have become increasingly prominent as power consumption continues to grow. The high chip temperature not only increases package/cooling cost (estimated at 1-3 dollar per watt [2]) but also adversely affects the reliability and performance of the computing systems, and can even cause a system to fail catastrophically. Even if a processor is not totally failed at the high temperature, a small increase in the temperature (e.g.10^oC) can result in significant reduction (50%) in its life span [3]. From the perspective of real-time systems, when the chip temperature exceeds certain limit, the self-protection controls in some new generation processors may be invoked automatically by reducing the performance and thus cause tasks to miss deadlines.

The leakage plays an essential role in power- and thermal-aware design in the sub-micron domain, where the leakage power consumption is becoming more and more significant in the overall power consumption. Liao et al. [4] have shown that the leakage power consumption can be 2-3 times higher than the dynamic power consumption for processors using the 65nm technology. In addition, there is a strong relationship between the leakage and temperature [4], [5] in sub-micron circuits. High power consumption leads to high chip temperature, and high chip temperature in turn increases power consumption dramatically. When changing the temperature from 65^oC to 110^oC, Liao et al. [4] have shown that the leakage power can increase as much as 38%. Evidently, it becomes a critical issue to address the temperature/leakage dependence in developing power and thermal aware techniques for next generations of electronic systems.

Since higher power consumptions lead to higher temperatures, one intuitive way is to applying existing power-aware scheduling techniques to solve this problem. However, it has been shown that an optimal schedule that can minimize the energy consumption is not necessarily the optimal solution in a thermal constrained environment [6]. New techniques explicitly addressing the thermal issues need to be developed.

Recently, the real-time scheduling problem under thermal constraints has attracted many research interests(e.g. [6]–[10]). Bansal et al. [6] and Chen et al. [7] studied the temperature-aware EDF scheduling problem and identified an upper bound for the temperature. These techniques cannot guarantee that real-time tasks can still meet deadlines when the maximal temperature is given. For a given maximal temperature, Bansal et al. [6] introduced an off-line technique to minimize the energy consumption for a *job set*. Chantem et al. [11] also proposed an MILP-based solution to minimize the peak temperature when executing a task graph on MPSoCs. However, as shown by Quan et al. [12], if the job set or the task graph needs to be periodically or repetitively executed, the schedules developed in [6], [11] can no longer guarantee the maximal temperature constraints. Several scheduling algorithms for periodic tasks under the maximal temperature constraint were also developed [12]. Zhang et al. [10] proposed to guarantee the maximal temperature constraints for a periodic task set by forcing the temperature at the end of its first hyperperiod to be no more than that at the starting point. As implied by the results in [12], this approach overly constrains the feasible region and can lead to very pessimistic scheduling results. To ease the analysis of thermal analysis, Wang et al. [8], [9] proposed a two-speed processor model, i.e. a processor runs at the maximal processor speed before it reaches the limit, and then runs with the "equilibrium speed" at which the processor enters the equilibrium state with its temperature unchanged. The maximal delay analysis presented in papers [8], [9] for periodic task sets under the fixed priority as well as the first-in-first-out policy can be used for the purpose of scheduability analysis. These approaches do not take the advantage of the fact that many processors support more than two running speeds. In addition, none of the above work has taken the leakage/temperature relationship into consideration.

A few recent papers incorporate the temperature/leakage dependency into the energy- or thermal-aware scheduling. He et al. [13] and Yuan et al. [14] studied how to reduce the leakage power at the system level. Yuan et al. [15] introduced an offline and an on-line scheduling algorithm that take into account the leakage/temperature interactions when scheduling a set of soft real-time jobs. This approach cannot guarantee that real-time periodic tasks can meet deadlines under the given maximal temperature. A number of other approaches formulate the temperature-constrained problem as a convex optimization problem (e.g. [16], [17]). The leakage/temperature relationship can thus be formulated as one of the constraints. The problem is that the computational complexity for convex optimization problems is very high. Therefore these approaches can only work at system level when the design solution space is small.

In this paper, we study the problem on how to guarantee the feasibility of periodic tasks under the maximal temperature constraint, with the interplay of temperature and leakage taken into consideration. We developed a novel power model that can capture the interaction between the temperature and leakage with high accuracy. Yet the model is also simple enough for ease of system level analysis. Based on this model, we then developed three conditions to check the feasibility of real-time tasks under the peak temperature requirement. We conducted experiments based on technical parameters derived from a processor using the 65nm technology. The results clearly demonstrate that the feasibility analysis without considering the interaction between temperature and leakage can be significantly overoptimistic.

The rest of the paper is organized as follows. Section 2 discusses the system models and formulates our problem formally. Section 3 presents three theorems to determine the schedulability of a schedule under temperature constraints. In Section 4, we presents our experimental results. Section 5 draws the conclusions.

2. System models

The real-time system considered in this paper contains n independent periodic tasks, $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$, scheduled according to the earliest deadline first (EDF) policy. Task τ_i is characterized using three parameters, *i.e.*, $\tau_i = (p_i, d_i, c_i)$. p_i, d_i ($d_i \leq p_i$), and c_i represent the period, the deadline and the worst case execution time for τ_i, respectively. We assume all tasks start at the same time, and each task contains an infinite sequence of periodically arriving instances called *jobs*.

2.1. Thermal model

We use the lumped RC model similar to Shadorn et al. [18] to capture the thermal phenomena of the processor. Specifically, assuming a fixed ambient temperature (T_{amb}), let $T(t)$ denote the temperature at time t. Then we have

$$RC\frac{dT(t)}{dt} + T(t) - RP(t) = T_{amb}, \tag{1}$$

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

$$\frac{dT(t)}{dt} = aP(t) - bT(t), \tag{2}$$

where $a = 1/C$ and $b = 1/RC$. For the rest of the paper, we assume that the initial temperature of the processor equals the ambient temperature, i.e. $T_0 = 0$.

2.2. Processor and power model

We assume the processor can run in different modes, with each mode being characterized by a pair of parameters (v_i, f_i), where v_i is the supply voltage and f_i is the working frequency in mode i. Even though the circuit delay changes with the temperature dynamically, as shown in equation (3) [4],

$$f_{max} = \frac{1}{t_d} \propto \frac{(V_{dd} - v_t)^\mu}{V_{dd}T^\eta}, \tag{3}$$

where t_d is the circuit delay, and μ and η are technology-related constants, we assume that the processor working frequency in each mode is fixed, and is the one that can accommodate the peak temperature (i.e. by assigning the peak temperature in equation 3) across the chip. Let f_{max} be the largest f_i among different modes. We can normalize the processor working frequency with f_{max} and get the normalized processor speed for each mode. In what follows, unless otherwise specified, we use the term processor speed or working frequency interchangeably.

The power consumption (P) of the processor consists of two parts: the dynamic power(P_{dyn}) and the leakage power (P_{leak}).

$$P = P_{dyn} + P_{leak}. \tag{4}$$

The dynamic power is independent of the temperature, but the leak power is sensitive to the temperature. Recently, several researches have disclosed the leakage and temperature relationship at the microarchitecture and device level [4], [5], [19]. According to Liao et al. [4], the leakage power can be estimated using the following formulas,

$$P_{leak} = N_{gate} \cdot I_{leak} \cdot V_{dd} \tag{5}$$

where N_{gate} is the total number of gates, V_{dd} is the supply voltage and

$$I_{leak} = I_s \cdot (\mathcal{A} \cdot T^2 \cdot e^{((\alpha \cdot V_{dd} + \beta)/T)} + \mathcal{B} \cdot e^{(\gamma \cdot V_{dd} + \delta)}) \tag{6}$$

where I_s is the leakage current at certain reference temperature and supply voltage, T is the temperature, $\mathcal{A}, \mathcal{B}, \alpha, \beta, \gamma, \delta$ are empirically determined constants.

As reported in Liao et al. [4], using equation (6) can accurately estimate the leakage current under different temperatures, with relative error less than 1%. While equation (6) captures the complex relationship between leakage the temperature, it is too complicated to be used for real-time analysis. Liu et al. [19] found that using linear approximation method to model the leakage/temperature dependence can maintain reasonable accuracy, i.e. with error within 1% using the piece-wise linear function or less than 5.5% using single linear function, but the leakage model is significantly simplified. Based on this idea, we define the leakage power for the processor running in mode k as

$$P_{leak}(k) = (C_0(k) + C_1(k)T) \cdot v_k, \tag{7}$$

where $C_0(k)$ and $C_1(k)$ are constants that depend on the running mode, i.e. k. In what follows, we omit variable k for sake of conciseness.

The dynamic power consumption is independent to the temperature and can be formulated as $P_{dyn} = C_2 v_k^3$ [20]. The overall power consumption in mode k is thus given by the following formula.

$$P(k) = (C_0 + C_1 T) \cdot v_k + C_2 v_k^3. \tag{8}$$

C_0, C_1 and C_2 can be determined in practice once the practical power consumptions at different temperatures are profiled. In Section 4, we show an example and study the accuracy of this model using technical parameters drawn from a processor using the 65nm technology.

When the processor runs in mode k, by combining equation (2) and (8), we have temperature variations as follows:

$$\frac{dT(t)}{dt} = A(k) - B(k)T(t), \tag{9}$$

where

$$\begin{aligned} A(k) &= a(C_0 v_k + C_2 v_k^3) && (10) \\ B(k) &= (b - aC_1 v_k) && (11) \end{aligned}$$

If we run processor in mode k during interval $[t_1, t_2]$, with temperature at $t = t_1$ be $T(t_1)$, by solving equation (9), we can get the temperature at $t = t_2$ as

$$T(t_2) = \frac{A(k)}{B(k)} + (T(t_1) - \frac{A(k)}{B(k)})e^{-B(k)(t_2 - t_1)}. \tag{12}$$

2.3. Problem formulation

Varying processor supply voltage and working frequency is one of the most effective ways to manage the power consumption dynamically. We call a schedule that dictates how to vary the processor supply voltage and working frequency as the *speed schedule*. We are interested in studying if a processor speed schedule that can guarantee the deadlines of all tasks can still do so under the maximal temperature constraint. Since the processor can only run in a number of modes, the speed schedule is thus formally defined as follows.

Definition 1: Given periodic task set $\mathcal{T}$, let L be the least common multiple (LCM) of the periods, i.e.,$p_0, p_1, ..., p_{n-1}$. The speed schedule $\hat{S}(t)$ is defined as a sequence of $< [st_i, ed_i], mode_i >$, where

- $[st_i, ed_i]$ is an interval in which the processor runs in $mode_i$,
- $\bigcup_i [st_i, ed_i] = [0, L]$, and
- $[st_i, ed_i] \bigcap [st_j, ed_j] = \emptyset$ if $i \neq j$.

With the thermal and processor models introduced as above, our problem can be formulated as follows:

Problem 1: Given

- a hard real-time task set $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$,
- a variable voltage processor that can run in m different modes, i.e. (v_i, f_i), $i = 0, ...,m - 1$,
- the maximal allowable temperature T_{max},
- and a speed schedule $\hat{S}(t)$ with l intervals, i.e. $< [t_i, t_{i+1}], mode_i >$, $l = 0, 1, ..., l - 1$,

determine if $\mathcal{T}$ can meet the required deadlines using $\hat{S}(t)$ with the temperature stays below T_{max} all the time.

3. Feasibility analysis

One common approach to analyze the schedulability of a periodic real-time task set is to determine if all jobs located within the first hyperperiod—the interval $[0, L]$ where L is the least common multiple of task periods—can meet their deadlines. As long as all jobs are feasible in $[0, L]$, by replicating the schedule, all other jobs will meet their deadlines as well. Interestingly, as shown by Quan et al. [12], when temperature impacts are taken into consideration, this approach does not apply any more. In this section, we introduce three feasibility conditions to predict if a periodic real-time task set can meet deadlines under the maximal temperature constraint.

3.1. Checking the temperature at the end of first hyperperiod

The reason that a periodic task set feasible within the first hyperperiod is not necessarily feasible later in its life time is that the temperature at the end of a hyperperiod may be higher than that at the beginning of the hyperperiod. If this is case, starting at a new hyperperiod, the processor will run at a higher initial temperature and continue to reach an even higher temperature at the end of this hyperperiod. As this process continues, the temperature may eventually exceed the peak temperature. Conversely, if the ending temperature of the first hyperperiod is lower than the initial temperature, a feasible schedule that can satisfy the maximal temperature constraint will still be feasible during its life time. We summary this conclusion in the following theorem.

Theorem 1: Assume that $\hat{S}(t)$ can guarantee the deadlines of $\mathcal{T}$ under the maximal temperature constraint T_{max}. Then, when repeating $\hat{S}(t)$, all task deadlines can be guaranteed with temperature below T_{max} if $T(L) \leq T(0)$.

Theorem 1 can be proved by simply noting that, since $T(L) \leq T(0)$, the second hyperperiod always starts at the same or a more favorable situation. Therefore, the temperature constraints will not be violated.

From Theorem 1, as long as we can ensure that the temperature within the first hyperperiod is not higher than T_{max}, and as long as the ending temperature is not higher than the initial temperature, we can determine whether the schedule is feasible or not under the given maximal temperature constraint. However, assuming the initial temperature be the ambient temperature, unless some aggressive cooling strategies are applied, the temperature of a processor will always increase when executing tasks. Therefore, the applicability of Theorem 1 is very limited. Next, we introduce two other theorems that can effectively deal with the case when $T(L) > T(0)$.

3.2. Checking the temperature safe modes

Recall that, in Section 2.2, the processor can work in different modes, each of which is associated with a distinct pair of supply voltage and working frequency. For some of the modes, no matter how long the processor runs in that mode, the final temperature will never exceed the given T_{max}. We call these processor modes as the *safe modes*.

To determine if a processor mode, i.e. mode k, is safe, we can set

$$\frac{dT(t)}{dt}\,|_{T(t)=T_{max}} = 0, \tag{13}$$

Based on equation (2) and (8), we have

$$a((C_0 + C_1 T_{max}) \cdot v + C_2 v^3) - bT_{max} = 0. \tag{14}$$

Note that equation (14) is the classic *depressed cubic equation* [21]. In addition, if we transform equation (14) slightly, we have

$$C_2 v^3 = -a((C_0 + C_1 T_{max})) \cdot v + bT_{max}. \tag{15}$$

From Section 2, we can see that $C_2 > 0$, and $a((C_0 + C_1 T_{max})) > 0$. Therefore, as illustrated in Figure 1, equation (14) has only one single *real* root for v, which can be solved analytically [22]. We call the solution to equation (14) as the *equilibrium voltage*. Note that different processor running modes may have different equilibrium supply voltages since C_0 and C_1 in equation (15) are different in different modes.

Formally, we have the following lemma to determine whether or not a processor running mode is a safe mode.

Lemma 1: Let v_e be the equilibrium voltage (i.e. the solution to equation (14)) for processor's mode k (i.e. (v_k, f_k)). Then this mode is a *safe mode* if $v_e \geq v_k$.

Proof: We prove it by contradiction. Assume that at time $t \leq t_0$ we have $T(t) = T_{max}$ but at $t_0 + \triangle t$, we have $T(t_0 + \triangle t) > T_{max}$. So we must have

$$\frac{dT(t)}{dt}\,|_{t=t_0} = \frac{dT(t)}{dt}|_{T(t)=T_{max}} > 0. \tag{16}$$

On the other hand, since $v_e \geq v_k$, from equation (14), we have

$$\begin{aligned} \frac{dT(t)}{dt}|_{t=t_0} &= a((C_0 + C_1 T_{max}) \cdot v_k + C_2 v_k^3) - bT_{max} \\ &\leq a((C_0 + C_1 T_{max}) \cdot v_e + C_2 v_e^3) - bT_{max} \\ &= 0, \end{aligned} \tag{17}$$

which contradicts equation (16). □

Based on Lemma 1, we can formulate our second feasibility checking method in the following theorem.

Theorem 2: Let $\hat{S}(t)$ be the speed schedule within interval $[0, L]$ that can guarantee the deadlines of $\mathcal{T}$ under the maximal temperature constraint T_{max}, and let s_{max} be the maximal speed in $\hat{S}(t)$. Also let s_{mf} be the highest speed among the processor safe modes. Then if $s_{max} \leq s_{mf}$, when repeating $\hat{S}(t)$ later in the schedule, the temperature will never exceed T_{max}.

Theorem 2 can be easily proved following the similar proof as that for Lemma 1. Note that, as long as the maximal temperature T_{max} and the processor is given, the highest speed (s_{mf}) among the processor safe modes is well determined.

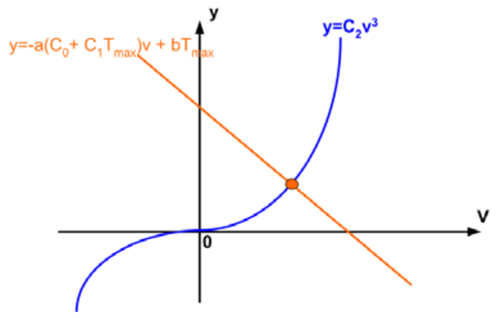


Figure 1. Since the slope for the linear function $y = -a((C_0 + C_1 T_{max})) \cdot v + bT_{max}$ is less than zero, there is only one cross point for function $y = -a((C_0 + C_1 T_{max})) \cdot V + bT_{max}$ and function $y = C_2 V^3$. So equation (14) has only one *real* root.

Also, it is much less costly to get the maximal speed (s_{max}) in a schedule (such as those generated by the approach in [23]) rather than to get the entire speed schedule for a periodic task set. Therefore this approach can be effectively used for the purpose of design space exploration.

Even though the feasibility condition formulated in Theorem 2 can be used for cases when the ending temperature of the first hyperperiod is higher than the initial temperature, this feasibility condition is still only a sufficient condition. In other word, when the maximal processor speed is higher than the maximal safe speed of the processor, the schedule may still be feasible under the maximal temperature. In what follows, we introduce the third feasibility condition, which is also a stronger condition and can be used to check the schedulability for these cases.

3.3. The necessary and sufficient condition

To guarantee the maximal temperature constraint, we need to make sure that this constraint is not violated at the end point of each hyperperiod and anywhere inside the hyperperiod. It helps then to identify the possible locations within the hyperperiod that this constraint may be violated. In what follows, we borrow the term of the *island interval* from Quan et al. [12] but change its definition slightly.

Definition 2: An interval $[t_i, t_j]$ in $\hat{S}(t)$ is called an *island interval* if the processor needs to run in a non-safe processor mode within this interval, or a *non-island interval* otherwise.

For an island interval, we have the following observation.

Lemma 2: Let $[t_1, t_2]$ be an island interval. If for any $t \in [t_1, t_2]$, we have $T(t) \leq T_{max}$, then we have $T(t) \leq T(t_2)$.

Lemma 2 can be proved by noticing that the temperature in an island interval monotonically increases and following the similar proof as that for Lemma 1. According to Lemma 2, the highest temperature for an island interval always occurs at its end, given that the maximal temperature constraint is not violated. Therefore, to verify if the temperature constraint is violated within a given hyperperiod, we only need to check temperatures at at the ends of all island intervals, plus the one at the end of the hyperperiod.

Our goal is to make sure that the temperature constraint is not violated during the entire life cycle when executing a periodic task set. Exhaustively checking temperature constraint for all hyperperiods is apparently impossible. In addition, it is not adequate to draw a conclusion that a schedule is feasible under the maximal temperature constraint simply because the temperature constraint is not violated within the first hyperperiod. So, if we want to check temperatures only at the first hyperperiod, additional constraints must be imposed. The following theorem provides such "additional" constraints.

Theorem 3: Let the ith interval in $\hat{S}(t)$ be $[t_i, t_{i+1}]$ and let its processor mode be k. Define A_i, B_i such that

$$A_i = A(k) = a(C_0 v_k + C_2 v_k^3), \quad (18)$$
$$B_i = B(k) = (b - aC_1 v_k). \quad (19)$$

Let $[t_{(j-1)}, t_j]$ be an arbitrary island interval in $\hat{S}(t)$, and let

$$K_j = exp(-B_0(t_1 - t_0) - ... - B_j(t_j - t_{(j-1)})) \quad (20)$$
$$K = exp(-B_0(t_1 - t_0) - ... - B_l(t_l - t_{(l-1)})) \quad (21)$$

where $t_l = L$. Then repeating $\hat{S}(t)$ later in the schedule, the temperature will never exceed T_{max} *iff* the following conditions hold:

- $0 \leq K < 1$;
- $T(L) \leq T_{max}(1 - K)$;
- $T(t_j) \leq T_{max} - \frac{T(L)}{1-K} K_j$.

Proof: See Figure 2. Let starting points for intervals in $\hat{S}(t)$ be $t_0, t_1, ..., t_{(l-1)}$, respectively. After repeating $\hat{S}(t)$, let the corresponding points during the second hyperperiod be $t'_0, t'_1, ..., t'_{(l-1)}$, correspondingly. Note that $t_0 = 0, t'_0 = t_l = L$ and $t'_l = 2L$.

According to equation (12), we have

$$T(t_1) = \frac{A_0}{B_0} + (T(t_0) - \frac{A_0}{B_0})e^{-B_0(t_1 - t_0)}$$

and

$$T(t'_1) = \frac{A_0}{B_0} + (T(t'_0) - \frac{A_0}{B_0})e^{-B_0(t'_1 - t'_0)}.$$

Since $(t'_1 - t'_0) = (t_1 - t_0)$, we have

$$T(t'_1) - T(t_1) = (T(t'_0) - T(t_0))e^{-B_0(t_1 - t_0)}. \quad (22)$$

Similarly, we have

$$T(t'_2) - T(t_2) = (T(t'_1) - T(t_1))e^{-B_0(t_1 - t_0) - B_1(t_2 - t_1)},$$

$$...$$

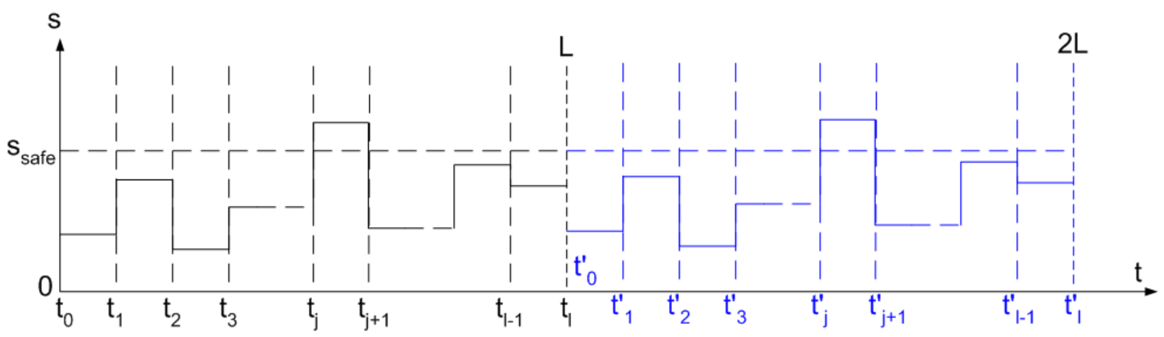


Figure 2. A speed schedule within 2 hyperperiods.

Therefore, we have

$$\begin{aligned} & T(2L) - T(L) \\ = \; & T(t'_l) - T(t_l) \\ = \; & (T(L) - T(0))e^{-B_0(t_1-t_0)-\ldots-B_{l-1}(t_l-t_{(l-1)})} \\ = \; & (T(L) - T(0))K. \end{aligned} \tag{23}$$

In the same way, we can see that

$$\begin{aligned} T(3L) - T(2L) &= (T(2L) - T(L))K && (24) \\ T(4L) - T(3L) &= (T(3L) - T(2L))K && (25) \\ &\ldots \end{aligned}$$

Therefore, $T(L) - T(0), T(2L) - T(L), T(3L) - T(2L), ..., T(qL) - T((q-1)L)$ form a geometric series and we have

$$T(qL) = T(0) + \frac{(T(L) - T(0))(1 - K^q)}{1 - K}. \tag{26}$$

Since $0 \le K < 1$, as $q \to \infty$, we have

$$\lim_{q\to\infty} T(qL) = T(0) + \frac{(T(L) - T(0))}{1 - K}. \tag{27}$$

After T_0 is calibrated to 0, $T(qL) \le T_{max}$ if and only if

$$T(L) \le T_{max}(1 - K). \tag{28}$$

We now need to make sure that the maximal temperature constraint is not violated in any island interval. Let $[t_{j-1}, t_j]$ be an arbitrary island interval in $\hat{S}(t)$. Follow the same procedure as stated above, we have

$$\begin{aligned} & T(L + t_j) - T(t_j) \\ = \; & (T(L) - T(0))e^{-B_0(t_1-t_0)-\ldots-B_{j-1}(t_j-t_{j-1})} \\ = \; & (T(L) - T(0))K_j. \end{aligned}$$

Similarly, we have

$$\begin{aligned} T(2L + t_j) - T(L + t_j) &= (T(2L) - T(L))K_j, \\ T(3L + t_j) - T(2L + t_j) &= (T(3L) - T(2L))K_j, \\ &\ldots \\ T(qL + t_j) - T((q-1)L + t_j) &= (T(qL) - \\ & \quad T((q-1)L))K_j, \end{aligned}$$

Add all above questions together, we have

$$T(qL + t_j) - T(t_j) = (T(qL) - T(0))K_j. \tag{29}$$

Similarly, since $0 \le K < 1$, as $q \to \infty$, with equation (26), we can get

$$T(qL + t_j) = T(t_j) + \frac{(T(L) - T(0))}{1 - K}K_j. \tag{30}$$

So, after T_0 is calibrated to 0, $T(qL + t_j) \le T_{max}$ if and only if

$$T(t_j) \le T_{max} - \frac{T(L)}{1 - K}K_j. \tag{31}$$

□

Note that, after the speed schedule $\hat{S}(t)$ is defined, K and K_j are well defined. We then can check temperatures at the end of first hyperperiod as well as those at the end of island intervals. $\hat{S}(t)$ is a feasible schedule if the three conditions in Theorem 3 hold.

Further discussions

From Theorem 3 and its proof, we have a number of interesting observations. Corollary 1, for example, is a straightforward conclusion from proof of Theorem 3.

Corollary 1: If $K > 1$ and $T(L) > T(0)$, the processor temperature will run away and reach infinity.

Corollary 1 can be easily proved from equation (26). When $K > 1$,

$$\begin{aligned} \lim_{q\to\infty} T(qL) &= \lim_{q\to\infty}(T(0) + \frac{(T(L) - T(0))(1 - K^q)}{1 - K}) \\ &= \infty. \end{aligned}$$

This implies that the heat generated by the processor exceeds its cooling capability, and the temperature continues to rise until the system breaks down.

On the other hand, when $0 \leq K < 1$, the processor temperature will eventually enter a *stable status* if the system does not break down before that. The *stable status* is formally defined as follows.

Definition 3: Assume a processor is running a periodic schedule $\hat{S}(t)$ with period L, the processor temperature is called to be in a *stable status* if for a given threshold, i.e. $0 < \varepsilon << 1$,

$$|T((i+1)L) - T(iL)| < \varepsilon, \tag{32}$$

where $i \geq 0, i \in Z$.
When the processor temperature enters the stable status, the temperature profile does not change much from one hyperperiod to another hyperperiod. Also, from the proof of Theorem 3, the temperature when the processor is in its stable status can be analytically formulated as follows.

Lemma 3: Assume a processor is running a periodic speed schedule $\hat{S}(t)$ with period L. Let $T(0)$ and $T(L)$ be temperatures at $t = 0$ and L, respectively. Then when the processor temperature reaches its stable status, the temperature at the starting (or ending) point of a period, denoted as $T(L')$, can be formulated as

$$T(L') = T(0) + \frac{(T(L) - T(0))}{1 - K}. \tag{33}$$

Note that similar lemmas can also be developed to calculate the temperature at each specific scheduling point, based on equation (30) in the proof of Theorem 3.

4. Experimental results

In the previous section, we present three feasibility testing methods (Theorem 1, 2, and 3) to verify if a feasible schedule developed within the first hyperperiod is globally feasible or not under the given maximal temperature constraint. The first two, i.e. Theorem 1 and 2, are sufficient conditions, and the third one, i.e. Theorem 3, is a more elaborated necessary and sufficient condition. All three methods take the leakage/tempearture dependency into account based on the processor and power model described in Section 2. In this section, we use experiments to test our proposed processor and power model, and study the effectiveness of these three schedulability analysis techniques.

We built our processor model based on the work by Liao et al. [4] for a processor using 65nm technology. Liao et al. developed an analytical formula (equation (6)) that can estimate the leakage current with less than 1% error. We used the same formula to compute the leakage currents for temperature from 40^oC to 110^oC with step size of 10^oC, which were used to determine curve fitting constants C_0 and C_1 in equation (7). To compare with the results presented in [4], we picked only two active modes (with supply voltages being 0.85V and 1.05V) and one shutdown mode for the processor. Figure 3a compares our linear approximation results (equation(7)) and the results via the accurate leakage estimation method (equation (6)). The relative errors are shown in Figure 3b. As we can see from Figure 3a and Figure 3b, the leakage current calculated using the linear approximation method is very close to that by a much more complex method (equation(6)), with error less than 5% for 0.85V and less than 3% for 1.05V.

Table 1. Processor parameters

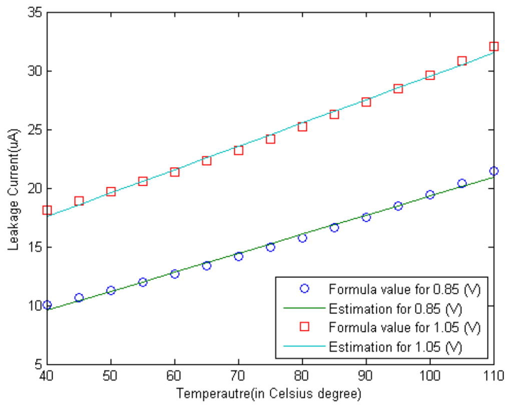
$V_{dd}(V)$	C_0	C_1	C_2	Frequency
0.00	0.0	0.0	0.0	0.0
0.85	3.0973	0.1621	15.9	0.8513
1.05	9.6375	0.1988	15.9	1.0

As indicated in [4], the circuit delay varies with both temperature and supply voltage. We set the frequency of the processor such that it can accommodate the longest delay at the highest temperature (110^oC) based on the related formula given in [4]. To obtain the leakage power consumption, the total number of gates, i.e. N_{gate} in equation (5), was set to be 10^6. The dynamic power consumption (and thus constant C_2) was determined based on the experimental results reported in [4].

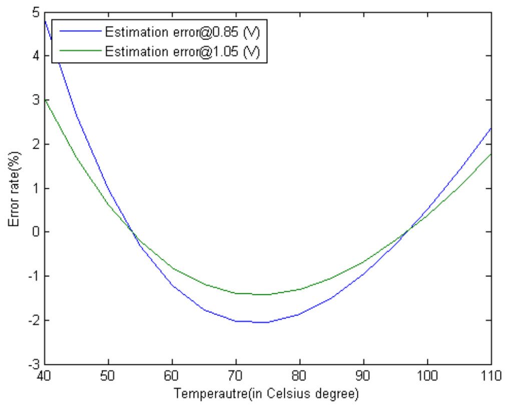
Table 1 lists the processor supply voltages, curve-fitting constants, and frequencies, which are normalized to the maximal one. For comparison purpose, we also consider another processor model in our experiments. The only difference of this model is that it assumes a constant leakage (i.e. the one under the ambient temperature). For the thermal constants, we selected $Rth = 0.8K/W$, $Cth = 340J/K$ [2], and the ambient temperature was set to 25^oC.

We next used the processor models developed above to study the feasibility analysis methods proposed in this paper. Four feasibility checking methods were studied. The first one (namely **EndCheck**), based on Theorem 1 checks if the temperature at the end of the hyperperiod is no more than the initial temperature. The second one (namely **SafeCheck**) applies Theorem 2 and uses the processor safe speed to check the feasibility. The third one (namely **IslandCheck**) employs Theorem 3, checking temperatures at the ending points of the first hyperperiod and all island intervals in the first hyperperiod. The fourth one (namely **ConstLeak**) is proposed in [12] that does not take the leakage temperature dependency into consideration.

The real time tasks were randomly generated with periods distributed evenly in range $[1000, 5000]$ seconds. Deadlines were determined by multiplying periods with a constant, called the *deadline-period ratio*. The execution time for each task was also randomly generated, which is evenly distributed between 1 and its deadline. The feasible speed schedules, generated based on the optimal method to mini-



(a) The linear approximation.



(b) The relative errors.

Figure 3. The linear approximation of the leakage current and the relative error for 65nm technology.

mize the dynamic energy [23], were used as our test cases. Since the approach in [23] assumes a processor model with continuously variable speed, we always rounded up a processor speed to the next higher available one in our experiments.

In our first set of experiments, we fixed the deadline-period ratio at 0.3 (i.e. *DPratio*=0.3) and varied the peak temperature constraint from 40^oC to 110^oC, with step size of 1^oC. For each peak temperature, we generated 100 test cases that can satisfy deadlines if the temperature factor is not taken into consideration. All these test cases were then tested using the four methods stated above and the numbers of schedulable task sets are shown in Figure 4.

Figure 4 shows clearly the significant impacts of peak temperature requirement to the schedule's feasibility. Note that Figure 4 only presents the results for peak temperature range in $[40,55]^oC$. This is because, in our experiments, for a peak temperature higher than 53^oC, all 100 schedules randomly generated as above can satisfy the temperature constraints based on **IslandCheck**, **SafeCheck**, and **ConstLeak**. When the peak temperature constraint getting tighter, however, the feasibility drops quickly. In Figure 4, when the peak temperature is set to 45^oC, more than 40% of the original feasible schedules become infeasible.

Figure 4 also reveals the applicability and effectiveness of each feasibility analysis method. We can see that, without considering the leakage/temperature dependency, **ConstLeak** can be overoptimistic in predicting the schedulability of a task set. Note that, when the peak temperature set to 49^oC, more than 30% of feasible schedules based on **ConstLeak** are in fact infeasible based on corresponding results by **IslandCheck**. In addition, it is not surprising to see in Figure 4 that none of the task sets can be predicted as feasible using **EndCheck**. This is because that, starting from the ambient temperature, the processor temperature always increases to one that is above the ambient temperature after

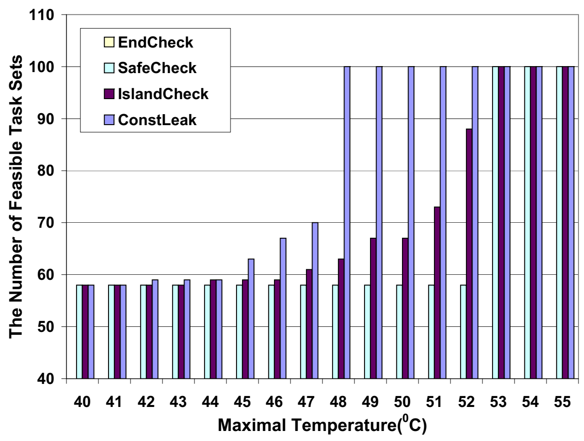


Figure 4. The numbers of predicted feasible task sets by four approaches under different maximal temperature constraints (deadline-period ratio = 0.3). The results for maximal temperature higher than 55^oC are the same as that of 55^oC and thus omitted.

executing tasks, given the thermal settings stated before. Moreover, we can see that **SafeCheck** is overpessimistic in predicting the feasibility for a schedule. This is because a schedule occasionally using a speed higher than the processor safe speed can still reach a temperature lower than the required maximal temperature. In Figure 4, when the maximal temperature is set to be 52^oC, about 34% of the feasible task sets cannot be verified properly by **SafeCheck**. When the given maximal temperature becomes very high, all processor running modes become safe modes, and thus **SafeCheck** obtains the same results as that by **IslandCheck** in Figure 4.

In our second set of experiments, we fixed the maximal temperature at 50^oC, and varied the deadline-period ratio. The deadline-period ratio was varied from 0.1 to 0.9 with

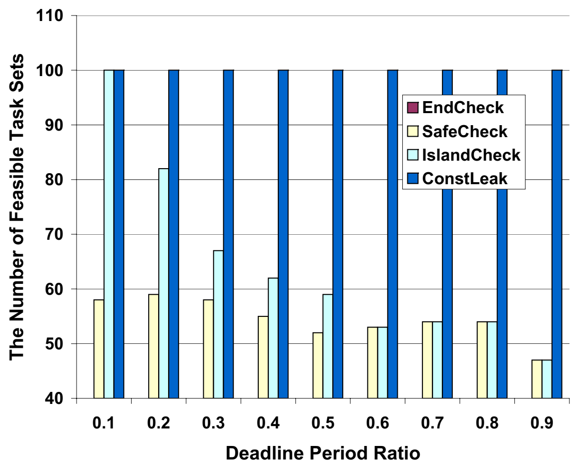


Figure 5. The numbers of predicted feasible task sets by four approaches under different deadline-period ratio. ($T_{max} = 50^oC$).

step size of 0.1. The numbers of feasible schedules determined by each method are shown in Figure 5.

While similar conclusions with regard to the feasibility analysis effectiveness and applicability can be drawn from Figure 5, the results further highlight the importance of taking the leakage/temperature dependency into consideration. Note that as the deadline period ratio increases, the intervals within which the processor runs in an active mode become longer. Therefore, the leakage impacts become more significant, and the temperature grows faster. **ConstLeak** considers only constant leakage but not the intertwine between leakage and temperature, as that by **SafeCheck** and **IslandCheck**. As a result, while we observe no difference in terms of numbers of feasible schedules with **ConstLeak** as the deadline period ratio increases, the results based on both **SafeCheck** and **IslandCheck** are significantly different. When the deadline period ratio is 0.9, **ConstLeak** mispredicts the feasibility for 52% of the task sets, as shown in Figure 5. These experimental results clearly show the significant role that the leakage/temperature relationship plays for processors using the sub-micron technology.

5. Summary and the future work

As semiconductor technology continues to evolve toward the deep sub-micron era, the leakage in the CMOS circuit and its interaction with temperature become a too significant factor to be ignored.

In this paper, we make a number of contributions: (*i*) we introduce a novel leakage mode with leakage/temperature dependency taken into consideration. This model leverages the circuit and micro-architecture level leakage model for ease of system level analysis. (*ii*) Based on the proposed power model, we present three feasibility analysis techniques to determine if a period task set can meet deadlines under a given maximal temperature constraint. (*iii*) Our experimental results, based on technical parameters derived from a processor using 65nm technology, show that our leakage current model have a relative error less than 5%. In addition, our experimental results on the feasibility analysis demonstrate the effectiveness of our methods and clearly highlight the importance to deal with the impacts of leakage/temperature relationship.

The feasibility analysis techniques presented in this paper establish a solid basis for further extensive research on leakage-aware thermal analysis for real-time systems. With these feasibility analysis techniques, we can determine if a given schedule can meet both timing and maximal temperature constraints. However, it is by no means a trivial effort to develop such a schedule. In our current work, for simplicity, we use the well-known method (i.e [23]) that can minimize the dynamic energy. Such a schedule does not necessarily minimize the overall energy consumption when considering the leakage power consumption in the overall power consumption [24], let alone the leakage/temperature dependency. Furthermore, previous work (such as [6], [12]) has clearly shown that an energy minimization scheduling technique is not necessarily the optimal solution in terms of scheduling real-time tasks under thermal constraints. It is therefore imperative that new scheduling techniques be developed under the maximal temperature constraints. In addition, to develop real-time scheduling methods that can minimize the energy consumption under the maximal temperature constraint with the consideration of the leakage and temperature interplay is an interesting problem and calls for further research.

Acknowledgment

The authors would also like to thank the anonymous reviewers for their constructive remarks that helped improve the quality of this paper. This work was supported by the NSF under Career Award CNS-0545913 and DUE-06333641.

References

[1] S. Borkar, "Thousand core chips: a technology perspective," in *DAC*. New York, NY, USA: ACM, 2007, pp. 746–749.

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

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

[4] 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*, vol. 24, no. 7, pp. 1042 – 1053, 2005.

[5] 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.

[6] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *Journal of the ACM*, vol. 54, no. 1, pp. 1–39, 2007.

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

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

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

[10] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *ICCAD*. Piscataway, NJ, USA: IEEE Press, 2007, pp. 281–288.

[11] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," in *DATE '08: Proceedings of the conference on Design, automation and test in Europe*. New York, NY, USA: ACM, 2008, pp. 288–293.

[12] 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.

[13] L. He, W. Liao, and M. R. Stan, "System level leakage reduction considering the interdependence of temperature and leakage," in *DAC*. New York, NY, USA: ACM, 2004, pp. 12–17.

[14] L. Yuan, S. Leventhal, and G. Qu, "Temperature-aware leakage minimization technique for real-time systems," in *ICCAD*. New York, NY, USA: ACM, 2006, pp. 761–764.

[15] 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*, vol. 0, pp. 660–670, 2007.

[16] 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*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 204–209.

[17] 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*. New York, NY, USA: ACM, 2007, pp. 111–116.

[18] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management," in *HPCA '02: Proceedings of the 8th International Symposium on High-Performance Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 2002, p. 17.

[19] Y. Liu, R. P. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *DATE '07: Proceedings of the conference on Design, automation and test in Europe*. San Jose, CA, USA: EDA Consortium, 2007, pp. 1526–1531.

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

[21] P. J. Nahin, *The Story of* $\sqrt{-1}$. Boston: Princeton University Press, 1998.

[22] Wikepdeia, "Cubic function," *http://en.wikipedia.org/wiki/Cubic_equation*, 2008.

[23] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *FOCS*, 1995, pp. 374–382.

[24] G. Quan and L. Niu, "Fixed priority scheduling for reducing overall energy on variable voltage processors," *RTSS'04*, Dec 2004.