Sustainable Computing: Informatics and Systems 3 (2013) 274–285

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Thermal aware overall energy minimization scheduling for hard real-time systems☆

Huang Huang, Ming Fan*, Gang Quan

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

ARTICLE INFO

Article history:
Received 30 January 2013
Received in revised form 3 May 2013
Accepted 25 June 2013

Keywords:
Energy minimization
Scheduling
Dynamic voltage scaling
Leakage/temperature dependency

ABSTRACT

As the semiconductor technology proceeds into the deep sub-micron era, the leakage and its dependency with the temperature become critical in dealing with power/energy minimization problems. In this paper, we study the problem on how to schedule a hard real-time system to achieve the minimal overall energy, including both dynamic and leakage energy consumption. We first develop an energy estimation method that can be used to accurately and efficiently calculate the overall energy consumption of a candidate schedule. Based on the proposed energy equation, we then develop two scheduling methods, i.e. an off-line and an on-line method, to minimize the overall energy consumption for real-time systems. Our experimental results demonstrate that the proposed energy estimation method can achieve up to two orders of magnitude speedup compared with an existing approach while maintaining good accuracy. In addition, with a large number of different test cases, both our off-line and on-line approaches can significantly outperform existing related works.

Published by Elsevier Inc.

1. Introduction

The human's appetite for high-performance computing systems has driven the semiconductor technology into the deep sub-micron (DSM) era. The continued shrinking of the transistor size, together with increasingly complicated circuit architectures, have resulted in a significant increase of the power density. It is predicted that in a matter of a few years, the number of transistors that can be integrated into one 300 mm^2 die will reach 100 billion and its power consumption can be as high as 300 W [1]. One of the immediate consequences is the energy consumption issue which has posed tremendous challenge to digital system designers of both embedded and server platforms. For battery-driven embedded systems, given the fact that the growing of battery technology is far lagging behind the development of the semiconductor technology, unless more advanced low-power or power-aware design techniques are applied, the applicability of such devices will be severely limited. For power-rich server systems, on the other hand, the high energy consumption directly leads to high temperature, which not only increases the packaging/cooling costs, degrades the reliability/performance of the system, but also significantly increases the leakage power consumption due to the strong leakage/temperature dependency in the DSM domain. Therefore, power/energy minimization techniques are urgently demanded at every design abstraction levels.

Early research efforts are mainly focusing on how to reduce the dynamic energy consumption of a system. Dynamic voltage scaling (DVS) is well known to be one of the most effective dynamic energy reduction techniques. Due to the convex relationship between the dynamic power consumption and the processor speed level, a quadratic dynamic power reduction can be achieved at the expense of a linear performance reduction. However, as the leakage becomes more and more prominent in the DSM domain, the leakage energy component as well as its interdependence with the temperature have to be properly taken into consideration. As evidenced in [2], the leakage power will increase by 38% when chip temperature raising from 65 °C to 110 °C. In fact, thermal/temperature constraint is becoming an increasingly critical issue in the computing system deign [3]. Due to the leakage/temperature dependency, the leakage energy consumption is as important as the dynamic energy consumption. Therefore, an energy optimization design technique can become ineffective or inefficient if the interdependency between leakage and temperature is not properly addressed.

In this paper, we are interested in studying the problem of energy minimization for hard real-time scheduling on single-core systems by considering the dependency between leakage and temperature. The complexity of the problem lies in the fact that leakage energy consumption depends on both supply voltage level and temperature. For instance, two identical speed schedules may result in different energy consumptions simply because their initial temperatures are different. Therefore, how to accurately and

☆ This work is supported in part by NSF under projects CNS-0969013, CNS-0917021 and CNS-1018108.

* Corresponding author. Tel.: +1 7864731464.
E-mail addresses: hhuan001@fiu.edu (H. Huang), mfan001@fiu.edu, fanming0626@gmail.com (M. Fan), gaquan@fiu.edu (G. Quan).

2210-5379/$ – see front matter. Published by Elsevier Inc.
http://dx.doi.org/10.1016/j.suscom.2013.06.002

efficiently estimate the energy consumption of various candidate speed schedules is the key to solve this problem.

We summarize the major contributions of this work below:

- First, we develop an effective closed-form energy estimation method with the leakage/temperature dependency taken into consideration.
- Secondly, based on the proposed energy estimation method, we further develop two scheduling algorithms to minimize the overall energy consumption. The first algorithm is an off-line algorithm, targeting at a real-time system consisting of a set of periodic tasks with the same period and deadline. The second algorithm is an on-line algorithm, which is intended for more general real-time systems consisting of multiple sporadic tasks.
- Finally, we conduct extensive experiments to evaluate the performance of our proposed energy estimation method as well as the online/offline energy efficient scheduling algorithms. Our experimental results show that the proposed energy estimation method can achieve a speedup up to two orders of magnitude compared with the existing approach while maintaining high accuracy. Our experimental results also show that the proposed scheduling algorithms consistently outperform the existing approaches in terms of energy reduction.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 introduces the system models used in this paper and Section 4 provides a motivational example. In Section 5 we show how to derive the proposed energy estimation equations, based on which we present our overall energy minimization scheduling algorithms in Section 6. Experimental results are discussed in Section 7. Section 8 concludes this paper.

2. Related work

The power/energy minimization problem has been researched extensively for a couple of decades [4–10]. Early research efforts are mainly focusing on reducing the dynamic power/energy, i.e. the predominate component of the overall power/energy consumption. By taking advantage of the convex relationship between the dynamic power and supply voltage, many technique (e.g. [4,5]) strive to lower down the processor supply voltage and working frequency. As the leakage becomes more prominent, a few techniques are proposed to minimize the overall energy, which consists of both dynamic and leakage energy. For example, Jejurikar et al. [6,7] proposed a scheduling technique, based on the so called *critical speed* to balance the dynamic and leakage energy consumption, with the goal to minimize the overall energy consumption. These approaches assume that the leakage current is constant. However, as semiconductor technology ventures into the DSM domain, the leakage/temperature dependency becomes too significant to be ignored. As shown in [11], a power/thermal aware design technique simply becomes ineffective if the interdependence of leakage and temperature is not properly addressed.

Recently, there are a number of works [3,12–15] published on minimizing the overall energy consumption with the leakage/temperature dependency taken into consideration. Specifically, Yang et al. [15] develop a quadratic leakage model to simplify the leakage/temperature dependency. Based on this model, they proposed a *Pattern-based* approach to schedule a periodic task and reduce the energy consumption when a processor reaches its thermal steady state. This approach mainly focuses on processors without *DVS* capability. It reduces the energy consumption at the temperature steady state by periodically switching the processor between the active and dormant mode. In our research, we assume that the leakage depends not only on the temperature but also on the supply voltage as well. Bao et al. [12] employ the circuit-level leakage power model (piece-wise linear (PWL) leakage model) to model the interdependency between the leakage, temperature and the supply voltage. They propose a method to distribute idle intervals judiciously when scheduling a task graph such that the temperature of the processor can be effectively cooled down to reduce the leakage energy consumption. In our approach, we use a different leakage model, i.e. same to the one used in [16]. Our energy estimation method is derived based on this model. It forms the basis of the proposed scheduling algorithms. Note that, all these scheduling algorithms are off-line scheduling algorithms.

While a schedule developed off-line may be more effective to provide certain guarantee, an on-line schedule can be more effective in saving energy. Moreover, for a more complicated real-time system, such as a real-time system consisting of multiple sporadic tasks scheduled by the *EDF* policy, the temperature rarely reaches the steady state. In such a scenario, an on-line scheduling algorithm becomes more desirable.

Several work were published to address the on-line energy optimization problem. For example, Yuan et al. [17] introduce a simple heuristic to turn on or off a processor based on its workload and temperature. This approach directly employs the circuit-level leakage model [2,18] to capture the interdependence of leakage and temperature. However, as a result of the complexity from the non-linear and high order terms of this model, the approach in [17] can only be applied for soft real-time systems. Bao et al. [13] propose an on-line temperature aware *DVS* technique to minimize the energy consumption when scheduling a task graph. This approach requires extensive off-line static analysis to generate a lookup table (LUT) for each task under various temperature conditions, so that the scheduler can adjust the processor speed accordingly to minimize energy consumption on-line. There are also other on-line thermal aware approaches such as [19–23]. They focus more on peak temperature reduction or throughput maximization under a given peak temperature constraint.

Our on-line scheduling algorithm seeks to minimize the overall energy consumption of a hard real-time system consisting of a set of aperiodic tasks, scheduled according to the *EDF* policy. To the best of our knowledge, we are not aware of any other thermal-aware scheduling technique that can guarantee the hard deadline for an aperiodic task set.

3. Preliminaries

3.1. System models

In this paper, we choose the processor model as single-core systems. For the real-time applications, we first consider a set of hard periodic tasks with the same period, or equivalently, one single consolidated task with the period equal to its deadline. We further extend our research to a more general real-time application model, in which task set (Γ) consists of n sporadic tasks, denoted as $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ and scheduled according to the *EDF* policy. For each task $\tau_i \in \Gamma$, we have $\tau_i \equiv \{a_i, c_i, d_i, p_i\}$. Specifically, a_i represents the arrival time of τ_i. c_i is the worst case execution time (*WCET*) of τ_i under the maximum clock frequency. d_i is the relative deadline of τ_i with respect to its arriving time. p_i is the minimum inter-arrival time between any two consecutive jobs of τ_i. In this paper, we refer p_i as the period of τ_i.

The thermal model used in this paper is similar to the one that has been used in similar research (e.g. [15,23–26]). Specifically, $T(t)$ and T_{amb} are the chip temperature and the ambient temperature, respectively. $P(t)$ denotes the power consumption (in W) at time t, and R, C are the thermal resistance (in °C/W) and thermal capacitance (in J/°C), respectively. Then, we have

$(RC(dT(t)/dt)) = RP(t) + (T(t) - T_{amb})$. We can scale T such that T_{amb} is zero and then we have

$$\frac{dT(t)}{dt} = aP(t) - bT(t), \quad (1)$$

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

We assume that the processor can provide n different supply voltage levels with each level associated with a different clock frequency. We further assume that during each speed transition, an extra energy overhead of E_{sw} will be consumed and a timing penalty of τ will be incurred during which the clock is halted that no computation can take place.

The overall power consumption of a processor is composed of two parts: the dynamic power P_{dyn} and leakage power P_{leak}. The dynamic power consumption is independent to the temperature and can be formulated as $P_{dyn} = C_{load} f v_k^2$, where C_{load} is the equivalent parasitic capacitance, f is the clock frequency and v is the supply voltage. Since the working frequency is in proportion to the supply voltage level, we can further assume $P_{dyn} = C_2 v_k^3$ [27], where v_k is the kth supply voltage level and C_2 is a constant. The leakage power can be calculated as $P_{leak} = N_{gate} \cdot I_{leak} \cdot v_k$, where N_{gate} represents the number of gates and I_{leak} is the leakage current, which can be formulated by a non-linear exponential Eq. (2)

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

where I_s is the leakage current at certain reference temperature and supply voltage, T is the operating temperature, $\mathcal{A}$, $\mathcal{B}$, α, β, γ, δ are empirically determined technology constants.

As reported in [28,29] that the leakage current changes super linearly with the temperature and by using linear approximation method to model the leakage/temperature dependency, reasonable accuracy can be maintained. Thus, the leakage power for the processor running in mode k can be effectively estimated as [24]

$$P_{leak}(k) = C_0(k)v_k + C_1(k)Tv_k, \quad (3)$$

where $C_0(k)$ and $C_1(k)$ are constants. As we can see from Eq. (3), the leakage power depends on both the supply voltage and temperature. Then, total power consumption at the kth processor mode is

$$P(k) = C_0(k)v_k + C_1(k) \cdot Tv_k + C_2 v_k^3. \quad (4)$$

Accordingly, the temperature dynamic at the speed level k can be formulated as

$$\frac{dT(t)}{dt} = A(k) - B(k)T(t), \quad (5)$$

where $A(k) = a(C_0(k)v_k + C_2 v_k^3)$ and $B(k) = b - aC_1(k)v_k$. Hence, for a given time interval $[t_0, t_e]$, if the initial temperature is T_0, by solving Eq. (5), the ending temperature can be formulated as below:

$$\begin{aligned} T_e &= \frac{A(k)}{B(k)} + \left(T_0 - \frac{A(k)}{B(k)}\right) e^{-B(k)(t_e - t_0)}, \\ &= G(k) + (T_0 - G(k))e^{-B(k)(t_e - t_0)}. \end{aligned} \quad (6)$$

In Eq. (6), by letting $(t_e - t_0) \rightarrow \infty$, we have t_e equals $G(k)$ which is called the steady state temperature of the kth speed level. For an initial temperature T_0, if the steady state temperature of a processor speed level is greater than T_0, the temperature will increase, or decrease otherwise.

4. Motivational examples

As leakage varies with temperature, one key challenge in our research is to calculate the overall energy consumption of a speed schedule accurately and efficiently.

Let us consider a speed schedule shown in Fig. 1(a). The speed schedule, $S \equiv [A, B, C, D]$, consists of four intervals with each characterized by a speed level (s_1 to s_4) and a duration $((t_1 - t_0) \ldots (t_4 - t_3))$.

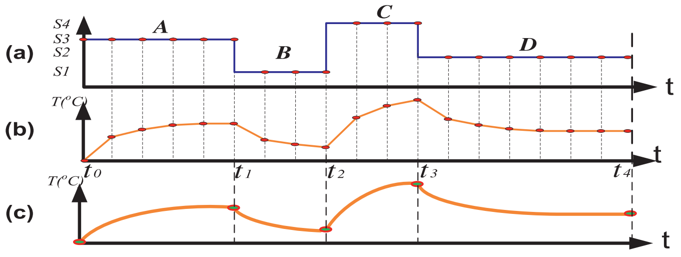


Fig. 1. A speed schedule and its temperature curve.

Since the leakage energy used to take only a small portion of the overall energy consumption, some early works [4–7,30] simply ignore the leakage or assume it to be constant. However, as shown in [11], these energy models can result in large estimation errors in the DSM domain, where leakage/temperature dependency is prominent.

An alternative method, similar to that in [31], is to divide an interval into a number of small sub-intervals (as shown in Fig. 1(b)). Within each sub-interval, one can assume that the power consumption remains constant. To calculate the energy consumption of S, we can divide each interval into multiple sub-intervals with equal length, i.e. ρ. Then, the energy consumption of the *first* sub-interval, i.e. from t_0 to $t_0 + \rho$, can be calculated in the following steps,

$$\begin{aligned} E_{dyn} &= C_2 \cdot {S_3}^3 \cdot \rho, \\ E_{leak} &= s_3 \cdot \rho \cdot I_s \cdot (\mathcal{A} \cdot T_0^2 \cdot e^{((\alpha \cdot s_3 + \beta)/T_0)} + \mathcal{B} \cdot e^{(\gamma \cdot s_3 + \delta)}), \\ E_{total} &= E_{dyn} + E_{leak}. \end{aligned} \quad (7)$$

The dynamic energy component, which is independent to the temperature, is calculated based on our power model introduced in Section 3, while the leakage energy consumption is obtained by using the circuit-level model (Eq. (2)). The temperature within this sub-interval is assumed to be constant, which is the initial temperature T_0. By adding these two components, we can obtain the overall energy consumption of the *first* sub-interval. With Eq. (2), this method can achieve relatively accurate energy estimation as long as the sub-interval (ρ) is sufficiently small. However, the computational cost can be very sensitive to the accuracy of this approach.

Assume that we need to compare the energy consumption of the schedule S with another schedule $S' \equiv [D, C, B, A]$, and find the better one in terms of energy reduction. Even though S and S' have identical dynamic energy consumption, the leakage energy consumptions are totally different simply because the temperature curves produced by running S and S' are totally different. Note that, the overall energy consumptions are different even for the same schedule with different starting temperatures. The computational cost would be prohibitively high if we use this method to calculate the energy consumption on-line. Evidently, an accurate energy estimation method with low computation cost is highly desirable.

In the sections that follow, we first introduce our method to calculate the energy consumption for a schedule. We then present an off-line and an on-line scheduling method to minimize the energy consumption.

5. Energy estimation equation

From the motivational example above, we can see that it is critical to develop a method that can rapidly and accurately estimate the energy consumption of a given schedule. As one of the major components of overall energy consumption, the leakage energy as well as its strong dependency with temperature have made the energy estimation a non-trivial task. In this section, we first discuss how to calculate the energy consumption for a single speed interval. Then, we discuss how to calculate the energy consumption for a periodic schedule.

5.1. Energy calculation for single speed interval

Without losing of generality, let us consider running a processor in mode k, i.e. applying the kth speed level, during interval $[t_0, t_e]$. Based on our thermal model, the temperature varies following Eq. (1). By integrating both sides of Eq. (1), we have

$$\begin{aligned}\int_{t_0}^{t_e}\frac{dT(t)}{dt}\cdot dt &= a\cdot\int_{t_0}^{t_e}P(t)dt-b\cdot\int_{t_0}^{t_e}T(t)dt,\\ T(t_e)-T(t_0) &= a\cdot E(t_0,t_e)-b\cdot\int_{t_0}^{t_e}T(t)dt,\\ E(t_0,t_e) &= \frac{1}{a}\cdot\left(T(t_e)-T(t_0)+b\cdot\int_{t_0}^{t_e}T(t)dt\right).\end{aligned}\tag{8}$$

where $T(t_0)$ and $T(t_e)$ denote the starting and ending temperature of interval $[t_0, t_e]$ and $E(t_0, t_e)$ represents the energy consumed during this interval. Note that in Eq. (8), $T(t_e)$ can be analytically calculated from Eq. (6) that

$$T(t_e)=G(k)+(T(t_0)-G(k))e^{-B(k)(t_e-t_0)}.\tag{9}$$

Also, the integral term $\int_{t_0}^{t_e}T(t)dt$ can be expressed as,

$$\begin{aligned}\int_{t_0}^{t_e}T(t)dt &= \int_{t_0}^{t_e}G(k)dt+\int_{t_0}^{t_e}(T_0-G(k))\cdot e^{-B(k)(t-t_0)}dt\\ &= G(k)\cdot(t_e-t_0)+\frac{1}{B(k)}\cdot(G(k)-T_0)(e^{-B(k)(t_e-t_0)}-1).\end{aligned}\tag{10}$$

Replacing $T(t_e)$ and the integral term $\int_{t_0}^{t_e}T(t)dt$ in Eq. (8) with Eq. (9) and (10), the overall energy consumed within the interval $[t_0, t_e]$ can be formulated by a closed-form equation

$$\begin{aligned}E(t_0,t_e)=\frac{1}{a}\cdot(G(k)+(T(t_0)-G(k))e^{-B(k)(t_e-t_0)}-T_0+b\cdot(G(k)\\ \cdot(t_e-t_0)+\frac{1}{B(k)}\cdot(G(k)-T(t_0))(e^{-B(k)(t_e-t_0)}-1))).\end{aligned}\tag{11}$$

From Eq. (11), we can see that in order to obtain the energy consumption of a given interval, we only need to know the duration of the interval $(t_e - t_0)$, the initial temperature $(T(t_0))$ and the current processor speed level (k), which determines the value of $G(k)$ and $B(k)$, according to our system models.

With Eq. (11), we can quickly calculate the energy consumption of a given speed schedule. Recall the example shown in Fig. 1. Now, if applying Eq. (11), the overall energy consumption of the schedule S can be formulated as

$$E_{total}=E(t_0,t_1)+E(t_1,t_2)+E(t_2,t_3)+E(t_3,t_4).\tag{12}$$

It is not difficult to see that by applying Eq. (11), the proposed energy calculation method (Fig. 1(c)/Eq. (12)) has significantly lower computational cost than the energy estimation method in [31] (Fig. 1(b)/Eq. (7)). The number of steps required to estimate the energy consumption of the complete schedule is equal to the number of intervals the schedule has, i.e. *4* in this example. The computational overhead reduction is achieved from two aspects. First, by using the proposed closed-form energy equation, there is no need to divide one speed interval into several small sub-intervals. Second, for each evaluation interval, instead of calculating the dynamic and leakage energy separately, we can obtain the overall energy consumption in one step.

5.2. Energy calculation for periodic schedule

Now, let us consider the case if a speed schedule we are dealing with is periodic (as shown in Fig. 2) and we want to calculate the energy consumption when running this schedule for N periods. Before temperature reaches the steady state, the temperature curves at different periods are different. As a result, the (leakage) energy consumption within each period is also different. We can certainly employ the closed-from equation introduced above to calculate the total energy consumption interval by interval, which is far more efficient than using the approach in [31]. However, in this case, we still need to find the starting/ending temperature of each of the $4N$ intervals and calculate the corresponding integral individually. The computational costs can still be very high if N is large. In what follows, we introduce another energy estimation method that work more efficiently.

Consider the example shown in Fig. 2 that a periodic schedule $\tilde{S}\equiv[A, B, C, D]$ is running with an arbitrary starting temperature. Without losing of generality, we assume that the schedule $\tilde{S}$ repeats for a total number of N periods (or copies). We use $T(t_\beta^\alpha)$ to denote the starting temperature of the βth interval in the αth copy of the periodic schedule $\tilde{S}$. Also, in this example we assume that for interval A, B, C and D, the processor runs in the speed level S_3, S_1, S_4 and S_2, respectively and we have $S_4 > S_3 > S_2 > S_1$.

To obtain the overall energy consumption of $\tilde{S}$, we apply Eq. (8) to each speed segment. Then, we have

$$\begin{aligned}E(t_0^0,t_1^0) &= \frac{1}{a}\left(T(t_1^0)-T(t_0^0)+b\int_{t_0^0}^{t_1^0}T(t)dt\right)\\ E(t_1^0,t_2^0) &= \frac{1}{a}\left(T(t_2^0)-T(t_1^0)+b\int_{t_1^0}^{t_2^0}T(t)dt\right)\\ E(t_2^0,t_3^0) &= \frac{1}{a}\left(T(t_3^0)-T(t_2^0)+b\int_{t_2^0}^{t_3^0}T(t)dt\right)\\ E(t_3^0,t_0^1) &= \frac{1}{a}\left(T(t_0^1)-T(t_3^0)+b\int_{t_3^0}^{t_0^1}T(t)dt\right)\\ &\cdots\\ E(t_2^{N-1},t_3^{N-1}) &= \frac{1}{a}\left(T(t_3^{N-1})-T(t_2^{N-1})+b\int_{t_2^{N-1}}^{t_3^{N-1}}T(t)dt\right)\\ E(t_3^{N-1},t_0^{N}) &= \frac{1}{a}\left(T(t_0^{N})-T(t_3^{N-1})+b\int_{t_3^{N-1}}^{t_0^{N}}T(t)dt\right)\end{aligned}\tag{13}$$

Sum up the above $4N$ equations we get

$$\begin{aligned}E_{sum}=\frac{1}{a}\cdot\Bigg(T(t_0^N)-T(t_0^0)+b\cdot\Bigg(\sum_{x=0}^{N-1}\int_{t_0^x}^{t_1^x}T(t)dt+\sum_{x=0}^{N-1}\int_{t_1^x}^{t_2^x}T(t)dt\\ +\sum_{x=0}^{N-1}\int_{t_2^x}^{t_3^x}T(t)dt+\sum_{x=0}^{N-1}\int_{t_3^x}^{t_0^{x+1}}T(t)dt\Bigg)\Bigg),\end{aligned}\tag{14}$$

where E_{sum} is the total energy consumption that we want to find. $T(t_0^N)$ and $T(t_0^0)$ are the final and initial temperature of the periodic schedule $\tilde{S}$, respectively. Each summation term is corresponding to the summation of all N integral terms for interval A, B, C and D.

Recall that in Eq. (10), if the kth speed level is applied during an interval, the corresponding integral term can be analytically expressed. $G(k)$ and $B(k)$ are known *a priori* given a processor model,

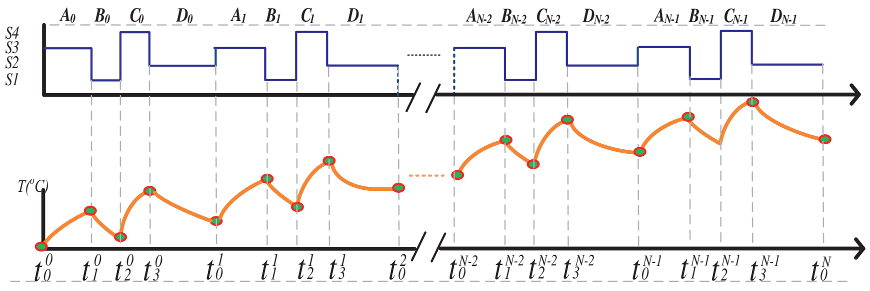


Fig. 2. A periodic schedule and its temperature curve.

and $T(t_0)$ is the initial temperature of the segment being evaluated. Similarly, from Eq. (14), it is not difficult to see that the overall energy consumption E_{sum} depends upon the starting temperature of each speed interval, i.e. $T(t_0^0)$, $T(t_1^0)$, $T(t_2^0)$, $T(t_3^0)\ldots T(t_2^{N-1})$ and $T(t_3^{N-1})$.

To obtain these values, one straightforward way is to keep track of the entire schedule $\tilde{S}$ and calculate the temperature at those specific time instants and then apply the proposed integral based method for all $4N$ intervals and sum them up. However, the complexity of this method can be extremely high if there are too many number of intervals. A more computational efficient method is to formulate the temperature values analytically. According to [16], the differences between the ending temperature and the starting temperature of each single copy of a periodic schedule form a geometric series. Specifically, we have

$$\begin{aligned} T(t_0^2)-T(t_0^1) &= (T(t_0^1)-T(t_0^0))\cdot K_0,\\ T(t_0^3)-T(t_0^2) &= (T(t_0^2)-T(t_0^1))\cdot K_0,\\ &\cdots\\ T(t_0^N)-T(t_0^{N-1}) &= (T(t_0^{N-1})-T(t_0^{N-2}))\cdot K_0, \end{aligned} \tag{15}$$

where $K_0=\exp(-B(3)(t_1^0-t_0^0)-B(1)(t_2^0-t_1^0)-B(4)(t_3^0-t_2^0)-B(2)(t_0^1-t_3^0))$.

The significance of this observation is that we can obtain the temperature information within the qth period of the periodic schedule $\tilde{S}$ based on those within the first copy. For example, the starting temperature of qth period of the schedule, i.e. $T(t_0^q)$, can be effectively calculated by

$$T(t_0^q)=T(t_0^0)+\frac{(T(t_0^1)-T(t_0^0))\cdot(1-K_0^q)}{1-K_0}. \tag{16}$$

Similarly, other temperature information within the qth period can also be formulated as[1]:

$$\begin{aligned} T(t_1^q) &= T(t_1^0)+\frac{(T(t_0^1)-T(t_0^0))(1-K_0^q)K_1}{1-K_0},\\ T(t_2^q) &= T(t_2^0)+\frac{(T(t_0^1)-T(t_0^0))(1-K_0^q)K_2}{1-K_0},\\ T(t_3^q) &= T(t_3^0)+\frac{(T(t_0^1)-T(t_0^0))(1-K_0^q)K_3}{1-K_0}, \end{aligned} \tag{17}$$

where

$$\begin{aligned} K_1 &= \exp(-B(1)(t_1^0-t_0^0)),\\ K_2 &= \exp(-B(4)(t_1^0-t_0^0)-B(1)(t_2^0-t_1^0)),\\ K_3 &= \exp(-B(2)(t_1^0-t_0^0)-B(1)(t_2^0-t_1^0)-B(2)(t_3^0-t_2^0)). \end{aligned} \tag{18}$$

By applying Eq. (16) and (17), we can see that the temperature at all speed transition instants can be obtained analytically after we get the temperature information within the first period of $\tilde{S}$, i.e. $T(t_0^0)$, $T(t_1^0)$, $T(t_2^0)$, $T(t_3^0)$ and $T(t_0^1)$.

Now let us first consider all "A" intervals in Fig. 2. The starting temperature of the qth "A" interval has already been formulated in Eq. (16). Therefore, the summation of the initial temperature of all "A" intervals can be expressed as

$$\sum_{q=0}^{N-1}T(t_0^q)=N\cdot T(t_0^0)+\frac{T(t_0^1)-T(t_0^0)}{1-K_0}\cdot\left(N-\frac{1-K_0^N}{1-K_0}\right) \tag{19}$$

Once the initial temperature of all "A" intervals are available, the summation term of the total N corresponding integrals (in Eq. (14)) can be formulated as

$$\sum_{x=0}^{N-1}\int_{t_0^x}^{t_1^x}T(t)dt=N\cdot G(3)\cdot((t_1^0-t_0^0)+\beta_A)-\beta_A\cdot\sum_{q=0}^{N-1}T(t_0^q), \tag{20}$$

where $\beta_A=1/B(3)\cdot\exp(-B(3)(t_1^0-t_0^0)-1)$.

Similarly, based on equation group (17), the summation of the initial temperature of all "B" intervals, "C" intervals and "D" intervals are

$$\begin{aligned} \sum_{q=0}^{N-1}T(t_1^q) &= NT(t_1^0)+\frac{T(t_0^1)-T(t_0^0)}{1-K_0}K_1\left(N-\frac{1-K_0^N}{1-K_0}\right),\\ \sum_{q=0}^{N-1}T(t_2^q) &= NT(t_2^0)+\frac{T(t_0^1)-T(t_0^0)}{1-K_0}K_2\left(N-\frac{1-K_0^N}{1-K_0}\right),\\ \sum_{q=0}^{N-1}T(t_3^q) &= NT(t_3^0)+\frac{T(t_0^1)-T(t_0^0)}{1-K_0}K_3\left(N-\frac{1-K_0^N}{1-K_0}\right). \end{aligned} \tag{21}$$

Correspondingly, the summation of the integral terms of all "B" intervals, "C" intervals and "D" intervals are

$$\begin{aligned} \sum_{x=0}^{N-1}\int_{t_1^x}^{t_2^x}T(t)dt &= NG(1)((t_2^0-t_1^0)+\beta_B)-\beta_B\cdot\sum_{q=0}^{N-1}T(t_1^q),\\ \sum_{x=0}^{N-1}\int_{t_2^x}^{t_3^x}T(t)dt &= NG(4)((t_3^0-t_2^0)+\beta_C)-\beta_C\cdot\sum_{q=0}^{N-1}T(t_2^q),\\ \sum_{x=0}^{N-1}\int_{t_3^x}^{t_0^{x+1}}T(t)dt &= NG(2)((t_0^1-t_3^0)+\beta_D)-\beta_D\cdot\sum_{q=0}^{N-1}T(t_3^q), \end{aligned} \tag{22}$$

where $\beta_B=1/B(1)\cdot\exp(-B(1)(t_2^0-t_1^0)-1)$; $\beta_C=1/B(4)\cdot\exp(-B(4)(t_3^0-t_2^0)-1)$ and $\beta_D=1/B(2)\cdot\exp(-B(2)(t_0^1-t_3^0)-1)$.

[1] The details regarding how to derive Eq. (16) and (17) can be found in [16].

Therefore, the overall energy consumption of the periodic schedule $\tilde{S}$ after running for N copies can be formulated as

$$\tilde{E}_{total} = \frac{1}{a} \cdot \left(\frac{(T(t_0^1) - T(t_0^0)) \cdot (1 - K_0^N)}{1 - K_0} + b \cdot \left(\sum_{x=0}^{N-1} \int_{t_0^x}^{t_1^x} T(t)dt + \sum_{x=0}^{N-1} \int_{t_1^x}^{t_2^x} T(t)dt + \sum_{x=0}^{N-1} \int_{t_2^x}^{t_3^x} T(t)dt + \sum_{x=0}^{N-1} \int_{t_3^x}^{t_0^{x+1}} T(t)dt \right) \right) \tag{23}$$

Note that, in Eq. (23), given an initial temperature, e.g. $T(t_0^0)$, we only need to calculate the starting temperature of intervals within the first copy of $\tilde{S}$ to get the entire temperature information of the schedule and as a result, the overall energy consumption can be calculated accordingly. It is worth of mentioning that the accuracy to employ Eq. (23) to estimate energy consumption is contingent upon the accuracy of using linear approximation method to estimate the leakage. In Section 7, we use experiments to quantitatively evaluate the accuracy and efficiency of our proposed energy estimation method.

In the next section, we apply the proposed energy estimation method to solve the overall energy minimization scheduling problems.

6. Our energy minimization scheduling algorithms

In the previous section, we proposed a simple yet effective energy estimation method to calculate the overall energy consumption for a given schedule. Then, by formulating the differences between the ending and the starting temperature of each consecutive copy of a periodic schedule as geometric series, we further proposed an efficient method to estimate the overall energy consumption of a given periodic schedule.

In this section, based on the proposed energy estimation methods and the so-called *M-Oscillating* technique [24], two energy minimization scheduling algorithms are presented. The first one is targeting at reducing the overall steady state energy consumption of a real-time system consisting of only one periodic task, while the second one can be applied for a more general case that a real-time system consists of multiple sporadic tasks scheduled under the *EDF* policy.

6.1. Energy minimization for a single task under the thermal steady state

In this subsection, we propose an off-line scheduling approach to minimize the overall energy consumption of a system under the thermal steady state. We assume that the task set consists of a set of periodic tasks with equal period and deadline. We can equivalently assume such a system consists of only one periodic task with its deadline equal to its period. Formally, the problem that we want to address in this subsection can be formulated as follows:

Problem 1. Given a hard real-time task set with period p and worst case execution time c, develop a feasible schedule such that the overall energy consumption can be minimized under the thermal steady state.

6.1.1. Constructing an M-Oscillating schedule by considering transition overhead

The proposed scheduling algorithm is based on the principle of the *M-Oscillating* approach [24], which has shown to be effective in controlling peak temperature. Given a task with period of p and the worst case execution time of c, it is not always possible to use the constant speed $S_{con} = c/p$ to execute the task and two neighboring speeds of S_{con} (S_i and S_j, ($S_i < S_j$)) are used as shown in Fig. 3(a). Let the processor runs at S_i for t_i seconds followed by t_j seconds of S_j to complete the workload. The *M-Oscillating* approach calls for alternating S_i and S_j for every t_i/m and t_j/m seconds, respectively.

Note that, in the original *M-Oscillating* approach [24], the transition overhead incurred by the processor mode switching is not considered. However, in reality, due to the transition overhead, the clock will halt for a short interval τ at each speed transition and a workload loss is imposed [23]. One solution to compensate this workload loss is to increase the processor speed so that more workloads can be finished with the same time. However, this option is not always viable, e.g. when the current speed is already the maximum speed. Alternatively, one could extend the high speed interval and reduce the low speed interval to make this compensation. Therefore, when considering the transition timing overhead and letting t_{im} (and t_{jm}) be the adjusted length of the low (and high) speed interval in an *M-Oscillating* schedule, then based on Fig. 3(b), we can establish the following relationships,

$$m(t_{im} + t_{jm} + 2\tau) = p, \tag{24}$$

$$S_i t_{im} + S_j t_{jm} = \frac{c}{m}. \tag{25}$$

The first equation denotes that the total duration of the adjusted *M-Oscillating* approach equals the period of the task, while the second equation guarantees the required workload can be completed by the deadline. Hence, with any given m, we can immediately obtain values of t_{im} and t_{jm} accordingly.

Furthermore, based on the above equations, the amount of time, i.e. δ, that needs to be taken away from low speed interval can be calculated as

$$\delta = \frac{(S_i + S_j) \cdot \tau}{S_j - S_i} \tag{26}$$

From Eq. (26), we can easily notice that δ is a positive number. Thus, the number of transitions, i.e. m, cannot be arbitrarily increased since the low speed duration t_i in the ideal two-speed schedule has to be sufficiently large to accommodate m speed transitions. Therefore, the maximum allowable m can be calculated by letting

$$m \cdot (\delta + \tau) \le t_i \tag{27}$$

or

$$m \le \frac{t_i}{\delta + \tau} \tag{28}$$

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

$$m \le m_{Max} = \lfloor \frac{t_i}{\delta + \tau} \rfloor, \tag{29}$$

One special case of our approach occurs when the calculated S_{con} is less than the lowest available speed, then we choose the lowest speed and the idle mode for oscillation. Under this scenario, we do not need to adjust the high speed duration and the low speed duration to compensate the workload loss since there is no task execution during the idle mode in the first place. The t_{im} and t_{jm} can thus be calculated as (t_i/m) and (t_j/m), respectively, while the upper bound of m is given by $\lfloor (t_i/2\tau) \rfloor$.

At this point, given the *WCET* and the deadline/period of the task, we can find a feasible *M-Oscillating* schedule that can guarantee the real-time constraint of the task, and at the same time, with the non-negligible transition overhead being considered. Now the problem

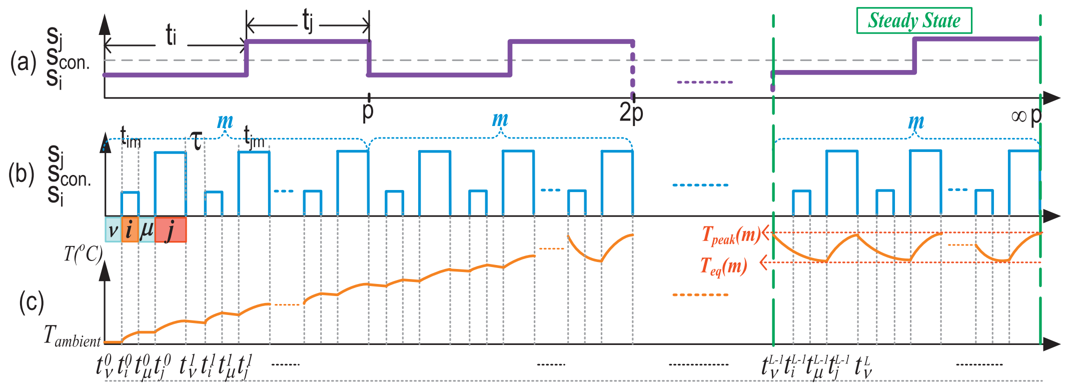


Fig. 3. M-Oscillating and the corresponding temperature curve at thermal steady state.

is how to determine the best value of m that achieves the minimal overall energy consumption under the thermal steady state. Since the (leakage) energy consumption is strongly depending on the temperature, we need to obtain the temperature information of a candidate schedule before we can evaluate its energy consumption. Therefore, we next study the thermal characteristic of the *M-Oscillating* schedule.

6.1.2. *Calculating the steady state temperature and energy consumption*

Consider an *M-Oscillating* schedule and the corresponding temperature curve, which are depicted in Fig. 3(b) and (c). Without losing of generality, we assume that there are totally m divisions and correspondingly, $2m$ times of mode switching within one task period. The duration that the processor spends in ith and jth mode within one division, i.e. t_{im} and t_{jm}, can be calculated by Eq. (24). The processor is assumed to be turned into idle mode between each speed transition for a small time frame τ. To ease the presentation, we denote that the processor is in mode μ during each low-high transition and in mode ν during each high-low transition even thought they represent the same idle mode. Similar to the example we showed in Fig. 2, in an *M-Oscillating* schedule, $T(t_M^\alpha)$ denotes the starting temperature of the processor mode M in the αth division.

As we start to run the processor with the *M-Oscillating* schedule from certain initial temperature, i.e. T_{amb} in this case, the temperature will gradually increase. Eventually, when the heat generated by the processor matches the heat that can be removed by the heatsink, the processor is said to be in its thermal steady state, during which, the temperature will follow a periodic pattern instead of increase indefinitely. Our goal is to minimize the overall energy consumption at the thermal steady state.

From Fig. 3(b) and (c), it is not difficult to see that in the thermal steady state, the processor reaches its peak temperature (T_{peak}) at the end of each high speed interval and then drops to a so-called equilibrium temperature (T_{eq}) after each idle period that immediately follows the low speed interval (μ interval). Note that, based on the previously proposed energy equation, the overall energy consumption of the *M-Oscillating* schedule under the thermal steady state can be formulated once the temperature information at all mode switching instants are available. Then, for all possible value of m, we choose the one that results in the minimum overall energy consumption as our solution. One straightforward way to calculate the peak temperature for a given m is to trace the temperature change using Eq. (6) from the T_{amb} until the temperature saturated at certain time instant and pick the highest one as the T_{peak}. However, this method can be very time consuming if the number of candidate schedules, i.e. m, is large. One better solution is to treat the *M-Oscillating* schedule as a periodic schedule and formulate the temperature change analytically. Similar to Eq. (16), the temperature at time instant t_ν^L can be formulated as

$$T(t_\nu^L) = T(t_\nu^0) + \frac{(T(t_\nu^1) - T(t_\nu^0)) \cdot (1 - K_0^L)}{1 - K_0} \tag{30}$$

where $K_0 = \exp(-B(j)t_{jm} - B(i)t_{im} - 2B_{idle}\tau)$ and $B_{idle} = B(\mu) = B(\nu)$. Then, the peak temperature of a given m can be obtained by setting $L \to \infty$. Since K_0 is always less than 1 [16], we have

$$\lim_{L\to\infty} T(t_\nu^L) = T(t_\nu^0) + \frac{T(t_\nu^1) - T(t_\nu^0)}{1 - K_0}. \tag{31}$$

Note that once T_{peak} is found, other temperature values at the mode transition points within the same segment are readily available based on Eq. (6). Thus, the overall energy consumption in one period at the steady state can be formulated as

$$\begin{aligned}\tilde{E}(m)_{steady} &= m \cdot (E(t_\nu^{L-1}, t_i^{L-1}) + E(t_i^{L-1}, t_\mu^{L-1}) + E(t_\mu^{L-1}, t_j^{L-1}) \\ &\quad + E(t_j^{L-1}, t_\nu^L)) + 2m \cdot E_{sw},\end{aligned} \tag{32}$$

where $E(t_i^{L-1}, t_\mu^{L-1})$ and $E(t_j^{L-1}, t_\nu^L)$ denote the energy consumptions of the low and high speed intervals, respectively. $E(t_\nu^{L-1}, t_i^{L-1})$ and $E(t_\mu^{L-1}, t_j^{L-1})$ are corresponding to the energy consumption of the idle period. E_{sw} is the switching energy overhead and there are totally $2m$ transitions taking place.

Our problem, therefore, is to minimize $\tilde{E}(m)_{steady}$ subject to $m \le m_{Max}$. Because of the simplicity of both our energy estimation technique as well as the peak/equilibrium temperature calculation method, we can use simple exhaustive search to find the optimal value of m.

6.2. *Energy minimization of multiple sporadic tasks under EDF policy*

In Section 6.1, we have introduced an effective energy minimization speed scheduling approach targeting at the thermal steady state. This approach is off-line in nature since we assume a task always takes its *WCET* and the processor has already reached the thermal steady state. However, in real world applications, the actual execution time of a task can be only a small fraction of its *WCET*. A schedule developed off-line may be more effective to provide certain guarantee, but an on-line schedule can be more effective in saving energy. Moreover, for a more complicated real-time system, such as a real-time system consisting of multiple sporadic tasks scheduled by the *EDF* policy, it rarely reaches the thermal stable state. In such a scenario, the off-line algorithms simply become not effective at all.

In this subsection, we extend our method and develop an on-line scheduling algorithm for task sets with multiple tasks scheduled under the *EDF* policy. Based on our system models, the problem is formally defined as follows,

Problem 2. Given a hard real-time task set Γ consisting of N sporadic tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$, develop an on-line speed schedule under the *EDF* policy such that the overall energy consumption can be minimized.

We still employ the *M-Oscillating* technique in the proposed on-line method. One of the key challenges for the on-line algorithm is to calculate the energy consumption on-line. Therefore, in the remaining part of this subsection, we first introduce how to formulate the energy consumption of an *M-Oscillating* schedule without requiring such a schedule reaches its thermal steady state. Next, we discuss how to combine the *M-Oscillating* with and the existing *EDF* policy to schedule multiple sporadic tasks.

6.2.1. *Online M-Oscillating energy calculation without steady state temperature information*

The energy consumption of an *M-Oscillating* schedule during the transient stage, i.e. before reaching the steady state, can be formulated by using the same method proposed in Section 5.2. Consider an *M-Oscillating* schedule and the corresponding temperature curve in Fig. 4. The *M-Oscillating* schedule can be considered as a periodic speed schedule with one period consisting of four speed intervals, namely, μ, j, ν and i. Therefore, the equations we derived in Section 5.2, i.e. Eq. (23), can be effectively applied to formulate the energy consumption of an *M-Oscillating* schedule during the transient stage.

Assuming at certain scheduling point, if there are m divisions and the current temperature is $T(t_\mu^0)$, then the overall energy consumption is

$$\tilde{E}(m) = \frac{1}{a}\cdot\left(\frac{(T(t_\mu^1)-T(t_\mu^0))\cdot(1-K_0^m)}{1-K_0} + b\cdot\left(\sum_{x=0}^{m-1}\int_{t_\mu^x}^{t_j^x}T(t)dt + \sum_{x=0}^{m-1}\int_{t_j^x}^{t_\nu^x}T(t)dt + \sum_{x=0}^{m-1}\int_{t_\nu^x}^{t_i^x}T(t)dt + \sum_{x=0}^{m-1}\int_{t_i^x}^{t_\mu^{x+1}}T(t)dt\right)\right), \quad (33)$$

where the four summation terms can be expressed as

$$\begin{aligned}
\sum_{x=0}^{m-1}\int_{t_\mu^x}^{t_j^x}T(t)dt &= m\cdot G_{idle}\cdot(\tau+\beta_\mu) - \beta_\mu\cdot\sum_{q=0}^{m-1}T(t_\mu^q),\\
\sum_{x=0}^{m-1}\int_{t_j^x}^{t_\nu^x}T(t)dt &= m\cdot G(j)\cdot(t_{jm}+\beta_j) - \beta_j\cdot\sum_{q=0}^{m-1}T(t_j^q),\\
\sum_{x=0}^{m-1}\int_{t_\nu^x}^{t_i^x}T(t)dt &= m\cdot G_{idle}\cdot(\tau+\beta_\nu) - \beta_\nu\cdot\sum_{q=0}^{m-1}T(t_\nu^q),\\
\sum_{x=0}^{m-1}\int_{t_i^x}^{t_\mu^{x+1}}T(t)dt &= m\cdot G(i)\cdot(t_{im}+\beta_i) - \beta_i\cdot\sum_{q=0}^{m-1}T(t_i^q),
\end{aligned}$$

where $\beta_j = 1/B(j)\cdot\exp(-B(j)t_{jm}-1)$; $\beta_i = 1/B(i)\cdot\exp(-B(i)t_{im}-1)$; $\beta_\mu = \beta_\nu = 1/B_{idle}\cdot\exp(-B_{idle}\tau-1)$ and $G_{idle} = G(\mu) = G(\nu)$.

Similarly, each summation term (summation of initial temperatures) on the R.H.S of the above equations can further be calculated as

$$\begin{aligned}
\sum_{q=0}^{m-1}T(t_\mu^q) &= m\cdot T(t_\mu^0) + \frac{T(t_\mu^1)-T(t_\mu^0)}{1-K_0}\left(m - \frac{1-K_0^m}{1-K_0}\right),\\
\sum_{q=0}^{m-1}T(t_j^q) &= m\cdot T(t_j^0) + \frac{T(t_\mu^1)-T(t_\mu^0)}{1-K_0}K1\left(\frac{1-K_0^m}{1-K_0}\right),\\
\sum_{q=0}^{m-1}T(t_\nu^q) &= m\cdot T(t_\nu^0) + \frac{T(t_\mu^1)-T(t_\mu^0)}{1-K_0}K2\left(\frac{1-K_0^m}{1-K_0}\right),\\
\sum_{q=0}^{m-1}T(t_i^q) &= m\cdot T(t_i^0) + \frac{T(t_\mu^1)-T(t_\mu^0)}{1-K_0}K3\left(\frac{1-K_0^m}{1-K_0}\right),
\end{aligned}$$

where $K_0 = \exp(-B(j)t_{jm} - B(i)t_{im} - 2B_{idle}\tau)$, $K1 = \exp(-B_{idle}\tau)$, $K2 = \exp(-B(j)t_{jm} - B_{idle}\tau)$ and $K3 = \exp(-B(j)t_{jm} - 2B_{idle}\tau)$.

Note that, in Eq. (33), given an initial temperature, e.g. t_μ^0, we only need to calculate the temperature at those time instants within the first sub-interval, e.g. t_j^0, t_ν^0, t_i^0 and t_μ^1, to get the entire temperature information and as a result, the overall energy consumption of an *M-Oscillating* schedule with m divisions can be calculated efficiently.

6.2.2. *Combining M-Oscillating with online EDF*

In this subsection, based on the proposed energy estimation method, we present our on-line leakage aware energy minimization scheduling algorithm, namely, the "*On-line M-Oscillating*" (*OLMO*) approach.

To guarantee the real-time constraint, we adopt the method in [4] to calculate the constant speed. According to [4], for an arriving task τ_i (either new arrival or resumed after preemption) at time instant t, the constant speed can be determined as follows.

We first calculate the *worst-case completion time* (*WCCT*) and *worst-case remaining time* (*WCRT*) for each arrived (but not finished) task. Three cases are considered:

- τ_i preempts τ_j
  $WCCT_i = t + c_i$
  $WCRT_i = c_i$
  $WCRT_j = WCRT_j - s_j(t-l)$ /*l:previous context switch time*/
- τ_i resumes after τ_k
  $WCCT_i = WCCT_i + WCCT_k - t_p$ /* τ_i was preempted at t_p*/
- otherwise
  $WCRT_i = c_i$
  if($d_k > d_i$ or $WCCT_i < t$), $WCCT_i = t + WCRT_i$
  else $WCCT_i = WCCT_k + c_i$,

The required constant speed of τ_i can thus be calculated as

$$s_{con} = \frac{WCRT_i}{WCCT_i - t}. \quad (34)$$

Now the two neighboring speeds (S_i and S_j) are found based on the calculated S_{con} and the given processor model. To guarantee the same workload during the period t_p, the low speed duration t_i and the high speed duration t_j can be formulated as $t_j = t_p \times ((S_{con} - S_i)/(S_j - S_i))$ and $t_i = t_p - t_j$, respectively.

Similarly, to incorporate the non-negligible transition overhead, we use Eq. (24) to find the duration of the adjusted high speed (t_{jm}) and low speed (t_{im}) interval under different m values.

At each scheduling point, given the initial temperature $T(t_0^0)$, the overall energy consumption, i.e. $\tilde{E}(m)$, can thus be formulated as a function of m (Eq. (33)). Then, our problem is to minimize $\tilde{E}(m)$ subject to $m \le m_{Max} = \lfloor((t_i)/(\delta+\tau))\rfloor$ and the real-time constraint.

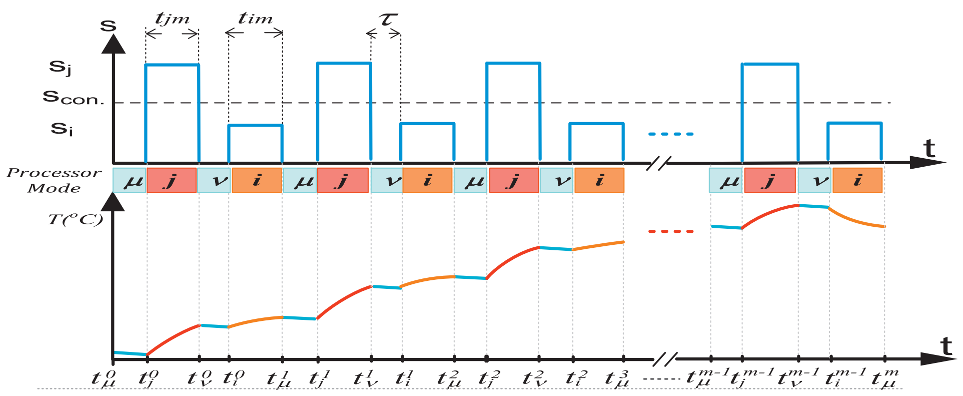


Fig. 4. M-Oscillating and the corresponding temperature curve at transient stage.

We can again use a sequential search to find the optimal value of m during the run-time. The detailed algorithm is presented in Algorithm 1.

As an on-line scheduling algorithm, the complexity of the proposed *OLMO* is a crucial factor to be considered. In our approach, the most time consuming task is the searching process for the optimum m. The complexity of the searching algorithm is $O(m_{Max})$, where m_{Max} is the theoretical upper bound of m, i.e. $m_{Max} = \lfloor ((t_i)/(\delta+\tau)) \rfloor$. Based on the task and processor models we used in this paper, in most cases, the optimal m can be found within 20 iterations which would pose negligible overhead as evidenced later in Section 7.1. To further improve the efficiency of finding the m_{opt}, we can use some more elaborative searching algorithm such as those target at unimodal functions [32].

Algorithm 1. On-line M-Oscillating (OLMO) algorithm

```
1:  while context switch to task $\tau_i$ at time t do
2:    Calculate the constant speed $S_{con}$ by Eq. (34);
3:    Find two nearest neighboring speeds $S_i$ and $S_j$ of the constant speed;
4:    Calculate ideal low/high speed duration $t_i$ and $t_j$
5:    Identify upper bound of $m$:
6:    if $S_i \neq idle$ then
7:      $m_{Max} = \lfloor \frac{t_i}{\delta+\tau} \rfloor$
8:    else
9:      $m_{Max} = \lfloor \frac{t_i}{2\tau} \rfloor$
10:   end if
11:   Assign: $E_{min} = \infty$ and $m_{opt} = 0$
12:   for $m$=1:$m_{Max}$ do
13:     if$S_i \neq idle$ then
14:       $t_{im} = \frac{t_i}{m} - \tau - \delta$;
15:       $t_{jm} = \frac{t_j}{m} - \tau + \delta$;
16:     else
17:       $t_{im} = \frac{t_i}{m}$
18:       $t_{jm} = \frac{t_j}{m}$
19:     end if
20:     Calculate $\tilde{E}(m)$; Eq. 23
21:     if$\tilde{E}(m) < E_{min}$
22:       $E_{min} \leftarrow \tilde{E}(m)$
23:       $m_{opt} \leftarrow m$
24:     end if
25:   end for
26:   Return: $\{S_i, S_j, t_{im}, t_{jm}, m_{opt}\}$
27: end while
```

In the next section, we use experiments to validate the accuracy of the proposed energy estimation methods and the performance of our scheduling algorithms.

7. Experimental results

In this section, we use experiments to evaluate our methods. We first examine the accuracy and efficiency of the proposed energy estimation equations. Then, the performance of the proposed energy minimization scheduling techniques will be tested and discussed.

7.1. Accuracy and efficiency of energy estimation technique

In this subsection, we study the accuracy and efficiency of our energy estimation methods. We adopted the processor model similar to the one in [16], which is built based on the work by Liao et al. [2] using the 65 nm technology. Specifically, we use Eq. (2) to compute the leakage currents for temperature from 20 °C to 140,°C with a step size of 5,°C, and supply voltage from 0.6 V to 1.2 V with a step size of 0.0 5 V. These results were used to determine the temperature invariants $C_0(k)$ and $C_1(k)$ in Eq. (3) through curve-fitting. The thermal constants are chosen according to [33]. The ambient temperature is set to 25 °C. The timing penalty of speed transition τ is set to 5 ms and the transition energy overhead E_{sw} is 0.01 J [15].

First, we test the performance of our energy estimation method for a single voltage schedule interval, i.e. the method given by Eq. (11). We let the processor run in intervals with different lengths (as shown in the first column of Table 1) and by using different modes. We chose the energy estimation method proposed in work [31] as our baseline. Specifically, an interval is split into a series of small intervals in each of which the power consumption is close to a constant. The power consumption within each interval is thus calculated based on Eq. (7). For accurate energy estimation, we used a very small time interval, e.g. 0.01 s for each sub-interval. We compared our proposed method, denoted as *PROP*, with that of work [31] from the aspects of energy consumption and computational cost. We denoted the Average CPU Time, Average Relative Error and Average Speedup as A.C.T., A.R.E and A.S., respectively. The final results were shown in Table 1.

Our experimental results clearly show that the proposed energy estimation method is very accurate and computationally efficient. As can be seen in Table 1, the energy consumption estimated by using our proposed method closely matches the one by using the approach in [31], with the maximum relative error no more than 2.7% among all our test cases. The computational cost, on the other hand, is significantly reduced, i.e. up to 177 times of speedup or over 57 times of speedup in average for our test cases. We can also see from Table 1 that the longer the interval is, the more efficient our method can be. The reason is that the computational cost of our method is independent to the length of the interval, but for the method in [31], the CPU time is very sensitive to the interval length since it needs to divide the entire schedule length into small sub-intervals. Moreover, Table 1 also indicates that in general the relative estimation errors of our method decrease as we increase

Table 1
Accuracy and efficiency of energy estimation method for single voltage schedule interval.

Interval length (s)	V_{dd} (v)	Energy (J) REAL	Energy (J) PROP	Error	A.C.T (s) REAL	A.C.T. (s) PROP	A.S.
5	0.6	29.66	28.86	2.7%	0.028	0.0025	11×
	0.8	68.39	68.23	2.3%			
	1.0	152.75	154.74	1.3%			
	1.2	387.49	392.73	1.3%			
10	0.6	59.37	57.79	2.6%	0.038	0.0027	14×
	0.8	139.13	136.10	2.1%			
	1.0	306.45	310.56	1.2%			
	1.2	780.20	790.68	1.3%			
20	0.6	118.79	115.66	2.6%	0.064	0.0026	24×
	0.8	278.49	273.53	2.0%			
	1.0	623.74	631.80	1.3%			
	1.2	1.57×10^3	1.59×10^3	1.3%			
50	0.6	297.04	289.26	2.7%	0.17	0.0028	60×
	0.8	696.56	681.83	2.1%			
	1.0	1.56×10^3	1.58×10^3	1.2%			
	1.2	3.92×10^3	3.97×10^3	1.3%			
100	0.6	594.13	578.60	2.7%	0.48	0.0027	177×
	0.8	1.39×10^3	1.36×10^3	2.1%			
	1.0	3.16×10^3	3.12×10^3	1.3%			
	1.2	7.84×10^3	7.95×10^3	1.3%			

the supply voltage levels. This is because for the same interval length, higher supply voltage level results in larger absolute value of energy consumption, which increases the denominator when we calculate the relative errors.

We next use experiments to examine the performance of the proposed on-line energy estimation method, i.e. Eq. (33). We generated five groups of tasks with different deadlines (D), e.g. 5 s, 10 s, 20 s, 50 s and 100 s. Each group consists of 1000 tasks with randomly generated actual execution times varying from $0.5D$ to $0.95D$. For each task, we calculate the energy consumption of the *M-Oscillating* schedules with m varied from 1 to m_{Max}. We assume that the initial temperature is the same as the ambient temperature. Three approaches were used to obtain the energy consumption.

- The proposed online periodic schedule energy estimation method, i.e. Eq. (33) in Section 6.2.1. We refer this method as the *PSE* method.
- The single interval based energy calculation method. Specifically, we use Eq. (11) derived in Section 6.1 to calculate the energy consumption of each interval and apply Eq. (6) to trace the temperature. We call it the *SIE* method.
- The energy calculation method similar to the one in [31].

Based on our previous discussion, for a single interval, calculating the energy by using Eq. (11) has already demonstrated significant speedup over the method in [31], while the latter has better performance in terms of accuracy (Table 1). Therefore, in this experiment, to evaluate the accuracy of the *PSE* method, we still compare the energy values calculated by the *PSE* method with the ones obtained by the method in [31]. The *SIE* method is used as the baseline approach to show the additional speedup achieved by the *PSE* method. We collected and summarized the corresponding experimental results in Table 2.

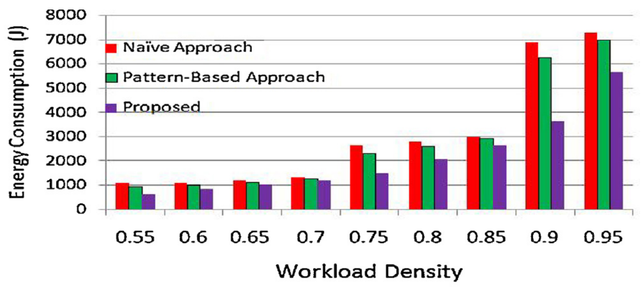
As can be seen in Table 2, our experimental results show the superiority of our the *PSE* method. On one hand, the energy consumption estimated by using the *PSE* method well matches the one obtained by the approach in [31] with the relative error kept within 4.1%. The CPU time, on the other hand, is significantly reduced compared with the *SIE* method, i.e. up to 210× of speedup or over 94× of speedup in average for different task groups. The speedups of the *PSE* method over the *SIE* method come from the following facts. For each valid m, the *PSE* method can efficiently obtain the potential energy consumption based on the current temperature reading and four simple temperature calculations, i.e. the ending temperatures of the 4 intervals within the first period of an *M-Oscillating* schedule. The *SIE* method, on the other hand, needs to calculate the energy consumption interval by interval and therefore, $4 \times m$ steps are required to find the overall energy consumption. As m becomes larger, the computational time increases significantly. Moreover, the *SIE* method needs the temperature information at every speed transition point as the input, thus, a complete thermal simulation is required for the entire schedule under each m, which can be extremely time consuming. This experiment shows the fact that the accuracy of the *PSE* method is maintained while the computational overhead is significantly reduced, which is crucial for our on-line scheduling algorithm.

7.2. Energy minimization under thermal steady state

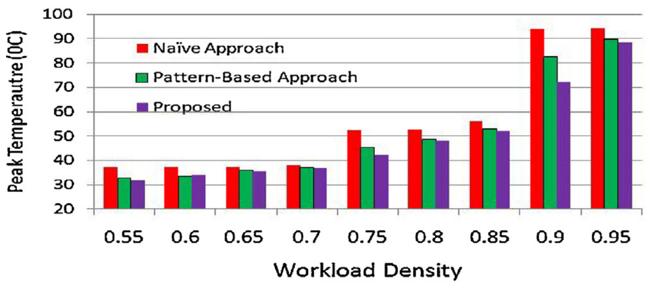
To study the energy saving performance of our scheduling approach under the thermal steady state, we compared our approach with two existing methods,

Table 2
Evaluation of transient state energy calculation method.

Interval length (s)	Ave. m_{max}	A.C.T. (s) [31]	A.C.T. (s) SIE	A.C.T (s) PSE	A.S. PSE vs. SIE	A.R.E. PSE vs. [31]
5	63	4.68	0.034	0.00092	36×	4.1%
10	117	9.63	0.093	0.0019	49×	3.4%
20	248	18.38	0.30	0.0037	81×	2.8%
50	625	46.2	1.81	0.014	129×	1.9%
100	1180	108	6.03	0.03	210×	1.1%



(a) Energy Vs. Workload



(b) Temperature Vs. Workload

Fig. 5. Performance comparison of three different scheduling approaches.

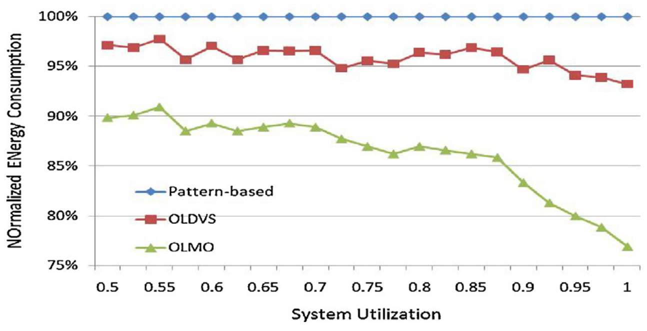


Fig. 6. Normalized energy consumption of three scheduling approaches.

- A simple naive approach that executes the task with the maximal speed and turn to sleep mode upon the finish of task execution.
- The *Pattern-based* approach [15] that uses a single speed to run the task. This approach lets the processor's active state to be scattered into a number of intervals so that the processor can have a chance to cool down.

The periods (p_i) of the tasks were set to 100 s. The worst case execution times were randomly generated with workload density ($u_i = c_i/p_i$) varying from 0.55 to 0.95 with a step width of 0.05. The corresponding workload (c_i) of each task were calculated by $c_i = u_i \times p_i$. The reason we do not further reduce the workload density is that when $u_i < 0.55$, the low speed mode used by the Pattern-based approach and the proposed technique become the same. The temperature constraint was set to $T_{max} = 85\,°C$. The energy consumption of each task was calculated and plotted in Fig. 5(a).

From Fig. 5(a), while both the *Pattern-based* approach and the proposed technique can achieve significant energy reduction, the proposed method consistently outperforms the *Pattern-based* approach under different test scenarios. Compared with the naive approach, on average, 26.5% energy can be saved by using our method versus 13.2% by the Pattern-based approach. Compared with the *Pattern-based* approach, the energy reduction of our method achieves from two facts. (i) By using the speed oscillating, the leakage energy is reduced as a result of the lower peak temperature. (ii) By applying the two neighboring speed instead of maximum speed and idle mode, the dynamic energy consumption is reduced as well.

We also compare the peak temperature of each schedule, as shown in Fig. 5(b). The peak temperature achieved by the proposed approach can be 21.8 °C and 11.4 °C (or 6.4 °C and 2.1 °C in average) lower than the naive approach and the *Pattern-based* approach, respectively. Therefore, our approach can be even more effective when the peak temperature constraint becomes tighter.

7.3. *Online energy minimization under EDF policy*

We next evaluate the performance of our proposed *OLMO* algorithm by comparing with the *Pattern-based* approach [15] and the *OLDVS* approach [4]. This method uses two neighboring speeds of the *constant speed* to execute the task so that the workload can finish exactly at the deadline.

The task model used in this experiment is set as follows. The utilization (u_i) of each task (τ_i) is uniformly distributed within [0.02, 1.00]. The minimum inter-arrival time (P_i) of τ_i also has a uniform distribution, ranging between [50, 500]. To simulate the actual execution time (e_i), we define the *actual-to-worst ratio* (*e2E_ratio*) as the ratio of e_i over E_i, which was generated randomly for each job with uniform distribution between [0, 1]. Then for each job of τ_i, e_i can be obtained by $e_i = e2E\_ratio \cdot E_i$. The system utilization (workload density) is set between [0.5,1.0] with 5% increment. For each testing case, 1000 tasks were generated and bounded by the corresponding system utilization. The overall energy consumptions by three different approaches were collected and then normalized to the ones achieved by the *Pattern-based* approach.

The results are plotted in Fig. 6 which show that the *OLMO* significantly outperforms the other two existing approaches. Compared with the *Pattern-based* approach, the *OLMO* can save more than 10% energy on average at low system utilization, while up to 20% energy can be saved at high system utilization (e.g. 11% and 23% energy savings at system utilization equal to 0.5 and 1.0, respectively and 14% saving on average). Compared with the *OLDVS*, the *OLMO* achieves 10% energy saving on average, and the energy saving also increases with the increment of the system utilization. Compared with both the *Pattern-based* approach and the *OLDVS*, the proposed *OLMO* can significantly reduce the overall energy consumption, particularly under higher system utilization. This is because the processor tends to choose higher speed levels when the system utilization is high. As a result, the absolute values of the energy consumption as well as the temperature are high. The *M-Oscillating* can achieve significant temperature reduction, especially when the system stays in high temperature range. The minimized temperature curve directly translate to a reduced leakage energy consumption which is another contributor of energy saving.

8. Conclusions

In this paper, we study the problem on how to reduce the overall energy consumption of a hard real-time system by applying real-time scheduling techniques with the leakage/temperature dependency taken into consideration. We first develop an energy estimation method that can be used to accurately and efficiently calculate the overall energy consumption of a candidate schedule. Then, based on the proposed energy equation, two scheduling approaches are proposed. The first one is an off-line approach, which focuses on how to minimize the energy of a single periodic task under the thermal steady state. The second one is an on-line approach that can reduce the overall energy consumption of a hard real-time system scheduled according to the *EDF* policy. Our experimental results demonstrate that the proposed energy estimation method can achieve significant speedup compared with existing approaches while maintaining good accuracy. In addition, with a large number of different test cases, both the on-line and off-line approaches we proposed in this paper outperform the previous works consistently.

References

[1] S. Borkar, Thousand core chips: a technology perspective, in: DAC, 2007, pp. 746–749.
[2] W. Liao, L. He, 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 (2005) 1042–1053.
[3] H.F. Sheikh, I. Ahmad, Z. Wang, S. Ranka, An overview and classification of thermal-aware scheduling techniques for multi-core processing systems, in: Sustainable Computing: Systems and Informatics, vol. 2, 2012, pp. 151–169.
[4] C.-H. Lee, K. Shin, On-line dynamic voltage scaling for hard real-time systems using the EDF algorithm, in: Real-time Systems Symposium, 2004. Proceedings. 25th IEEE International, 2004, pp. 319–335, http://dx.doi.org/10.1109/REAL.2004.38.
[5] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: FOCS, 1995, pp. 374–382.
[6] J.-J. Chen, H.-R. Hsu, T.-W. Kuo, Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems, in: RTAS, 2006, pp. 408–417.
[7] R. Jejurikar, C. Pereira, R. Gupta, Dynamic slack reclamation with procrastination scheduling in real-time embedded systems, DAC (2005) 111–116.
[8] N. Bansal, T. Kimbrel, K. Pruhs, Speed scaling to manage energy and temperature, Journal of the ACM 54 (2007) 1–39.
[9] G. Quan, Y. Zhang, W. Wiles, P. Pei, Guaranteed scheduling for repetitive hard real-time tasks under the maximal temperature constraint, in: ISSS + CODES, 2008.
[10] A. Coskun, D. Atienza, T. Rosing, T. Brunschwiler, B. Michel, Energy-efficient variable-flow liquid cooling in 3d stacked architectures, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2010, 2010, pp. 111–116.
[11] H. Huang, G. Quan, J. Fan, Leakage temperature dependency modeling in system level analysis, in: ISQED, 2010, pp. 447–452.
[12] M. Bao, A. Andrei, P. Eles, Z. Peng, Temperature-aware idle time distribution for energy optimization with dynamic voltage scaling, in: DATE, 2010, pp. 21–27.
[13] M. Bao, A. Andrei, P. Eles, Z. Peng, On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration, in: Design Automation Conference, 2009, pp. 490–495.
[14] H. Huang, G. Quan, Leakage aware energy minimization for real-time systems under the maximum temperature constraint, in: DATE, 2011.
[15] C.-Y. Yang, J.-J. Chen, L. Thiele, T.-W. Kuo, Energy-efficient real-time task scheduling with temperature-dependent leakage, in: DATE, 2010, pp. 9–14.
[16] G. Quan, V. Chaturvedi, Feasibility analysis for temperature-constraint hard real-time periodic tasks, IEEE Transaction on Industrial Informatics 6 (2010) 329–339.
[17] L. Yuan, S. Leventhal, G. Qu, Temperature-aware leakage minimization technique for real-time systems, in: ICCAD, 2006, pp. 761–764.
[18] Hotspot 4.2 Temperature Modeling Tool, University of Virgina, 2009 http://lava.cs.virginia.edu/HotSpot
[19] P. Kumar, L. Thiele, Cool shapers: shaping real-time tasks for improved thermal guarantees, in: DAC, 2011, pp. 468–473.
[20] H. Yu, B. Veeravalli, Y. Ha, Leakage-aware dynamic scheduling for real-time adaptive applications on multiprocessor systems, in: DAC, 2010, pp. 493–498.
[21] H. Huang, G. Quan, J. Fan, M. Qiu, Throughput maximization for periodic real-time systems under the maximal temperature constraint, in: DAC, 2011, pp. 363–368.
[22] S. Zhang, K.S. Chatha, Thermal aware task sequencing on embedded processors, in: DAC, 2010, pp. 585–590.
[23] T. Chantem, X.S. Hu, R. Dick, Online work maximization under a peak temperature constraint, in: ISLPED, 2009, pp. 105–110.
[24] V. Chaturvedi, H. Huang, G. Quan, Leakage aware scheduling on maximal temperature minimization for periodic hard real-time systems, in: ICESS, 2010, pp. 1802–1809.
[25] J.-J. Chen, S. Wang, L. Thiele, Proactive speed scheduling for real-time tasks under thermal constraints, RTAS (2009) 141–150.
[26] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, G.D. Micheli, Temperature-aware processor frequency assignment for mpsocs using convex optimization, in: CODES + ISSS, 2007, pp. 111–116.
[27] J. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits: A Design Perspective, Prentice Hall, 2003.
[28] Y. Liu, R.P. Dick, L. Shang, H. Yang, Accurate temperature-dependent integrated circuit leakage power estimation is easy, in: DATE, 2007, pp. 1526–1531.
[29] L. Yongpan, Y. Huazhong, Temperature-aware leakage estimation using piecewise linear power models, IEICE Transactions on Electronics 93 (2010) 1679–1691.
[30] G. Quan, L. Niu, B. Mochocki, X.S. Hu, Fixed-priority scheduling for reducing both the dynamic and leakage energy on variable voltage processors, International Journal of Embedded Systems (2008).
[31] Y. Liu, H. Yang, R.P. Dick, H. Wang, L. Shang, Thermal vs energy optimization for dvfs-enabled processors in embedded systems, in: ISQED, 2007, pp. 204–209.
[32] R.W. Pike, Optimization for Engineering Systems, Van Nostrand Reinhold Company, 2001.
[33] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, D. Tarjan, Temperature-aware microarchitecture, in: ICSA, 2003, pp. 2–13.