On-Line Leakage-Aware 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
E-mail: hhuan001@fiu.edu, mfan001@fiu.edu, gaquan@fiu.edu

Abstract

As the semiconductor technology proceeds into the deep sub-micron era, leakage and its dependency with the temperature become critical in dealing with the power/energy minimization problem. In this paper, we develop an analytical method to estimate energy consumption on-line with the leakage/temperature dependency taken into consideration. Based on this method, we develop an on-line scheduling algorithm to reduce the overall energy consumption for a hard real-time system scheduled according to the Earliest Deadline First *(EDF) policy. Our experimental results show that the proposed energy estimation method can achieve up to 210X speedup compared with an existing approach while still maintaining high accuracy. In addition, with a large number of different test cases, the proposed energy saving scheduling method consistently outperforms two closely related researches in average by* 10% *and* 14% *respectively.*

1. Introduction

The continued scaling of semiconductor technology together with the increasingly complicated circuit architecture has resulted in an exponential increase of power density. The escalating power consumption has posed tremendous challenges not only on how to extend the battery life for portable devices, but also on how to reduce operating costs for server systems. Moreover, the soaring power consumption has directly caused 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, which is becoming a major component of the overall power consumption [9].

Power/energy minimization problem has been researched extensively for a couple of decades. Early researches were mainly focused on reducing the dynamic power/energy, i.e. the predominate component in the overall power/energ consumption. By taking advantage of the convex relationship between the dynamic power and supply voltage, many researches (e.g. [12, 20]) strived to lower down the processor supply voltage and working frequency. As the leakage becomes more prominent, a few techniques (e.g. [5, 10]) were proposed to minimize the overall energy, which consists of both dynamic and leakage energy. For example, Jejurikar et al. [5, 10] 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 leakage current is constant. However, as semiconductor technology ventures into the deep sub-micron domain, the leakage/temperature dependency becomes too significant to ignore. As shown in [8], a power/thermal aware design technique simply becomes ineffective if the interdependence of leakage and temperature is not properly addressed.

A number of researches (e.g. [1, 7, 19]) have been published on minimizing the overall energy consumption with leakage/temperature dependency taken into consideration. In [2], Bao et al. proposed 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. Yang et al. [19] developed 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 temperature reaches its stable state. This approach assumes a processor with no *dynamic voltage scaling* (*DVS*) capability. It reduces the energy consumption at the temperature stable status by periodically switching the processor between the active and dormant modes. Huang and Quan [7] used a linear model to capture the interdependency of leakage current and temperature. They developed a closed-form formula to simplify the calculation of energy consumption for a single speed interval. Based on this formula, they proposed a *DVS* scheduling method to minimize the energy consumption at the thermal steady state when scheduling a single periodic task. Note that, all these researches assume tasks always take their *worst case execution time* (*WCET*), and the corresponding scheduling algorithm design and analysis can only be conducted off-line.

We are interested in developing an on-line scheduling algorithm that can minimize the total energy consumption of a hard real-time system for the following reasons. First, it is a well known fact that the 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. Second, for a more complicated real-time system, such as a real-time system consisting of multiple sporadic tasks scheduled by *EDF*, it rarely reaches thermal stable state. In such a scenario, the off-line algorithms introduced above become useless. Bao et al. [1] proposed 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 starting temperature and time, 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 [11, 21]. They focus more on peak temperature reduction or throughput maximization under a peak temperature constraint. To our best knowledge, we are not aware of any other on-line energy minimization approach for hard real-time systems with temperature/leakage interdependency taken into consideration.

In this paper, we develop an on-line algorithm to minimize the energy consumption for a sporadic hard real-time system. Our algorithm incorporates a peak temperature reduction technique, i.e. the *M-Oscillating* technique [4], into an existing on-line *EDF* scheduling technique i.e. the *on-line DVS* approach (*OLDVS*) [12]. A key challenge in our approach is to estimate the energy consumption rapidly and accurately under different run-time conditions and schedules. To solve this problem, we derive an analytical energy estimation method that can efficiently and accurately obtain the potential energy consumption of a candidate schedule under different initial temperature conditions. Our experimental results show that the proposed online energy estimation method can achieve up to 210X of speedup compared with an existing approach [7] while maintaining high accuracy. In addition, under a large number of test cases with different workload densities, the proposed method consistently outperforms two existing approaches by 10% and 14% in average.

The rest of the paper is organized as follows. Section 2 introduces the system models and a motivational example. Section 3 presents our on-line energy estimation method and scheduling approach. Experimental results are discussed in Section 4. Section 5 concludes this paper.

978-1-4673-0772-7/12/$31.00 ©2012 IEEE

2. Preliminaries

System Models We first introduce the real-time task model considered in this paper. The task set Γ consists of n sporadic tasks, denoted as $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ 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 arriving time of τ_i. c_i is the *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. We also assume that the actual execution time is available when a task arrives.

The thermal model used in this paper is similar to the one that has been used in the similar researches (e.g. [4, 19, 3, 7, 6, 22]). Specifically, $T(t)$ and T_{amb} are the chip temperature and ambient temperature respectively. $P(t)$ denotes the power consumption (in $Watt$) at time t, and R, C are the thermal resistance (in $^oC/Watt$) and thermal capacitance (in $J/^oC$) respectively. Then, $RC\frac{dT(t)}{dt} = RP(t) + (T(t) - T_{amb})$. We can scale T such that T_{amb} is zero and then we have

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

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

We assume that the processor can provide n different supply voltage levels with each voltage level associated with a different clock frequency. We further assume that for 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 the 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 $P_{dyn} = C_{load} f v^2$, where C_{load} is the equivalent load capacitance, f is the clock frequency and v is the supply voltage. Assuming working frequency is proportional to supply voltage, we have $P_{dyn} = C_2 v_k^3$ [17], where v_k is the kth supply voltage level and C_2 is a constant. The leakage power is temperature dependent and can be calculated as $P_{leak} = N_{gate} \cdot I_{leak} \cdot v$, where N_{gate} represents the number of gates. I_{leak} is the leakage current, which can be formulated by a non-linear exponential equation as [13]

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

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

As leakage current changes super linearly with temperature [14], the leakage power for the processor running in mode k can be effectively estimated as $P_{leak}(k) = C_0(k)v_k + C_1(k)Tv_k$ [4], where $C_0(k)$ and $C_1(k)$ are constants. As we can see from the leakage model, the leakage power depends on both the supply voltage and temperature. The total power consumption at the kth processor mode is thus $P(k) = C_0(k)v_k + C_1(k) \cdot Tv_k + C_2 v_k^3$. Accordingly, the temperature dynamics at the speed level k can be formulated as

$$\frac{dT(t)}{dt} = A(k) - B(k)T(t) \tag{3}$$

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 equation (3), the ending temperature can be formulated as below:

$$\begin{aligned} T_e &= \frac{A(k)}{B(k)} + (T_0 - \frac{A(k)}{B(k)})e^{-B(k)(t_e - t_0)} \\ &= G(k) + (T_0 - G(k))e^{-B(k)(t_e - t_0)}. \end{aligned} \tag{4}$$

Based on our system models, a formal problem definition can be stated as,

PROBLEM 1. *Given a hard real-time task set Γ consisting of N sporadic tasks $\{\tau_1, \tau_2, ..., \tau_n\}$, develop an on-line speed schedule under the* EDF *policy such that the overall energy consumption is minimized.*

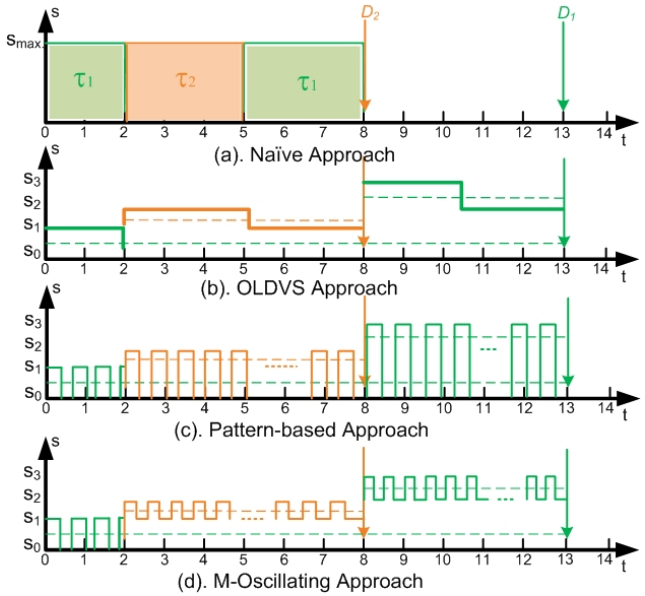

Figure 1. Different Speed Scheduling Approaches under *EDF* policy

Table 1. Normalized energy consumptions of four scheduling approaches

Approach	E_{dyn}	↘	E_{leak}	↘	E_{total}	↘
Pattern	0.31	-	0.69	-	1	-
OLDVS	0.28	9.6%	0.66	4%	0.94	6%
M-Osc.	0.28	9.6%	0.55	20%	0.83	17%
	E_{dyn}	↗	E_{leak}	↗	E_{total}	↗
Naive	1.38	440%	3.56	510%	4.94	494%

Motivational Example Let us consider a sporadic task set consisting of two tasks, i.e. $\tau_1 \equiv \{0, 5, 11, 11\}$ and $\tau_2 \equiv \{2, 3, 6, 6\}$, scheduled according to *EDF*. A simple naive approach, as shown in Figure 1(a), is to execute the task with the maximal speed and turn to sleep mode upon the finish of task execution. Another approach, i.e. the *on-line DVS* (*OLDVS*) approach [12], as illustrated in Figure 1(b), is to use two neighboring speeds of the *constant speed* (dotted lines), i.e. the speed that can finish the workload exactly at the deadline. Here, it is worth of mentioning that if the *constant speed* is less than the lowest speed level provided by the processor, then the lowest speed and the idle mode, i.e. $s = 0$, will be selected as two neighboring speeds. The third approach, i.e. the *Pattern-Based* approach [19], as illustrated in Figure 1(c), 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 forth approach, *M-Oscillating* approach [4], as illustrated in Figure 1(d), chooses the same speed levels as the *OLDVS* approach. It then divides the high speed and the low speed interval each into m divisions and oscillates between the two speeds to execute a task. This approach is shown to be very effective to reduce the peak temperature as well as the overall energy consumption [7].

Table 1 shows the normalized energy consumptions (w.r.t. the overall energy of the *Pattern-Based* approach) based on a processor model built based on the 65nm technology (more details can be found in Section 4). We can see that in this example, the naive approach uses only the maximal speed level for task execution (no *DVS*), and as a result, the energy consumption is several times higher than the other three approaches. As shown in Table 1, compared with *Pattern-Based* approach, the *OLDVS* saved 6% of the overall energy consumption. The energy saving comes from two aspects. First, using two neighboring speeds instead of the single speed reduces the dynamic energy consumption. Second, the reduced dynamic power consumption di-

rectly translates to lower temperature which also helps to save the leakage energy. Also, from Table 1 we can see that although both the *OLDVS* approach and the *M-Oscillating* approach consume the same amount of dynamic energy, the leakage energy consumptions by different policies are quite different. Additional 11% energy can be saved by *M-Oscillating* approach as a result of the optimized temperature achieved by speed oscillating.

One major challenge for on-line energy minimization scheduling is, with the complex interactions between the leakage power and the temperature, how to quickly and accurately estimate the energy consumption under different running conditions and schedules. One intuitive approach is to divide the schedule into a large number of small intervals within which the temperature can be treated as a constant. However, this approach can be extremely time consuming to ensure an accurate energy estimation. In our approach, we incorporate *M-Oscillating* algorithm into the *OLDVS* algorithm to form an efficient on-line scheduling algorithm for overall energy minimization. We also develop an analytical method that can quickly and accurately estimate the energy consumption for different schedules under different running conditions. The details of our approach is introduced below.

3. Our approach

In this section, we first introduce our on-line energy estimation method in detail. Based on this method, we present our algorithm to minimize the overall energy when scheduling a sporadic task set under *EDF* policy.

3.1 On-line energy estimation

The complexity of accurate energy estimation is due to the fact that the leakage energy depends not only on supply voltage but also on temperature. One straightforward method, similar to that in [15], is to calculate the leakage power by dividing the interval into small sub-intervals, within which one can safely assume the temperature (and thus the leakage power) as constant. With the circuit level model (e.g. equation (2)), this method can estimate total energy consumption accurately as long as the sub-interval is sufficiently small. However, the computational cost can be very sensitive to the accuracy of this approach. Another more efficient approach, i.e. [7], is to model the overall energy consumption as the integral of the temperature curve. A closed-form equation is derived based on the duration and the speed level being applied during a given scheduling interval. According to [7], this method can obtain the energy consumption with good accuracy, i.e. with 4.8% maximum relative error, while in terms of computational complexity, it can achieve up to four orders of magnitude of speedup upon the previous approach [15]. However, the closed-form energy formulation proposed in [7] can only be applied when the processor reaches thermal steady state, i.e. the temperature curve starts to change following a periodic pattern. It becomes less effective if the schedule starts from an arbitrary temperature. To find an accurate energy estimation method with low computational costs, in this subsection, we derive a novel energy estimation equation that can target a more general scenario rather than the thermal steady state. The applicability of the energy estimation equation is significantly improved while at the same time the efficiency and accuracy of the existing technique [7] is preserved.

Consider an *M-Oscillating* schedule and the corresponding temperature curve which are depicted in Figure 2. In this example, the processor is frequently switched between two neighboring speeds, i.e. S_i and S_j ($i < j$). Assuming there are totally $2m$ times mode switchings and in each of the m division, i.e. one high speed interval followed by a low speed interval, the durations that the processor spends in ith and jth mode are represented by t_{im} and t_{jm} respectively. 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 define that the processor is in mode μ during each low-high transition and in mode ν during each high-low transition even though they are representing the same idle mode. The notation of $T(t_M^{\alpha})$ can be interpreted as the starting temperature of the interval when applying processor mode M in the αth division.

If the processor runs in mode k for the interval $[t_0, t_e]$. The temperature varies following equation (1). By integrating both sides of

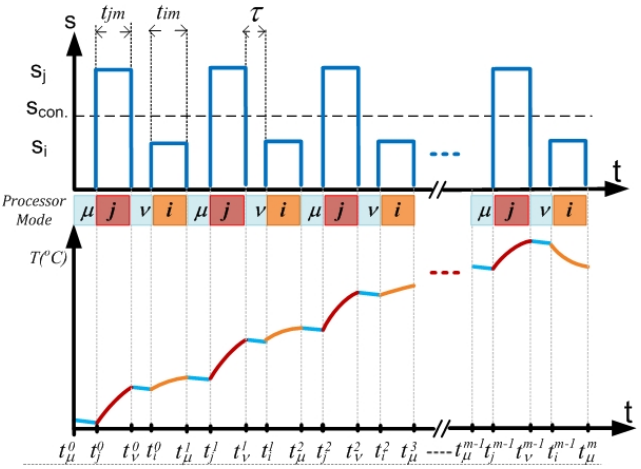

Figure 2. M-Oscillating and the corresponding temperature curve

equation (1), we have

$$\begin{aligned}
\int_{t_0}^{t_e} \frac{dT(t)}{dt}\cdot dt &= a\cdot\int_{t_0}^{t_e} P(t)dt - b\cdot\int_{t_0}^{t_e} T(t)dt \\
T(t_e)-T(t_0) &= a\cdot E(t_0,t_e) - b\cdot\int_{t_0}^{t_e} T(t)dt \\
E(t_0,t_e) &= 1/a\cdot(T(t_e)-T(t_0)+b\cdot\int_{t_0}^{t_e} T(t)dt) \qquad (5)
\end{aligned}$$

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.

Applying equation (5) to each speed interval in an *M-Oscillating* schedule, we have

$$\begin{aligned}
E(t_\mu^0,t_j^0) &= 1/a\cdot(T(t_j^0)-T(t_\mu^0)+b\cdot\int_{t_\mu^0}^{t_j^0} T(t)dt) \\
E(t_j^0,t_\nu^0) &= 1/a\cdot(T(t_\nu^0)-T(t_j^0)+b\cdot\int_{t_j^0}^{t_\nu^0} T(t)dt) \\
&\cdots \\
E(t_\nu^{m-1},t_i^{m-1}) &= 1/a\cdot(T(t_i^{m-1})-T(t_\nu^{m-1})+b\cdot\int_{t_\nu^{m-1}}^{t_i^{m-1}} T(t)dt) \\
E(t_i^{m-1},t_\mu^m) &= 1/a\cdot(T(t_\mu^m)-T(t_i^{m-1})+b\cdot\int_{t_i^{m-1}}^{t_\mu^m} T(t)dt)
\end{aligned} \qquad (6)$$

Sum up the above $4m$ equations we get

$$\begin{aligned}
E_{sum} &= 1/a\cdot(T(t_\mu^m)-T(t_\mu^0)+b\cdot(\sum_{x=0}^{m-1}\int_{t_\mu^x}^{t_j^x} T(t)dt \\
&+ \sum_{x=0}^{m-1}\int_{t_j^x}^{t_\nu^x} T(t)dt + \sum_{x=0}^{m-1}\int_{t_\nu^x}^{t_i^x} T(t)dt + \sum_{x=0}^{m-1}\int_{t_i^x}^{t_\mu^{x+1}} T(t)dt)) \qquad (7)
\end{aligned}$$

where E_{sum} is the total energy consumption that we want to find. $T(t_\mu^m)$ and $T(t_\mu^0)$ are the ending and initial temperature of the given *M-Oscillating* schedule respectively. Each of the summation term is corresponding to the summation of all m integral terms for mode μ, j, ν and i respectively. According to [7], if the kth speed level is applied during a segment, the corresponding integral term can be analytically expressed as

$$\int_{t_0}^{t_e} T(t)dt = G(k)\cdot(t_e-t_0)+1/B(k)\cdot(G(k)-T(t_0))(e^{-B(k)(t_e-t_0)}-1) \qquad (8)$$

where $G(k)$ and $B(k)$ are known *a priori* given a processor model, $T(t_0)$ is the initial temperature of the segment being evaluated.

From equation (7) and equation (8), it is not difficult to see that the overall energy consumption E_{sum} depends upon the temperature at each speed transition instants. To obtain these values, one straightforward way is to keep track of the entire *M-Oscillating* schedule and calculate the temperature at those specific time instants. However, the complexity of this method can be extremely high which makes our online algorithm simply ineffective. A more computational efficient method is to formulate the temperature values analytically. According to [16], the differences between the ending temperature and starting temperature of each sub-interval form a geometric series.

Let us first consider all "μ" intervals (the idle intervals before each high speed interval) . The starting temperature of the qth "μ" interval can be formulated as

$$T(t_\mu^q) = T(t_\mu^0) + \frac{(T(t_\mu^1) - T(t_\mu^0))(1-K_0^q)}{1-K_0} \tag{9}$$

where $T(t_\mu^1)$ is the ending temperature of the *first* division. $K_0 = exp(-B(j)t_{jm} - B(i)t_{im} - 2B_{idle}\tau)$, and $B_{idle} = B(\mu) = B(\nu)$.

Therefore, the summation of the initial temperature of the "μ" segments can be expressed as

$$\sum_{q=0}^{m-1} T(t_\mu^q) = m \cdot T(t_\mu^0) + \frac{T(t_\mu^1) - T(t_\mu^0)}{1-K_0}(m - \frac{1-K_0^m}{1-K_0}) \tag{10}$$

Note that in equation (10), given the initial temperature $T(t_\mu^0)$, we only need to evaluate the starting (ending) temperature of the *second* (*first*) sub-interval, e.g. $T(t_\mu^1)$, in order to obtain the summation of the initial temperature of all the μ intervals.

Now the summation of the integral terms for all the "μ" segments can be formulated as

$$\sum_{x=0}^{m-1} \int_{t_\mu^x}^{t_j^x} T(t)dt = m \cdot G_{idle} \cdot (\tau + \beta_\mu) - \beta_\mu \cdot \sum_{q=0}^{m-1} T(t_\mu^q) \tag{11}$$

where $\beta_\mu = 1/B_{idle} \cdot exp(-B_{idle}\tau - 1)$ and $G_{idle} = G(\mu) = G(\nu)$.

Similarly, the summations of the initial temperature of all high speed (j) intervals, ν idle intervals and low speed (i) intervals are

$$\begin{aligned}
\sum_{q=0}^{m-1} T(t_j^q) &= m \cdot T(t_j^0) + \frac{T(t_\mu^1) - T(t_\mu^0)}{1-K_0} K1 (m - \frac{1-K_0^m}{1-K_0}) \\
\sum_{q=0}^{m-1} T(t_\nu^q) &= m \cdot T(t_\nu^0) + \frac{T(t_\mu^1) - T(t_\mu^0)}{1-K_0} K2 (m - \frac{1-K_0^m}{1-K_0}) \\
\sum_{q=0}^{m-1} T(t_i^q) &= m \cdot T(t_i^0) + \frac{T(t_\mu^1) - T(t_\mu^0)}{1-K_0} K3 (m - \frac{1-K_0^m}{1-K_0})
\end{aligned} \tag{12}$$

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

Correspondingly, the summation of the integral terms for each high speed (j) intervals, ν idle intervals and low speed (i) intervals can be expressed as

$$\begin{aligned}
\sum_{x=0}^{m-1} \int_{t_j^x}^{t_\nu^x} T(t)dt &= m \cdot G(j) \cdot (t_{jm} + \beta_j) - \beta_j \cdot \sum_{q=0}^{m-1} T(t_j^q) \\
\sum_{x=0}^{m-1} \int_{t_\nu^x}^{t_i^x} T(t)dt &= m \cdot G_{idle} \cdot (\tau + \beta_\nu) - \beta_\nu \cdot \sum_{q=0}^{m-1} T(t_\nu^q) \\
\sum_{x=0}^{m-1} \int_{t_i^x}^{t_\mu^{x+1}} T(t)dt &= m \cdot G(i) \cdot (t_{im} + \beta_i) - \beta_i \cdot \sum_{q=0}^{m-1} T(t_i^q)
\end{aligned} \tag{13}$$

where $\beta_j = 1/B(j) \cdot exp(-B(j)t_{jm} - 1)$; $\beta_i = 1/B(i) \cdot exp(-B(i)t_{im} - 1)$

Therefore, given an *M-Oscillating* schedule with m divisions, the overall energy consumption can be formulated as

$$\begin{aligned}
\tilde{E}(m) &= 1/a \cdot (\frac{(T(t_\mu^1) - T(t_\mu^0)) \cdot (1-K_0^m)}{1-K_0} \\
&+ b \cdot (\sum_{x=0}^{m-1} \int_{t_\mu^x}^{t_j^x} T(t)dt + \sum_{x=0}^{m-1} \int_{t_j^x}^{t_\nu^x} T(t)dt \\
&+ \sum_{x=0}^{m-1} \int_{t_\nu^x}^{t_i^x} T(t)dt + \sum_{x=0}^{m-1} \int_{t_i^x}^{t_\mu^{x+1}} T(t)dt))
\end{aligned} \tag{14}$$

Note that, in equation (14), given an initial temperature, e.g. $T_{t_\mu^0}$, we only need to calculate the temperature of the time instants within the first division, e.g. t_j^0, t_ν^0, and t_i^0, to get the entire temperature information and as a result, the overall energy consumption can be calculated efficiently. We will use experiment to validate the proposed energy formula in Section 4.1.

3.2 The on-line scheduling algorithm

In this subsection, based on the proposed energy equation, we present our on-line leakage aware energy minimization scheduling algorithm by incorporating the concept of *OLDVS* ([12]) and *M-Oscillating* technique ([4]). We name the proposed algorithm as "*On-line M-Oscillating*" (*OLMO*) approach.

The appropriate speed level selection is contingent upon the proper choice of the *constant speed*. To guarantee the real-time constraint, we adopt the method in [12] to calculate the the *constant speed*. According to [12], for the 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 resumes after τ_k
  $WCCT_i = t + c_i$
  $WCRT_i = c_i$
  $WCRT_i = WCRT_j - s_j(t-l)$ /*1:previous context switch time*/

- τ_i preempts τ_j
  $WCCT_i = WCCT_i + WCCT_k - t_p$ /* 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 $s_{con} = \frac{WCRT_i}{WCCT_i - t}$.

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 high speed duration t_j can be formulated as $t_j = t_p \times \frac{S_{con} - S_i}{S_j - S_i}$ and $t_i = t_p - t_j$.

To incorporate the non-negligible transition overhead, we adopt the the method introduced in [7] to find the duration of the adjusted high speed (t_{jm}) and low speed (t_{im}) interval w.r.t. different m values. Specifically, $t_{im} = \frac{t_i}{m} - \tau - \delta$ and $t_{jm} = \frac{t_j}{m} - \tau + \delta$, where τ is the transition overhead and $\delta = \frac{(S_i + S_j) \cdot \tau}{S_j - S_i}$.

At each scheduling point, given the initial temperature, i.e. $T(t_\mu^0)$, the overall energy consumption, i.e. $\tilde{E}(m)$, can thus be formulated as a function of m (equation (14)). Our problem is therefore to minimize $\tilde{E}(m)$ subject to $m \le m_{Max} = \lfloor \frac{t_i}{\delta + \tau} \rfloor$ and the real-time constraint. We can simply use a sequential search to find the optimal value of m. Because of the simplicity of our energy estimation method, we can afford to search for the optimal m sequentially 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 process is the searching step for optimum m. Apparently the complexity of the searching algorithm is $O(m_{Max})$. The theoretical upper bound of m, e.g. $m_{Max} = \lfloor \frac{t_i}{\delta+\tau} \rfloor$, depends on the low speed duration of a given task, i.e. t_i, and the transition overhead τ. Based on the task and processor models 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 4.1. To further improve the efficiency of finding the m_{opt}, we can use some more elaborative searching algorithm such as those target at the unimodal functions.

Algorithm 1 On-Line M-Oscillating (OLMO) Algorithm

while context switch to task τ_i at time t **do**
  Calculate the constant speed $S_{con} = \frac{WCRT_i}{WCCT_i - t}$
  Find two nearest neighboring speeds S_i and S_j of the constant speed;
  Calculate ideal low/high speed duration t_i and t_j
  Identify upper bound of m:
  if $S_i \neq idle$ **then**
    $m_{Max} = \lfloor \frac{t_i}{\delta+\tau} \rfloor$
  else
    $m_{Max} = \lfloor \frac{t_i}{2\tau} \rfloor$
  end if
  Assign: $E_{min} = \infty$ and $m_{opt} = 0$
  for m=1:m_{Max} **do**
    if $S_i \neq idle$ **then**
      $t_{im} = \frac{t_i}{m} - \tau - \delta$;
      $t_{jm} = \frac{t_j}{m} - \tau + \delta$;
    else
      $t_{im} = \frac{t_i}{m}$
      $t_{jm} = \frac{t_j}{m}$
    end if
    Calculate $\tilde{E}(m)$; Eq. 14
    if $\tilde{E}(m) < E_{min}$ **then**
      $E_{min} \leftarrow \tilde{E}(m)$
      $m_{opt} \leftarrow m$
    end if
  end for
  Return: $\{S_i, S_j, t_{im}, t_{jm}, m_{opt}\}$
end while

4. Experimental results

In this section, we conduct a set of experiments to study the performance of the proposed on-line energy equation as well as our methods. We adopted the processor model similar to the one in [16] which is built based on the work by Liao et al. [13] using the 65nm technology. Specifically, we used equation (2) to compute the leakage currents for temperature from 20^oC to 140^oC with a step size of 5^oC, and supply voltage from 1.0 Volt to 1.3 Volt with a step size of 0.05V. These results were used to determine the temperature invariants $C_0(k)$ and $C_1(k)$ in our leakage model through curve-fitting. The thermal constants are chosen according to [18]. The ambient temperature is set to 25^oC. The timing penalty of speed transition τ is set to be 5ms and the transition energy overhead E_{sw} is 0.01J [19].

4.1 The accuracy and efficiency of proposed on-line energy estimation equation

We first study the accuracy and efficiency of the proposed energy estimation method. We generated five groups of tasks with different deadlines(D), e.g. 5s, 10s, 20s, 50s and 100s. Each group consists of 1000 tasks with randomly generated *WCET* varying from $0.5D$ to $0.95D$. For each task, we calculated the energy consumption scheduled by the *M-Oscillating* scheme with m iterated from 1 to m_{max}. We assumed that the initial temperature is the same as the ambient temperature. Three approaches were used to obtain the energy values.

- The proposed online periodic schedule energy estimation method, i.e. equation (14). We refer this method as *PSE*.
- The single interval based energy calculation method. Specifically, we use equation (5) derived in [7] to calculate the energy consumption of each interval and apply equation (4) to trace the temperature. We call it *SIE* method.
- The energy calculation method similar to the one in [15].

According to [7], for a single interval, calculating the energy by using equation (5) has already demonstrated significant speed up over the method in [15], while the latter one has better performance in terms of accuracy. Therefore, in this experiment, to evaluate the accuracy of *PSE* method, we still compared the energy values calculated by *PSE* method with the ones obtained by the method in [15]. The *SIE* method is used as the baseline approach to show the additional speedup achieved by the *PSE* method. The average CPU time (A.C.T.) of each task group achieved by different approaches as well as the averaged relative error (A.R.E) and speedup of the proposed energy estimation method were recorded and summarized in Table 2.

As can be seen in Table 2, our experimental results clearly show the superiority of our *PSE* method. On one hand, the energy consumption estimated by using *PSE* method well matches the one obtained by the approach in [15] with the relative error kept within 4.1%. The CPU time, on the other hand, is significantly reduced compared with *SIE* method, i.e. up to a maximum of 210X of speedup or over 94X 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, *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 division of *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.

4.2 The energy saving performance

We next evaluate the energy saving performance of our proposed algorithm by comparing with the *OLDVS* approach [12] and the *Pattern-Based* approach [19] as illustrated in Figure 1(b) and (c). 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, with range of $[50, 500]$. 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 generated randomly for each job with even 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 point, $1,000$ task sets will be generated bounded by the corresponding system utilization. The overall energy consumptions by three different approaches are collected and then normalized to the ones achieved by the *Pattern-Based* approach.

The results are plotted in Figure 3. Figure 3 shows 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. save 11% and 23% energy at system utilization equal to 0.5 and 1.0 respectively and 14% on average). Compared with *OLDVS*, *OLMO* achieves 10% energy saving on average, and the energy saving also increases with the increment of system utilization. Compared with both *Pattern-Based* and *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

Table 2. Comparison of the accuracy and efficiency of energy calculation equation

Deadline(s)	Ave.m_{max}	A.C.T.([15])	A.C.T.(SIE)	A.C.T(PSE)	Ave.Speedup (PSE vs.SIE)	A.R.E (PSE vs.[15])
5	63	4.68s	0.034s	0.00092s	36X	4.1%
10	117	9.63s	0.093s	0.0019s	49X	3.4%
20	248	18.38s	0.30s	0.0037s	81X	2.8%
50	625	46.2s	1.81s	0.014s	129X	1.9%
100	1180	108s	6.03s	0.03s	210X	1.1%

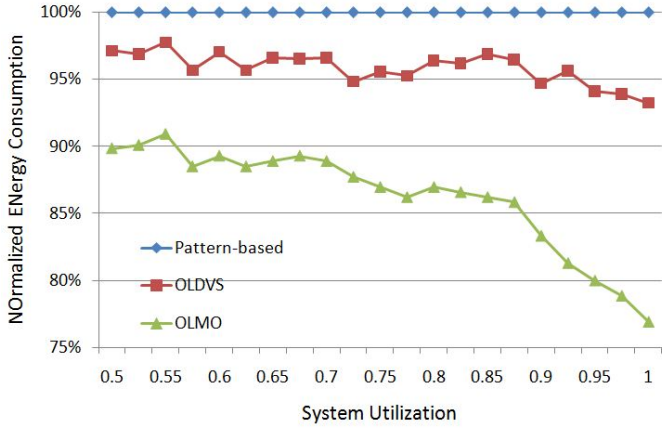

Figure 3. Normalized energy consumption of three scheduling approaches

temperature curve directly translate to a reduced leakage energy consumption which is the major contributor of energy saving. Therefore, the proposed *OLMO* appears to have better energy reduction performance when system utilization is high.

5. Conclusions

In this paper, we introduce an analytical method to estimate energy consumption on-line with the leakage/temperature dependency taken into consideration. By incorporating this method with an existing scheduling approach, we develop an on-line scheduling algorithm to reduce the overall energy consumption for a hard real-time system scheduled according to the *EDF* policy. Our experimental results show that the proposed energy estimation method can achieve up to 210X speedup compared with an existing approach while still maintaining high accuracy. In addition, with a large number of different test cases, the proposed method consistently outperforms two closely related researches in average by 10% and 14% respectively.

Acknowledgment

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

6. References

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

[2] M. Bao, A. Andrei, P. Eles, and Z. Peng. Temperature-aware idle time distribution for energy optimization with dynamic voltage scaling. In *DATE*, pages 21 – 27, 2010.

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

[4] V. Chaturvedi, H. Huang, and G. Quan. Leakage aware scheduling on maximal temperature minimization for periodic hard real-time systems. In *ICESS*, pages 1802–1809.

[5] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo. Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems. In *RTAS*, pages 408–417, 2006.

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

[7] H. Huang and G. Quan. Leakage aware energy minimization for real-time systems under the maximum temperature constraint. In *DATE*, 2011.

[8] H. Huang, G. Quan, and J. Fan. Leakage temperature dependency modeling in system level analysis. In *ISQED*, pages 447–452, 2010.

[9] ITRS. *International Technology Roadmap for Semiconductors*. International SEMATECH, Austin, TX., http://public.itrs.net/.

[10] R. Jejurikar, C. Pereira, and R. Gupta. Dynamic slack reclamation with procrastination scheduling in real-time embedded systems. *DAC*, pages 111 – 116, 2005.

[11] P. Kumar and L. Thiele. Cool shapers: Shaping real-time tasks for improved thermal guarantees. In *DAC*, pages 468–473, 2011.

[12] C.-H. Lee and 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*, pages 319 – 335, 2004.

[13] W. Liao, L. He, and K. Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(7):1042 – 1053, 2005.

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

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

[16] G. Quan and V. Chaturvedi. Feasibility analysis for temperature-constraint hard real-time periodic tasks. *IEEE Transaction on Industrial Informatics*, 6(3):329–339, 2010.

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

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

[19] C.-Y. Yang, J.-J. Chen, L. Thiele, and T.-W. Kuo. Energy-efficient real-time task scheduling with temperature-dependent leakage. In *DATE*, pages 9–14, 2010.

[20] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *FOCS*, pages 374–382, 1995.

[21] H. Yu, B. Veeravalli, and Y. Ha. Leakage-aware dynamic scheduling for real-time adaptive applications on multiprocessor systems. In *DAC*, pages 493–498, 2010.

[22] S. Zhang and K. S. Chatha. Thermal aware task sequencing on embedded processors. In *DAC*, pages 585 – 590, 2010.