JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 25, 251-272 (2009)

Energy Efficient Intra-Task Dynamic Voltage Scaling for Realistic CPUs of Mobile Devices*

CHIEN-CHUNG YANG, KUOCHEN WANG, MING-HAM LIN AND POCHUN LIN
Department of Computer Science
National Chiao Tung University
Hsinchu, 300 Taiwan

It is not energy efficient to run a CPU at full speed all the time for all kinds of tasks in mobile devices. This paper proposes two energy efficient intra-task *dynamic voltage scaling* (DVS) algorithms for CPUs. There are three main contributions in this paper. Firstly, unlike the tedious derivation in PACE [2], we have derived the same optimal speed schedule with minimal energy consumption in a *discrete* and elegant way by using the *Lagrange multiplier* procedure. Secondly, the CPU model assumed in PACE is ideal, meaning that such a CPU supports all possible frequencies/voltage levels. We call such CPUs as *ideal CPUs*. In reality, CPUs only support a limited set of frequency/voltage levels, and we call this kind of CPUs as *realistic CPUs*. Thirdly, since energy consumption is not a simple function of frequency, it is more reasonable to transform the original nonlinear programming problem to the *Multiple-Choice Knapsack Problem* (MCKP). Since the problem can be described by a multistage graph, we used dynamic programming to derive an *Optimal Schedule for Realistic CPUs* (OSRC) with minimal energy consumption for realistic CPUs by using actual power consumption specifications of realistic CPUs. Considering potential computation and transition overheads, we have also proposed a low overhead OSRC (LO-OSRC), which restricts the change of CPU frequency/voltage to only once in the speed schedule. By using actual data from the power consumption specifications of two classical CPUs for evaluation, experimental results have shown that the energy saving of the proposed OSRC (LO-OSRC) is up to 10.3% (9.4%) better than that of PACE for realistic CPUs.

Keywords: CPU, energy efficient, intra-task, dynamic voltage scaling, mobile device, real time

1. INTRODUCTION

With the advent of multimedia eras, mobile devices such as cell phones and personal digital assistants (PDAs) require powerful CPUs to handle a variety of multimedia applications as fast as possible. However, mobile devices are powered by batteries for mobility needs. In each battery operated system, we need effective power management to use energy efficiently to extend battery life. Such battery operated systems often build in embedded systems for special purposes. Different purposes need different power management schemes. In communication embedded systems, like cell phones, they often consume a lot of energy in radio interfaces. If the radio interface still works in full power when there is no data communicating, it wastes energy. In wireless sensor embedded sys-

Received February 27, 2007; revised August 7, 2007 & February 27, 2008; accepted August 20, 2008.
Communicated by Yao-Wen Chang.
* The insightful comments of the reviewers help improve the quality of the paper. This work was supported by the NCTU EECS-MediaTek Research Center under grant Q583 and the National Science Council under grant No. NSC96-2628-E-009-140-MY3. The authors would like to thank Jheng-Ming Chen for his help on the preparation of the paper.

tems, they often become active once in a week, or even once a month. To save energy, the basic method is to change a sensor node from active to sleep state when it is not used.

In high-speed embedded systems, they use high-speed CPUs (in contrast to other embedded systems) to handle multimedia applications. However, the faster the CPU is, the more energy it consumes. Note that CPU frequency is directly proportional to the voltage applied, and energy is proportional to the square of the voltage applied [1]. The basic idea to save energy by adjusting CPU frequency is that we don't need a high CPU frequency to handle simple jobs, like text editing which requires lower CPU frequency than that for media playing. Even in the media playing, a device only needs to satisfy the user's requirement by using a proper CPU frequency. It has been shown that decreasing the CPU frequency linearly, the energy consumption can be reduced quadratically. Of course, such CPUs must support frequency and voltage scaling, like Intel *XScale* [15, 16], *AMD Duron* [17] and Transmeta *Crusoe* [18].

We can not decrease CPU speed by reducing frequency and voltage without considering system performance. System performance must be considered as well. The question here is how to choose a proper CPU speed that meets the system performance requirement. In the OS level point of view, giving a task with deadline d, and *worst case execution cycles* (WCECs) C, we can determine an optimal static speed, C/d, such that the task can be completed before deadline d, where WCEC is the maximum possible CPU cycles that the execution of an application may need [6, 37]. A real time system can be classified into *hard* and *soft*. In a hard real time system, all its tasks must finish before their deadlines; otherwise the system will crash. In a soft real time system, it allows a proportion of tasks to miss their deadlines and the system still works. Both kinds of real time systems need to know a task's workload and deadline, and the OS scheduler arranges each task to run to meet its deadline. Often a task's CPU requirement is assigned off-line, which is usually overestimated to avoid deadline miss. As a result when the task completes early, CPU idle time (called *slack time*) occurs. Note that off-line scheduling is a static scaling decision. At every instant of task start its execution time is known [47]. Using only off-line decisions, one can always guarantee the deadlines by using the worst case speed [48, 50, 51]. Most of intra-task DVS algorithms, such as the path-based method (*e.g.*, in [52]) and the stochastic method [11] (*e.g.*, PACE [2]), using off-line scheduling because the off-line scheduling can reduce execution overhead [49].

Dynamic voltage scaling (DVS), involving dynamic adjustments of the supply voltage and the corresponding operating clock frequency, has emerged as one of the most effective energy minimization techniques [10]. A DVS algorithm tries to adjust the CPU frequency of a system to reduce the CPU slack time and the system still meet hard or soft real time constraints for a task. However, it is hard to predict a task's actual workload before the task finishes due to conditional branches or loops. In other words, we never know an optimal DVS solution before the execution of a task is completed. Because of the indefinite task's workload, it is not energy efficient if the DVS algorithm uses a static frequency for each task [13]. In summary, the main issue of the DVS is how to adjust CPU frequency and voltage dynamically and still guarantee system performance.

In this paper, we address the *energy efficient intra-task voltage scaling* for *ideal* CPUs and *realistic* CPUs under the stochastic model, respectively. It is targeted at hard real time systems. The rest of the paper is organized as follows. Section 2 gives background and related work. Section 3 derives an optimal energy efficient intra-task DVS

algorithm for ideal CPUs in a discrete way. In section 4, we formulate the DVS problem for realistic CPUs as the MCKP (multiple-choice knapsack problem) and propose two energy efficient algorithms. One is OSRC, which can be solved by dynamic programming [31] since the problem can be described by a multistage graph. The other is LO-OSRC, which has low computation and transition overhead. The evaluation of the proposed algorithms compared with related work is given in section 5. Finally, concluding remarks are outlined in section 6.

2. BACKGROUND AND RELATED WORK

2.1 Dynamic Voltage Scaling

F. Yao *et al.* [30] first proposed a simple model of job scheduling in real time systems, which was further enhanced by B. Mochocki *et al.* [25] with consideration of the voltage transition overhead. Note that the transition overhead in terms of time and energy is in the range between 25 to 150 μsec and in the range of micro joules (for example 4 μJ in *lpARM*, which is a DVS-capable low-power processor) [6], respectively. Ishihara and Yasuura [21] formulated the voltage scheduling problem for a processor, which can use only a small number of discrete voltages, as an *integer linear programming* (ILP) problem. Andrei *et al.* [27] proved that the discrete voltage selection formulated as an ILP problem is indeed NP-hard; no optimal solution can be found in polynomial time.

The real time DVS algorithms can be classified into *inter-task* and *intra-task* DVS algorithms based on the granularity at which the voltage scaling is performed [34]. In intra-task DVS algorithms, the CPU frequency is changed dynamically during the execution of a task, while inter-task DVS algorithms keep it static in a task and change it across tasks. A survey of DVS techniques was presented in [36].

2.2 Inter-Task DVS

Inter-task DVS is carried out on a task-by-task basis and the voltage assigned to a task is unchanged during the whole execution of the task [10]. We often apply *earliest deadline first* (EDF) schedulers in periodical real time systems, and the CPU frequency and voltage are constant during the execution of a task. Pillai and Shin [8] proposed a look-ahead technique to determine future computation need and defer task execution. The look-ahead scheme tries to defer as much work as possible, and sets the operating frequency to meet the minimum work that must be done now to ensure all future deadlines are met [10], so the current running task always works under the lowest CPU frequency possible.

2.3 Intra-Task DVS

Intra-task DVS dynamically adjusts the operating voltage within an individual task boundary according to the execution behavior to reflect the changes of the required number of cycles to finish the task before the deadline [10]. Shin *et al.* [10] first proposed an intra-task DVS for hard real time applications by using remaining WCEC-based speed assignment, and voltage scaling decisions are made at compile time to reduce runtime overhead. But it is not energy efficient if we frequently change the CPU speed. Shin and

Kim [19] used the *average case execution path* (ACEP) as a reference path to decrease CPU frequency at the beginning of the task running. The algorithm exploits the fact that the ACEPs are more likely to be followed at run time than the WCEPs (worst case execution paths), and it optimizes the energy consumption for such hot paths. The main contribution of the algorithm is that it enhances the original intra-task DVS algorithm by exploiting the probability of each execution path, while guaranteeing the worst-case timing constraints [19]. However, Seo *et al.* [14] claimed that an ACEP- based algorithm does not always achieve minimum average energy consumption and proposed an optimal-case execution path based on probability.

Gruian [5] first proposed a stochastic model for intra-task DVS algorithms, which is better than reclaim-based intra-task ones [6, 10, 18]. A reclaim-based intra-task DVS is that the CPU frequency is calculated dynamically during the execution of a task. Lorch and Smith [2] proved that the optimal speed is inversely proportional to the cube root of the tail probability in Gruian's stochastic model. However, this optimal speed scheduler does not work energy efficiently in real world if the CPU only support a limited set of frequency and voltage levels.

3. OPTIMAL SPEED SCHEDULE FOR IDEAL CPUS

We first discuss how to derive an optimal speed schedule with minimal energy consumption for ideal CPUs. Although Lorch and Smith [2] have derived such an optimal schedule, called *PACE*, their proof is tedious and they did not take the restriction of a limited set of supported frequency/voltage levels and the actual power consumption of a CPU into account. In the following, we first formulate the optimal speed schedule problem as a constrained nonlinear programming problem, and derive an optimal speed schedule for ideal CPUs using the Lagrange multiplier procedure [24].

3.1 Stochastic Intra-Task DVS

The switching power dissipation (P_{sw}) in CMOS circuits is defined as [1]

$$P_{sw} = C_{eff} \cdot V_{dd}^2 \cdot f \tag{1}$$

where C_{eff} is the effective switching capacitance, V_{dd} is the supply voltage, and f represents the CPU frequency. Obviously, reduction of the supply voltage results in lower power consumption. The relation between circuit delay (T_d) and supply voltage (V_{dd}) is approximated by [1]

$$T_d \propto \frac{C_L V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \tag{2}$$

where C_L is the load capacitance, V_{th} is the threshold voltage, and α is the velocity saturation index which varies between 1 and 2. Because the clock frequency is proportional to the inverse of the circuit delay, the reduction of the supply voltage results in reduction of the clock frequency [1]. In [2, 4, 5, 13, 14, 20], a linear relationship between V_{dd} and f is

assumed. It would not be beneficial to reduce the CPU frequency without also reducing the supply voltage, because in this case the energy per operation would be constant [29]. The energy consumption (E) of executing a task is defined as [4]

$$E = P_{switching} \cdot t = C_{eff} \cdot V_{dd}{}^2 \cdot f \cdot \frac{C_a}{f} = C_{eff} \cdot V_{dd}{}^2 \cdot C_a \tag{3}$$

where t is the actual task executing time and C_a is the actual number of execution cycles for a task.

Note that we just introduced a model that assumes a linear relationship between voltage and frequency in order to derive an optimal speed schedule for ideal CPUs. However, in the simulation (section 5), we used real CPU frequency and voltage level specifications (as shown in Tables 1 and 2) for experiments. The specifications came from real CPU specifications.

In this paper we focus on the intra-task DVS. Note that intra-task DVS algorithms adjust the supply voltage within individual task boundary [11]. So we consider only one task at a time. In addition, off-line scheduling is adopted to avoid possible deadline miss and to reduce execution overhead, as explained in section 1. Assume the WCEC of a task T is C_w. We denote random variable X associated with the actual number of cycles used by task T over the interval (0, C_w). The cumulative distribution function of random variable X is $F(x) = P(X \leq x)$ and the tail distribution function [5] is $F^c(x) = 1 - F(x) = P(X > x)$. During the execution of a task, a stochastic DVS algorithm often assigns an appropriate CPU speed depending on how many cycles that the task has completed. Like the definition in [2, 5], we denote the speed by an ascending function $s(x)$. Because it is a linear relationship between voltage and clock rate [20], the expected energy consumption while executing a task is proportional to

$$\int_0^{C_w} F^c(x) \cdot C_{eff} \cdot s^2(x) \cdot x \, dx. \tag{4}$$

Note that the optimal schedule problem can be represented by a multistage graph, which depends on the probability distribution of a task's work requirement. We denote the cumulative distribution function of this work by $F(x)$, which is the probability that the task requires no more than x cycles of works [39]. On the other hand, $F(x)$ is the probability that the task actually ever gets done. So the tail distribution function $F^c(x)$ represents the probability that the task requires more than x cycles of works [39]. That is, it represents the probability that the task will not complete at the current stage. Therefore, the energy consumption is proportional to $F^c(x)$.

3.2 Simplifying the Derivation in PACE by Lagrange Multiplier Procedure

Because it is infeasible to change CPU frequency/voltage continuously, we assign a set of n possible execution cycles sorted from minimum to maximum, $\{C_1, C_2, \ldots, C_n\}$, and only change CPU frequency/voltage after a task executing C_i cycles, where $i = 0, \ldots, n - 1$ and $C_0 = 0$. In other words, the CPU frequency between C_{i-1} and C_i is constant. We denote the constant frequency assigned in partition $[C_{i-1}, C_i]$ as f_i and rewrite Eq. (4) in a discrete form:

$$\sum_{i=1}^{n} F^c(C_{i-1}) \cdot C_{eff} \cdot f_i^2 \cdot (C_i - C_{i-1}). \tag{5}$$

Our goal is to find an optimal speed schedule with minimum energy consumption of objective function (5) under time constraint

$$\sum_{i=1}^{n} \frac{C_i - C_{i-1}}{f_i} = d. \tag{6}$$

That is, this is a constrained nonlinear programming problem to find the minimum value of objective function (5) with constraint (6). We use the Lagrange multiplier procedure [24] to solve it. First we relax the constrained model into an unconstrained form by weighting constraints in the objective function with Lagrange multiplier v. The result is a Lagrangian function

$$L(f_1, f_2, \ldots, f_n, v) = \sum_{i=1}^{n} F^c(C_{i-1}) \cdot C_{eff} \cdot f_i^2 \cdot (C_i - C_{i-1}) + v\left(\sum_{i=1}^{n} \frac{C_i - C_{i-1}}{f_i} - d\right). \tag{7}$$

Solving the stationary point of the Lagrangian function

$$\frac{\partial L}{\partial f_1} = F^c(C_0) \cdot C_{eff} \cdot 2f_1 \cdot C_1 - \frac{C_1 v}{f_1^2} = 0,$$
$$\frac{\partial L}{\partial f_2} = F^c(C_1) \cdot C_{eff} \cdot 2f_2 \cdot (C_2 - C_1) - \frac{(C_2 - C_1)v}{f_2^2} = 0,$$
$$\cdots$$
$$\frac{\partial L}{\partial f_n} = F^c(C_{n-1}) \cdot C_{eff} \cdot 2f_n \cdot (C_n - C_{n-1}) - \frac{(C_n - C_{n-1})v}{f_n^2} = 0$$

we obtain the optimal solution f_i^*

$$f_1^* = \sqrt[3]{\frac{v_1^*}{2C_{eff}F^c(C_0)}},\ f_2^* = \sqrt[3]{\frac{v_2^*}{2C_{eff}F^c(C_1)}},\ \ldots,\ f_n^* = \sqrt[3]{\frac{v_n^*}{2C_{eff}F^c(C_{n-1})}}$$

where v_i^* is the optimal voltage for each i.

Note that the value of stationary point f_i^* is in proportion to $[F^c(C_{i-1})]^{-1/3}$. The result is the same as that in [2], which derived the result in a continuous and complex way. In contrast, we obtained the same result in a discrete and concise way.

Next, we try to solve the stationary point in Lagrangian function (7). First, the derivative of Lagrangian function (7) with respect to v is set to 0:

$$\frac{\partial L}{\partial v} = \sum_{i=1}^{n} \frac{(C_i - C_{i-1})}{f_i} - d = 0.$$

Because $[F^c(C_0)]^{-1/3} = 1$, we replace f_i^* with $f_i^* \cdot [F^c(C_{i-1})]^{-1/3}$, and substituting gives

$$d = \sum_{i=1}^{n} \frac{(C_i - C_{i-1})}{f_1^* \cdot [F^c(C_{i-1})]^{-1/3}}. \tag{8}$$

Thus f_1^* and the other f_i^*'s are solved.

To verify the computed stationary point is indeed a global minimum, the Hessian matrix of Lagrangian function L is defined as

$$\mathrm{H}(f_1, \ldots, f_n) = \begin{pmatrix} \frac{\partial L}{\partial f_1^2} & \cdots & \frac{\partial L}{\partial f_1 \partial f_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial f_n \partial f_1} & \cdots & \frac{\partial L}{\partial f_n^2} \end{pmatrix}$$

and

$$\mathrm{H}(f_1^*, \ldots, f_n^*) = \begin{pmatrix} 6C_{eff}F^c(C_0)C_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 6C_{eff}F^c(C_{n-1})(C_n - C_{n-1}) \end{pmatrix}.$$

Obviously, $\mathrm{H}(f_1^*, \ldots, f_n^*)$ is positive definite that means the stationary point of function L is an unconstrained local minimum and is optimal for the objective function (5) [24].

4. PROPOSED OPTIMAL SCHEDULE FOR REALISTIC CPUs (OSRC)

4.1 The Problem of the PACE for Realistic CPUs

The main problem of the optimal speed schedule, PACE [2], is that it targeted at ideal CPUs with an unlimited set of voltage levels. For realistic CPUs, like Intel *XScale* [15, 16], *AMD Duron* [17] and Transmeta *Crusoe* [18], only a limited set of voltage levels for corresponding available frequency levels is supported. This means that PACE is not applicable to these realistic CPUs. Another problem is described as follows. When calculating the optimal speed schedule, if we only consider the dynamic power based on Eq. (1) and neglect the static and leakage powers, it may actually result in less energy saving for the optimal speed schedule. To obtain the optimal speed schedule, it is more reasonable to use the CPU power consumption data from actual measurements.

Tables 1 and 2 show the valid CPU frequency, corresponding voltage and power consumption in Intel PXA255 and PXA270 CPUs, respectively [15, 16]. The power consumption values are different from the theoretical values calculated from Eq. (1), because most of DVS researches did not consider the short circuit and leakage power consumptions [1]. So it is more reasonable and feasible to adopt actual power consumption values instead of those obtained from Eq. (1) in DVS algorithms. In this paper, we used the power consumption values directly from Tables 1 and 2 for evaluation. For illustration, we use PACE and our proposed OSRC to find optimal schedules for realistic CPUs using two example tasks. Table 3 gives the specifications of these two tasks: tasks 1 and 2.

Table 1. Power consumption specification for Intel PXA255.

Frequency (MHz)	Voltage (V)	Power (mW)
33 (idle)	1.0	45
200	1.0	178
300	1.1	283
400	1.3	411

Table 2. Power consumption specification for Intel PXA270.

Frequency (MHz)	Voltage (V)	Power (mW)
13 (idle)	0.85	44.2
104	0.9	115
208	1.15	279
312	1.25	390
416	1.35	570
520	1.45	747
624	1.55	925

Table 3. Specifications of two example tasks.

Task	C_i (Mc)	$F^c(C_i)$	deadline (*ms*)
Task 1	{5, 15}	{1, 0.2}	50
Task 2	{5, 10, 15}	{1, 0.3, 0.1}	50

4.2 PACE Approach: Rounding the Frequencies Obtained from PACE to the Nearest Available Frequencies [38]

First, we take task 1 running at Intel PAX255 as an example. From Eq. (8), if the start-up speed of task 1 is c, and task 1 takes 5 million cycles (Mc) and keeps running, the CPU speed should be raised to $1.71c$. By Eq. (8), we found the start-up speed c is 217 MHz, and the speed after 5 Mc is 370 MHz. Because Intel PXA255 does not support these speed settings, it is reasonable to choose the upper nearest available frequencies to avoid deadline miss. After rounding the ideal frequencies, the speed schedule is: 300 MHz at start-up and 400 MHz after 5 Mc having been executed. We use function $p(f)$, which is $C_{eff} \cdot f^3$, to describe the power consumption under speed f, and $p(f)$ can be obtained based on Table 1. Now we rewrite Eq. (5) as

$$\sum_{i=1}^{n} F^c(C_{i-1}) \cdot p(f_i) \cdot \frac{(C_i - C_{i-1})}{f_i}. \tag{9}$$

From Eq. (9), the expected energy consumption is 6.75 *mJ* if the energy consumption during the idle time period is included.

4.3 OSRC Approach

Our OSRC is targeted at real CPUs with limited available frequencies. Denoting the available frequencies by a linear combination was often used [22, 25-28]. By rewriting objective function (5), the stochastic DVS model is formulated as the MCKP. That is, if a CPU has a limited set of m speeds, $\{s_1, s_2, \ldots, s_m\}$, it is better to formulate the original constrained nonlinear programming problem (Eqs. (6) and (9)) as the MCKP. We denote f_i, which is the CPU frequency after a task executes C_{i-1} cycles and remains static until the task executes C_i cycles, as a linear combination of s_i

$$f_i = s_1 \cdot x_{i,1} + s_2 \cdot x_{i,2} + \ldots + s_m \cdot x_{i,m} \tag{10}$$

where

$$\sum_{j=1}^{m} x_{i,j} = 1,$$
$$x_{i,j} \in \{0, 1\}, j = 1, \ldots, m.$$

Using the same concept of the linear combination of speeds, the expected energy consumption ($E(f_i)$) for executing ($C_i - C_{i-1}$) cycles under static CPU frequency $f_i \in \{s_1, \ldots, s_m\}$, can be rewritten as

$$E(f_i) = e(s_1) \cdot x_{i,1} + e(s_2) \cdot x_{i,2} + \ldots + e(s_m) \cdot x_{i,m} \tag{11}$$

where

$$e(s_i) = p(s_i) \cdot \frac{(C_i - C_{i-1})}{s_i}.$$

From Eq. (5), the expected energy consumption based on the intra-task DVS stochastic model is the sum of the expected energy consumption in partition $[C_{i-1}, C_i]$. Therefore, the expected energy consumption under a limited set of frequencies combinations is

$$\sum_{i=1}^{n} F^c(C_{i-1}) \cdot E(f_i) = \sum_{i=1}^{n} \sum_{j=1}^{m} F^c(C_{i-1}) \cdot e(s_j) \cdot x_{i,j}. \tag{12}$$

From Eq. (6), we denote the time consumption of CPU executing ($C_i - C_{i-1}$) cycles under speed s_i as $t_i(s_j)$, which equals to $(C_i - C_{i-1})/s_j$. Thus, the total time consumption by a task is the sum of $t_i(s_j)$, and the time constraint under a limited set of frequencies combinations is given by

$$\sum_{i=1}^{n} \sum_{j=1}^{m} t_i(s_j) \cdot x_{i,j} \le d. \tag{13}$$

Note that the equal ("=") relation in Eq. (6) has been replaced by the less than and equal to ("≤") relation. This is because under a limited set of frequencies, the sum of $t_i(s_j) \cdot x_{ij}$ is hard to fit the deadline exactly. Now the problem is formulated as the MCKP and x_{ij}'s are the only variables that we should solve:

Minimizing $\sum_{i=1}^{n}\sum_{j=1}^{m} F^c(C_{i-1})\cdot e(s_j)\cdot x_{i,j}$

Subject to $\sum_{i=1}^{n}\sum_{j=1}^{m} t_i(s_j)\cdot x_{i,j} \le d$

$\sum_{j=1}^{m} x_{i,j} = 1, i = 1, \ldots, n,$

$x_{i,j} \in \{0, 1\}, i = 1, \ldots, n, j = 1, \ldots, m.$

In an intra-task DVS schedule, the maximum number of possible frequency changes during the execution of a task is the number of CPU frequency levels minus one. Because the frequency levels of realistic CPUs [15-18] are only a few, we use *dynamic programming* to solve the MCKP [31]. From section 3.2, a task consists of n partitions, $\{p_1, p_2, \ldots, p_n\} = \{C_1, C_2 - C_1, \ldots, C_n - C_{n-1}\}$, where C_1 is the smallest, C_n is the largest, and p_i is the number of positive execution cycles ($C_i - C_{i-1}$). And the best feasible solution of r partitions $\{p_i, \ldots, p_n\}$, where $r = n - i + 1$, is denoted as S_r. Note that S_r is a subset of the partitions $\{p_1, p_2, \ldots, p_n\}$. The total execution cycles are $x_{1,j} \cdot p_1 + x_{2,j} \cdot p_2 + \ldots + x_{n,j} \cdot p_n$, where $x_{i,j} \in \{0, 1\}$; $i = 1, \ldots, n$, $j = 1, \ldots, m$. Using recursion to solve S_r, based on Eq. (13), we have

$$S_r = \begin{cases} \text{null} & \text{if deadline miss,} \\ \{x_{n,j}\} & r = 1, \text{ and } s_j \text{ is the minimal speed} \ge \dfrac{C_n - C_{n-1}}{d_1}, \text{ where } d_1 = d \\ S_{r-1} \cup \{x_{n-r+1,j}\} & r > 1, \text{ and } E_j(S_{r-1}) + F^c(C_{n-r})\cdot e(s_j) \text{ is a minimum,} \end{cases} \quad (14)$$

where if $x_{i,j} = 1$, it means p_i is selected using the j^{th} frequency level without deadline miss at S_r.

Note that deadline d_r of S_r, based on Eqs. (13) and (14), can be written as:

$$d_r = \begin{cases} d & \text{if } r = n, \\ d_{r+1} - t_{n-r}(s_j) & \text{otherwise.} \end{cases} \quad (15)$$

Because deadline d_{r-1} of S_{r-1} varies with a selected speed s_j in partition p_{n-r+1}, the energy consumption of S_{r-1}, based on Eq. (12), can be written as:

$$E_j(S_{r-1}) = \sum_{i=n,\, x_{i,j} \in S_{r-1}}^{n-r+2} F^c(C_{i-1})\cdot e(s_j). \quad (16)$$

And the time consumed by S_{r-1}, based on Eqs. (13) and (15), can be written as:

$$T(S_{r-1}) = \sum_{i=n-r+2}^{n} t_i(s_j) \le d_{r-1},\ j \in \{1, \ldots, m\}. \quad (17)$$

In S_r, under the following two conditions, deadline miss will occur:

(1) $S_{r\text{-}1}$ is null (it means no possible solution of $t_{n-r+1}(s_j)$ from S_r)
(2) $S_{r\text{-}1}$ is not null, but for all $j \in \{1, \ldots, m\}$, such that

$$T(S_{r\text{-}1}) + t_{n\text{-}r+1}(s_j) > d_r.$$

The proposed OSRC procedure is shown in Fig. 1. For better understanding of this procedure, we summarize the parameters used in Table 4. Although a recursive procedure is used in the OSRC procedure, the computation will not take too much time due to a limited set of available CPU frequencies. Because voltage scaling is computationally expensive compared to a conventional scheduler and it may hamper the possible energy saving [32], the size of n (a task's possible number of execution cycles) should be small. That is, if a CPU can use only a small number of possible execution cycles, the voltage scheduling can minimize the energy consumption under any time constraint [38]. In addition, the main idea of stochastic DVS is based on that if a periodic task's *actual execution cycles* (AEC), which are the actual number of cycles spent executing a task [6, 43], follows a distribution, the optimal speed schedule can save the most energy. Note that the distribution can be obtained offline and the optimal speed schedule needs to be calculated only once, and will be used for a long period of time. Based on the above observation, the runtime time of the stochastic intra-task DVS is not a concern. For example, for PACE, its runtime for an example case is at most 77μs per task [39]. The runtime is small compared to the deadlines in all cases, and considering that the computation can be done at the end of each task in anticipation of the next task, it should only delay the completion of a

```
Procedure OSR(s_r)
if (r == 1) then                                   /* S_1 means the last partition. */
   Find minimum speed s_j ≥ (C_n − C_{n-1})/d_1;
   if (found s_j) then
      return {x_{i,j}};
   else
      return null;
else
   for ((every s_j) and (T(s_{r-1}) + t_{n-r+1}(s_j) ≤ d_r)) do   /* find every possible s_j that satisfies
      temp_S[j] = OSRC(S_{r-1})                        deadline constrain d_r and solve S_{r-1}
   end for                                             based on t_{n-r+1}(s_j) */
   for (each temp_S[j]) do                          /* find the minimum energy of each
      Find j where E_j(S_{r-1}) + F^c(C_{n-r}) * e(s_j) is minimum;   possible solution */
      best_j = j;
   end for
   if (found best_j) then                           /* if the best solution is found, mark
      return {x_{n-r+1, best_j}} ∪ temp_S[best_j];     x_{n-r+1, best_j} = 1 for equation x_{1,j} · p_1 +
   else                                                x_{2,j} · p_2 + … + x_{n,j} · p_n */
      return null;
   end if
end if
```

Fig. 1. OSRC procedure.

Table 4. Nomenclature.

Parameter	Definition
C_i	the i^{th} possible execution cycles of $\{C_1, C_2, \ldots, C_n\}$
$F^c(C_i)$	the corresponding tail distribution function of C_i
S_r	the best r partitions from $\{p_i, \ldots, p_n\}$
s_j	the j^{th} frequency level of CPU
$x_{i,j}$	1 if selected , 0 otherwise
$t_i(s_j)$	$(C_i - C_{i-1})/s_j$
$T(S_{r-1})$	$\sum_{i=n-r+2}^{n} t_i(s_j), j \in \{1, \ldots, m\}$.
$e(s_j)$	the power consumption under speed s_j
$E_j(S_{r-1})$	the total energy consumption of S_{r-1} based on $t_{n-r+1}(s_j)$ from S_r
$best\_j$	the best solution at the j^{th} frequency level in each round
d_r	the deadline constraint of S_r

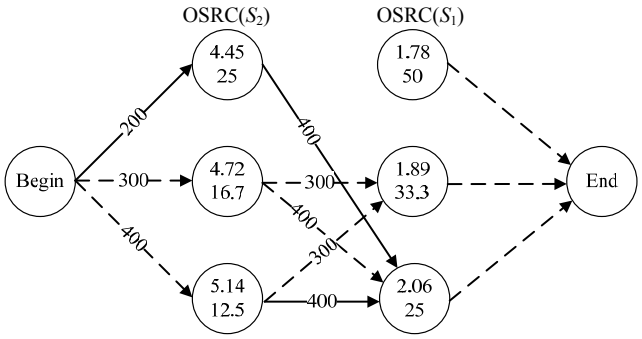

Fig. 2. The recursion graph for task 1.

task when there is no idle time before the next task starts. Therefore, the runtime overhead of the DVS algorithm could be ignored [39].

To demonstrate the merit of our OSRC over that of PACE, let's return to example task 1, as shown in Table 3, which consists of two partitions $\{p_1, p_2\} = \{C_1, C_2 - C_1\} = \{5\text{Mc}, 10\text{Mc}\}$, and the corresponding tail distribution functions are 1 and 0.2. Fig. 2 shows a recursion graph for task 1, which is a multistage graph. Note that each line is labeled with a selected frequency, and the dotted lines represent possible paths and the solid lines represent the optimal paths for each partition. Each node is labeled with two values: the upper value, representing energy consumption and the lower value, representing time consumed in each partition. The goal is to find a path which has a minimum sum of energy consumption, and the sum of time consumed must be less or equal to deadline d. We started at C_1 to find each feasible frequency level that satisfies deadline d_1. Based on the previous solution we found the next feasible solutions of $C_2, \ldots, C_n$ (the value of n in this example is 2). In the final round (the second round), we found three possible paths [(200, 400); (300, 300); (400, 300)] and the best one is (200, 400), which consumes the least energy (6.51 *mJ*).

Therefore, by using OSRC, the optimal speed schedule of example task 1 is {200

MHz, 400 MHz}, and the energy consumption is 6.51 *mJ*, which reduces 3.6% more energy than that by using PACE (6.75 *mJ* from section 4.2). Similarly, by using the OSRC algorithm, the optimal speed schedule of example task 2 is {200 MHz, 400 MHz, 400 MHz}, and the energy consumption is 6.5 *mJ*, which is still less than that by using PCAE (6.6 *mJ* from section 4.2).

4.4 LO-OSRC Approach

Although we can use the OSRC algorithm to find the optimal speed schedule, there are two problems in the OSRC. Firstly, The time complexity of the OSRC algorithm is $O(n^m)$, n is the number of possible partitions and m is the number of available CPU frequency levels. If n or m is large, the OSRC may waste a lot of time and energy. This is because the OSRC scheduler itself is a task that needs CPU time for execution. The energy consumption is proportional to CPU execution time. Therefore, if n or m is larger, the OSRC scheduler will execute longer and thus waste more energy. However, the size of n or m is usually small in realistic CPUs.

Secondly, the overhead of frequency/voltage transitions was not considered (PACE did not consider this as well). If there are too many frequency/voltage transitions in the optimal speed schedule of the OSRC, the transition overhead may result in energy inefficiency [44, 46]. If we restrict the change of CPU frequency/voltage to only once, not only these problems can be relieved but also the energy consumption is still very close to the optimal solution, which will be verified in the experiments (section 5). The proposed LO-OSRC procedure is shown in Fig. 3, and the time complexity of LO-OSRC is $O(nm^2)$.

```
Procedure Lo-OSRC( )
    for (every possible execution cycle Ci) do
        for (every possible CPU frequency sj) do
            set the speed between C0 and Ci to sj;
                find minimum speed sk ≥ (Cn − Ci)/din;
                /* din means the allowed time between Ci and Cn */
            if (the energy consumption of (Ci, sj, sk) < best_sol)
              (Ci, sj, sk) is best_sol;
            end if
        end for
    end for
```

Fig. 3. LO-OSRC procedure.

5. EVALUTION AND DISCUSSION

We evaluated the proposed optimal speed procedures for two realistic CPUs with different frequency/voltage levels: Intel PXA255 and PXA270. The two types of CPUs we used are ARM-based CPUs. The ARM architecture has been extremely successful in addressing a broad spectrum of embedded design requirements, and has evolved over a number of years to provide a foundation for a portfolio of core implementations offered by many different manufacturers. ARM is now the most widely used architecture for new

embedded designs [45]. Therefore, our evaluation results are applicable to such CPU architectures which support DVS.

The power consumption specifications for the two CPUs were shown in Tables 1 and 2. Note that we introduced a model that assumes a linear relationship between voltage and frequency in section 3.1 just for deriving an optimal speed schedule for ideal CPUs. However, in the following simulation, we used real CPU frequency and voltage level specifications (as shown in Tables 1 and 2) for experiments. The specifications came from two real CPU specifications. In addition, the power consumption in the idle state was also considered. Without loss of generality, the following parameter values are set. A single task's WCEC was set to the *worst case execution time* (WCET) × f_{max}, where WCET is set to 50 *ms* and f_{max} stands for the maximum CPU frequency. $\alpha \in \{0.2, 0.5, 0.8\}$, which is the ratio of *best case execution cycles/worst case execution cycles* (BCEC/WCEC). The reason to select these values for α is as follows. They reflect the variations of a task's AEC, which are between BCEC and WCEC, which follows a distribution. Since PACE assumed the normal distribution for AEC [2], for a fair comparison, we also assumed the same distribution for AEC. Such an assumption was also adopted in [5, 32, 39]. The reason for using the normal distribution was detailed in [39]. The mean and standard deviation were set to (WCEC + BCEC)/2 and (WCEC − BCEC)/6, meaning that 99.7 percent of the execution cycles falls into the interval [BCEC, WCEC] [13]. If a task's AEC does not fall into the above interval, it will be set to a closer limit: BCEC or WCEC. Because the speed schedule varies with the CPU utilization, we evaluated the task's *allowed execution time* (AET) that falls into the interval. [WCEC/f_{max}, WCEC/f_{min}] Note that AET represents the execution time that a task needs to finish its work, which is in the range of this closed interval. The simulation parameters (workloads) are listed in Table 5 [40-42].

Table 5. Simulation parameters.

Figure	CPU	WCEC(Mc)	BCEC(Mc)	AET(ms)	$N(\mu, \sigma^2)$
Fig. 4	PXA255	20	4	50~100	(12, 2.7)
Fig. 5	PXA270	31.2	6.24	50~300	(18.7, 4.2)
Fig. 6	PXA255	20	10	50~100	(15,1.7)
Fig. 7	PXA255	20	16	50~100	(18, 1.3)
Fig. 8	PXA270	31.2	15.6	50~300	(23.4, 2.6)
Fig. 9	PXA270	31.2	24.96	50~300	(28.1, 1.0)
Fig. 10	PXA270	31.2	6.24	50~300	(18.7, 4.2)
Fig. 11	PXA270	31.2	15.6	50~300	(23.4, 2.6)

We have implemented five schemes, including the proposed OSRC and LO-OSRC, for performance comparison:

- **WCE-stretch** [5]**:** The speed schedule assumes that the task will exhibit its worst case behavior, and choose the minimum static frequency.
- **PACE** [2]**:** The optimal speed schedule is derived by the theoretical formulation for ideal CPUs as described in section 4, and the unavailable frequencies are rounding up to the nearest available ones.

- **OSRC (proposed):** The speed schedule is derived by the proposed OSRC, as shown in Fig. 1.
- **LO-OSRC (proposed):** It is an improved OSRC, which has lower computation and transition overheads, as shown in Fig. 3.
- **LB (lower bound):** It is an oracle algorithm that knows the AEC in advance. Because of a limited set of CPU frequency/voltage levels, the unavailable frequencies were replaced by linear combinations of their two immediate frequencies in the LB scheme for maximum energy saving. Unlike other stochastic-related papers [2, 5], the expected energy consumption comparison is based on the actual CPU power specifications, as shown in Tables 1 and 2. And all schemes are normalized with respect to the WCE-stretch.

5.1 Impact of Voltage/Frequency Levels

Figs. 4 and 5 show the expected energy consumption comparison for two CPUs with different voltage/frequency levels: Intel PXA255 with three levels and PXA270 with six levels, respectively. α was set to 0.2. In Fig. 4, the sudden transition between AET/WCET = 1.2 and 1.4 in the LB curve is because the WCE-stretch speed schedule dropped the speed from 400 MHz to 300 MHz. When AET/WCET ≥ 2, the LB curve will be raised to 1 for the same reason. For the three levels CPU, Intel PXA255, the OSRC reduces CPU energy consumption between 0% and 10.2% with an average of 6.5%; the PACE reduces CPU energy consumption between − 1.2% and 9.9% with an average of 2.0%; the LB scheme reduces CPU energy consumption between 0% and 18.5% with an average of 10.8%.

For the six levels CPU, Intel PXA270, the OSRC reduces CPU energy consumption between 0% and 24.8% with an average of 15.9%; the PACE reduces CPU energy consumption between − 1.6% and 22.9% with an average of 5.6%; the LB scheme reduces CPU energy consumption between 0% and 26% with an average of 19.2%. From Figs. 4 and 5, we conclude that the more voltage/frequency levels, the more energy saving.

5.2 Impact of BCEC/WCEC (α) Ratio

We set α to 0.5 and then 0.8, and repeated simulations for these two types of CPUs. In Intel PXA255, as shown in Figs. 6 and 7, the OSRC can reduce 5.7% and 2.9% energy consumption in average (upper bound: 11.5% and 8.9%), respectively; the corresponding values for the PACE are 2.1% and 1.0%. In Intel PXA270, as shown in Figs. 8 and 9, the OSRC can reduce 13.4% and 6.7% energy consumption in average (upper bound: 17.3% and 13.3%), respectively; the corresponding values for the PACE are 4.4% and 2.0%. These experimental results show that for a smaller α (*e.g.*, α = 0.2), the average energy saving percentage is higher for both types of CPUs. However, for a larger α (*e.g.*, α = 0.8), the average energy saving percentage is smaller for both types of CPUs. That is, the OSRC and PACE schedules are closer to the WCE-stretch schedule for a larger α, especially for the 3 levels CPU. This is because a larger α will result in smaller slack time, which limits the aggressive frequency/voltage reduction in the OSRC and PACE schedules.

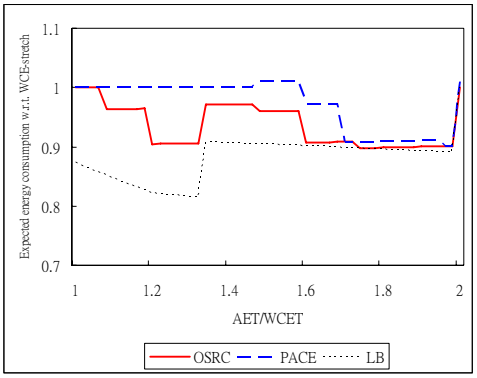

Fig. 4. The impact of three voltage/frequency levels on the expected energy consumption of Intel PXA255.

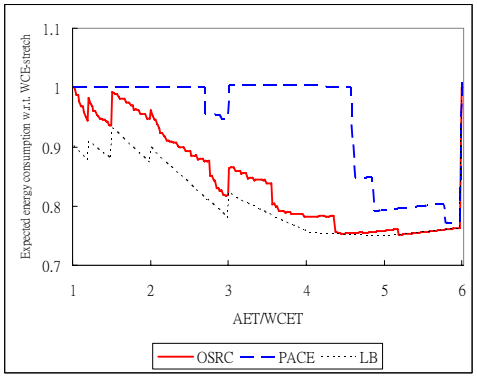

Fig. 5. The impact of six voltage/frequency levels on the expected energy consumption of Intel PXA270.

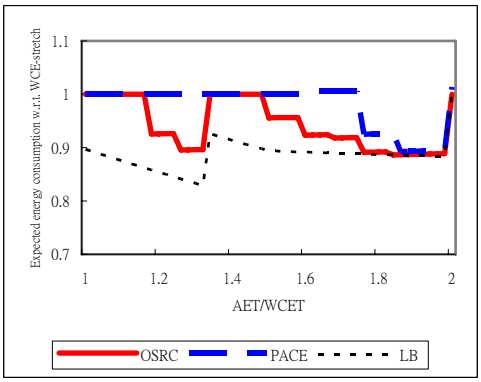

Fig. 6. The impact of α on expected energy consumption in PXA255 ($\alpha = 0.5$).

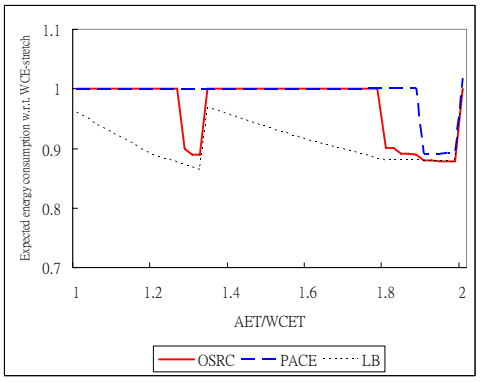

Fig. 7. The impact of α on expected energy consumption in PXA255 ($\alpha = 0.8$).

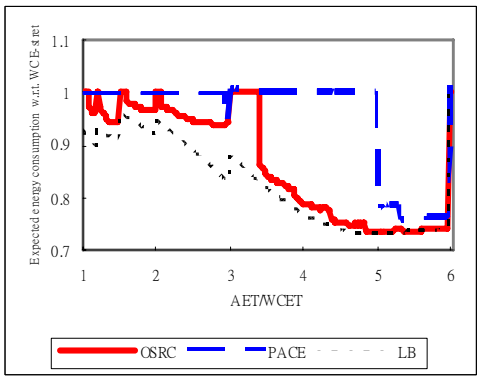

Fig. 8. The impact of α on expected energy consumption in PXA270 ($\alpha = 0.5$).

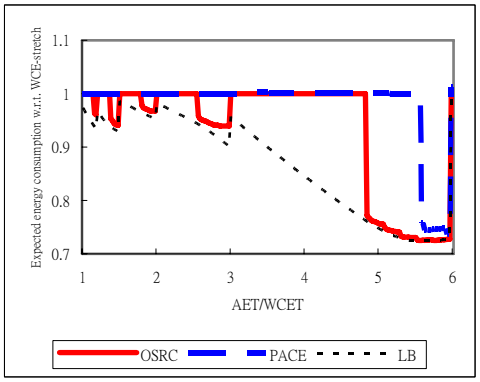

Fig. 9. The impact of α on expected energy consumption in PXA270 ($\alpha = 0.8$).

Table 6. Average energy saving percentage with respect to WCE-stretch.

Parameter (reference)	OSRC	PACE	LB
PXA255: $\alpha = 0.2$ (Fig. 4)	6.5%	2.0%	10.8%
PXA270: $\alpha = 0.2$ (Fig. 5)	15.9%	5.6%	19.2%
PXA255: $\alpha = 0.5$ (Fig. 6)	5.7%	2.1%	11.5%
PXA255: $\alpha = 0.8$ (Fig. 7)	2.9%	1.0%	8.9%
PXA270: $\alpha = 0.5$ (Fig. 8)	13.4%	4.4%	17.3%
PXA270: $\alpha = 0.8$ (Fig. 9)	6.7%	2.0%	13.3%

Note that the average energy saving percentage with respect to the WCE-stretch for each scheme in Fig. 4 through Fig. 9 is computed as (1 – *average of expected energy consumption with respect to WCE-stretch*). The results are summarized in Table 6. In terms of average energy saving percentage, the proposed OSRC is about 6.57% better than the PACE for realistic CPUs.

5.3 Comparison of OSRC and LO-OSRC

We have also evaluated the performance of LO-OSRC for Intel PXA270, as shown in Figs. 10 and 11, by setting α to 0.2 and 0.5, respectively. The LO-OSRC reduces 15.0% and 12.4% energy consumption in average (upper bound: 19.2% and 17.3%); in contrast, the corresponding values for the OSRC are 15.9% and 13.4%. The evaluation results for the four schemes based on Figs. 10 and 11 are summarized in Table 7. From Table 7, the average energy saving percentage in the LO-OSRC is very close to that in the OSRC, because the number of transitions in the optimal speed schedule obtained from the OSRC is usually small (observing from simulations, in most cases, the number of transitions is 2). If the overhead of computations and transitions are considered, the LO-OSRC will be more energy efficient than the OSRC. In summary, the energy saving of the OSRC (LO-OSRC) is up to 10.3% (9.4%) better than that of the PACE for realistic CPUs.

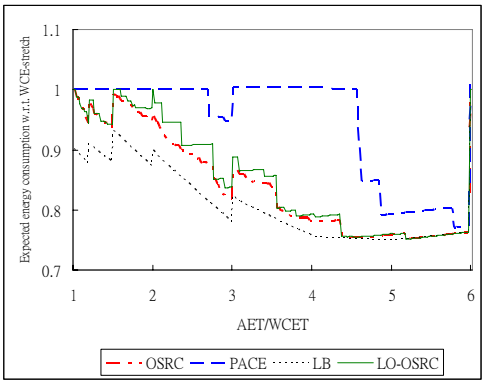

Fig. 10. The impact of α on expected energy consumption in PXA270 ($\alpha = 0.2$).

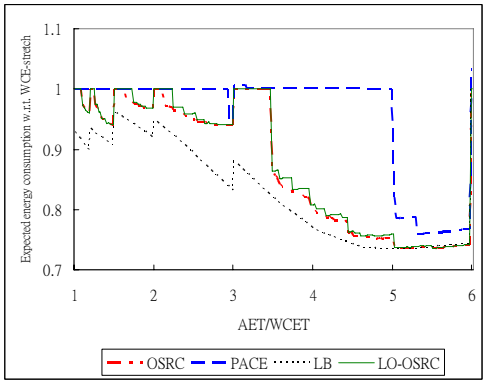

Fig. 11. The impact of αon expected energy consumption in PXA270 ($\alpha = 0.5$).

Table 7. Average energy saving percentage with respect to WCE-stretch.

Parameter (reference)	LO-OSRC	OSRC	PACE	LB
PXA270: $\alpha = 0.2$ (Fig. 10)	15.0%	15.9%	5.6%	19.2%
PXA270: $\alpha = 0.5$ (Fig. 11)	12.4%	13.4%	4.4%	17.3%

6. CONCLUSION

In this paper, we have derived an optimal speed schedule for ideal CPUs for hard real-time systems by the Lagrange multiplier procedure, in a simple and elegant way, compared to the PACE [2]. Because of limited available frequency/voltage levels in realistic CPUs, the optimal speed schedule for ideal CPUs can not be applied to realistic CPUs directly. To find an optimal speed schedule for a realistic CPU, we transform the original nonlinear programming problem into the Multiple-Choice Knapsack Problem (MCKP) based on the frequency/voltage levels and actual power consumption of realistic CPUs. With limited CPU frequency/voltage levels, the problem can be solved by using the OSRC procedure feasibly. Considering the potential computation and transition overheads, we have also proposed the LO-OSRC, which restricts the change of CPU frequency/voltage to only once in the speed schedule. To evaluate the merits of the proposed OSRC and LO-OSRC, the actual power consumption data of Intel PXA255 and PXA270 CPUs were used in the experiments. We have the following observations. Firstly, the experimental results show poor energy saving of the PACE for realistic CPUs, which is almost the same as that of the WCE-stretch. Using the OSRC for realistic CPUs, the average energy saving is very close to that of the low bound derived from the oracle algorithm. Secondly, we found that the CPU frequency/voltage levels affect the energy saving of the optimal speed schedule in the stochastic DVS model: the more the levels, the more the energy saving. Lastly, under the stochastic DVS model, our OSRC scheme can provide the best solution for realistic CPUs using dynamic programming. Evaluation results have shown that the energy saving of OSRC (LO-OSRC) is up to 10.3% (9.4%) better than that of PACE for realistic CPUs.

REFERENCES

1. B. Moyer, "Low-power design for embedded processors," *Proceedings of the IEEE*, Vol. 89, 2001, pp. 1576-1587.
2. J. R. Lorch and A. J. Smith, "PACE: A new approach to dynamic voltage scaling," *IEEE Transactions on Computers*, Vol. 53, 2004, pp. 856-869.
3. C. M. Krishna and Y. H. Lee, "Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems," *IEEE Transactions on Computers*, Vol. 52, 2003, pp. 1586-1593.
4. D. Zhu, D. Mosse, and R. Melhem, "Power-aware scheduling for AND/OR graph in real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, 2004, pp. 849-864.
5. F. Gruian, "Hard real-time scheduling for low-energy using stochastic data and DVS processors," in *Proceedings of International Symposium on Low Power Electronics*

and Design, 2001, pp. 46-51.
6. N. AbouGhazaleh, D. Mosse, B. Childers, R. Melhem, and M. Craven, "Collaborative operating system and compiler power management for real-time applications," in *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2003, pp. 133-141.
7. Y. Zhu and F. Mueller, "Feedback EDF scheduling exploiting dynamic voltage scaling," in *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2004, pp. 84-93.
8. P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, 2001, pp. 89-102.
9. E. Chan, K. Govil, and H. Wasserman, "Comparing algorithms for dynamic seed-setting of a low-power CPU," in *Proceedings of the 1st ACM International Conference on Mobile Computing and Networking*, 1995, pp. 13-25.
10. D. Shin, S. Lee, and J. Kim, "Intra-task voltage scheduling for low-energy hard real-time applications," *IEEE Design and Test of Computers*, 2001, pp. 20-30.
11. W. Kim, D. Shin, H. Yun, J. Kim, and S. Min, "Performance comparison of dynamic voltage scaling algorithms for hard real-time systems," in *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2002, pp. 219-228.
12. Laptop and Notebook Computers, Toshiba, http://www.toshibadirect.com/td/b2c/toshibanotebook.to.
13. H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, Vol. 53, 2004, pp. 584-600.
14. J. Seo, T. Kim, and C. Chung, "Profile-based optimal intra-task voltage Scheduling for hard real-time applications," in *Proceedings of the 41st Annual Conference on Design Automation*, 2004, pp. 87-92.
15. Intel, PXA255 Processor, Electrical, Mechanical, and Thermal Specification, 2004.
16. Intel, PXA270 Processor, Electrical, Mechanical, and Thermal Specification, 2004.
17. AMD Duron, http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_1260_1202,00.html.
18. LongRun2 Technology, http://www.transmeta.com/longrun2/index.html.
19. D. Shin and J. Kim, "A profile-based energy-efficient intra-task voltage scheduling algorithm for hard real-time applications," in *Proceedings of International Symposium on Low-Power Electronic and Design*, 2001, pp. 271-274.
20. M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proceedings of the 1st Symposium on Operating Systems Design and Implementation*, 1994, pp. 13-23.
21. T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processor," in *Proceedings of International Symposium on Low Power Electronic and Design*, 1998, pp. 197-202.
22. Y. Yu and V. K. Prasanna, "Resource allocation for independent real-time tasks in heterogeneous systems for energy minimization," *Journal of Information Science and Engineering*," Vol. 19, 2003, pp. 433-449.
23. T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, Vol. 6, 1995, pp. 109-133.

24. R. L. Rardin, *Optimization in Operations Research*, Prentice-Hall, New Jersey, 1998.
25. B. Mochocki, X. Hu, and G. Quan, "A realistic variable voltage scheduling model for real-time applications," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2002, pp. 726-731.
26. Y. Zhang, X Hu, and D. Chen, "Task scheduling and voltage selection for energy minimization," in *Proceedings of the 39th Conference on Design Automation*, 2002, pp. 183-188.
27. A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al-Hashimi, "Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems," in *Proceedings of the Conference on Design*, *Automation and Test in Europe*, 2004, pp. 518-523.
28. W. Kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors," in *Proceedings of the 40th Conference on Design Automation*, 2003, pp. 125-130.
29. T. L. Matrin and D. P. Siewiorek, "The impact of battery capacity and memory bandwidth on CPU speed-setting: a case study," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1999, pp. 200-205.
30. F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, 1995, pp. 374.
31. R. C. T. Lee, R. C. Chang, S. S. Tseng, and Y. T. Tsai, *Introduction to the Design and Analysis of Algorithms*, Unalis Corporation, 1999.
32. Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Proceedings of the 36th Design Automation Conference*, 1999, pp. 134-139.
33. A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi, "Quasi-static voltage scaling for energy minimization with time constraints," in *Proceedings of Design Automation and Test in Europe Conference*, 2005, pp. 514-519.
34. G. S. A. Kumar and G. Manimaran, "An intra-task DVS algorithm exploiting path probabilities for real-time systems," *ACM SIGBED Review*, Vol. 2, 2005, pp. 7-10.
35. R. D. Amstrong, D. S. Kung, P. Sinha, and A. A. Zoltners, "A computational study of a multiple-choice knapsack algorithm," *ACM Transactions on Mathematical Software*, Vol. 9, 1983, pp. 184-198.
36. T. Kim, "Application-driven low-power techniques using dynamic voltage scaling," in *Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2006, pp. 199-206.
37. D. Shin and J. Kim, "Intra-task voltage scheduling on DVS-enabled hard real-time systems," *IEEE Transactions on Computers*, Vol. 24, 2005, pp. 1530-1549.
38. T. Lshihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors" in *Proceedings of International Symposium on Low Power Electronics and Design*, 1998, pp. 197-202.
39. J. R. Lorch and A. J. Smith, "Improving dynamic voltage scaling algorithms with PACE," in *Proceedings of the ACM Special Interest Group on Measurement and Evaluation*, 2001, pp. 50-61.
40. B. Gorji-Ara, P. Chou, N. Bagherzadeh, M. Reshadi, and D. Jensen, "Fast and efficient voltage scheduling by evolutionary slack distribution," in *Proceedings of the*

Conference on Asia South Pacific Design Automation, 2004, pp. 659-662.
41. D. Shin and J. Kim, "Optimizing intra-task voltage using profile and data flow information," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, 2007, pp. 369-385.
42. H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, Vol. 53, 2004, pp. 584-600.
43. L. F. Leung, C. Y. Tsui, and W. H. Ki, "Minimizing energy consumption of hard real-time systems with simultaneous tasks scheduling and voltage assignment using statistical data," in *Proceedings of the Conference on Asia South Pacific Design Automation: Electronic Design and Solution Fair*, 2004, pp. 663-665.
44. A. Andrei, P. Eles, Z. Peng, M. T. Schmitz, and B. M. A. Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE Transactions on VLSI Systems*, Vol. 15, 2007, pp. 262-275.
45. W. Lyons, "Meeting the embedded design needs of automotive applications: RISC instruction set architecture," in *Proceedings of Design*, *Automation and Test in Europe*, Vol. 3, 2005, pp. 142-147.
46. H. Kweon, Y. Do, J. Lee, and B. Ahn, "An efficient power-aware scheduling algorithm in real time system," in *Proceedings of IEEE Pacific Rim Conference* on *Communications*, *Computers and Signal Processing*, 2007, pp. 350-353.
47. A. Baums, "Energy consumption optimization in hard real-time system CMOS processors," *Electronics and Electrical Engineering*, Nr. 4(68), 2006, pp. 19-22.
48. G. Yu, "Use of concurrency enhancement in off-line schedule construction" in *Proceedings of the 2nd Workshop on Parallel and Distributed Real-Time Systems*, 1994, pp. 32-37.
49. "Mixing off-line and on-line scheduling for Aperiodic operations with latency constraints," http://ralyx.inria.fr/2006/Raweb/aoste/uid42.html.
50. F. Gruian, "On energy reduction for hard real-time tasks with stochastic execution times," http://www.artes.uu.se/events/gsconf01/papers/ARTESCONF.pdf.
51. X. Zhong and C. Z. Xu "Energy-aware modeling and scheduling of real-time tasks for dynamic voltage scaling," in *Proceedings of the 26th IEEE International Real-Time Systems Symposium*, 2005, pp. 10.
52. D. Shin, J. Kim, and S. Lee, "Intra-task voltage scheduling for low-energy hard real-time applications," *IEEE Design and Test of Computers*, Vol. 18, 2001, pp. 20-30.

Chien-Chung Yang (楊建中) received the B.S. degree in Computer Science from the National Tsing Hua University, Taiwan, in 2001, and the M.S. degree in Computer and Information Science from the National Chiao Tung University, Taiwan, in 2005. He is currently a System Engineer in Sunplus Technology Co., Ltd. His research interests include power saving in mobile devices and image quality in LCD TV.

Kuochen Wang (王國禎) received the B.S. degree in Control Engineering from the National Chiao Tung University, Taiwan, in 1978, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Arizona in 1986 and 1991, respectively. He is currently a Professor in the Department of Computer Science, National Chiao Tung University and an Associate Director of the Computer and Network Center at the same university. He was a visiting scholar at the Department of Electrical Engineering, University of Washington from July 2001 to February 2002. From 1980 to 1984, he was a Senior Engineer at the Directorate General of Telecommunications in Taiwan. He served in the army as a second lieutenant communication platoon leader from 1978 to 1980. His research interests include wireless (ad hoc/sensor) networks, mobile computing, and power management for portable wireless/multimedia devices.

Ming-Ham Lin (林明翰) received the B.S. degree in Department of Management Information Systems from the National Chengchi University, Taiwan, in 2006, and the M.S. degree in Computer Science and Engineering from the National Chiao Tung University in 2008. His research interests include DVS scheduling algorithms for uni-processor and multi-core platforms.

Pochun Lin (林柏君) received the B.S. degree in Department of Applied Mathematics from the National Chung Hsing University, Taiwan, in 2006, and the M.S. degree in Network Engineering from the National Chiao Tung University, Taiwan, in 2008. He is currently a Software Engineer in ASUSTek Computer Inc. His research interests include low power for heterogeneous dual-core embedded real-time systems.